Fredrik Olsson

Bootstrapping Named Entity Annotation by Means of Active Machine Learning

Data linguistica
<http://hum.gu.se/institutioner/svenska-spraket/publ/datal/>

Editor: Lars Borin

Språkbanken • Språkdata
Department of Swedish Language
University of Gothenburg

21 • 2008

SICS dissertation series
<http://www.sics.se/publications/dissertations/>

SICS
Swedish Institute of Computer Science AB
Box 1263
SE-164 29 Kista
Sweden

50 • 2008

Fredrik Olsson

Bootstrapping Named Entity Annotation by Means of Active Machine Learning

A method for creating corpora

Gothenburg 2008

Data linguistica 21
ISBN 978-91-87850-37-0
ISSN 0347-948X

SICS dissertation series 50
ISRN SICS-D–50–SE
ISSN 1101-1335

Printed in Sweden by
Intellecta Docusys Västra Frölunda 2008

Typeset in LaTeX 2ε by the author

Cover design by Kjell Edgren, Informat.se

Front cover illustration:
Lågdagrar
by Fredrik Olsson ©

Author photo on back cover by Fredrik Olsson

ABSTRACT

This thesis describes the development and in-depth empirical investigation of a method, called BootMark, for bootstrapping the marking up of named entities in textual documents. The reason for working with documents, as opposed to for instance sentences or phrases, is that the BootMark method is concerned with the creation of corpora. The claim made in the thesis is that BootMark requires a human annotator to manually annotate fewer documents in order to produce a named entity recognizer with a given performance, than would be needed if the documents forming the basis for the recognizer were randomly drawn from the same corpus. The intention is then to use the created named entity recognizer as a pre-tagger and thus eventually turn the manual annotation process into one in which the annotator reviews system-suggested annotations rather than creating new ones from scratch. The BootMark method consists of three phases: (1) Manual annotation of a set of documents; (2) Bootstrapping – active machine learning for the purpose of selecting which document to annotate next; (3) The remaining unannotated documents of the original corpus are marked up using pre-tagging with revision.

Five emerging issues are identified, described and empirically investigated in the thesis. Their common denominator is that they all depend on the realization of the named entity recognition task, and as such, require the context of a practical setting in order to be properly addressed. The emerging issues are related to: (1) the characteristics of the named entity recognition task and the base learners used in conjunction with it; (2) the constitution of the set of documents annotated by the human annotator in phase one in order to start the bootstrapping process; (3) the active selection of the documents to annotate in phase two; (4) the monitoring and termination of the active learning carried out in phase two, including a new intrinsic stopping criterion for committee-based active learning; and (5) the applicability of the named entity recognizer created during phase two as a pre-tagger in phase three.

The outcomes of the empirical investigations concerning the emerging issues support the claim made in the thesis. The results also suggest that while the recognizer produced in phases one and two is as useful for pre-tagging as a recognizer created from randomly selected documents, the applicability of the recognizer as a pre-tagger is best investigated by conducting a user study involving real annotators working on a real named entity recognition task.

SAMMANFATTNING

Denna avhandling beskriver arbetet med att utveckla och utvärdera en metod, kallad BootMark, för att märka upp förekomster av namn i textdokument. Anledningen till att arbeta med dokument istället för, till exempel, meningar eller fraser, är att syftet med BootMark är att producera korpusar. Tesen är att BootMark tillåter en mänsklig annoterare att märka upp färre dokument för att kunna träna en namnigenkännare till en given prestanda, än vad som skulle ha behövts om namnigenkännaren tränats på ett slumpmässigt urval av dokument från samma korpus. Vidare är det tänkt att namnigenkännaren ska användas i ett förprocessningssteg i vilket namnen i de resterande texterna i korpusen först märks upp automatiskt för att sedan revideras manuellt. På så sätt ska annoteringsprocessen gå från att vara baserad på att den mänskliga annoteraren manuellt märker upp namn, till att annoteraren istället tar ställning till huruvida de automatiskt föreslagna annoteringarna behöver ändras.

BootMark-metoden består av tre faser. Den första fasen syftar till att producera en liten samling korrekt uppmärkta dokument att användas för att starta fas två. Den andra fasen nyttjar så kallad aktiv maskininlärning och är nyckeln till att BootMark kan reducera mängden dokument som en användare behöver annotera. I den tredje och sista fasen nyttjas den i fas två konstruerade namnigenkännaren för att omvandla annoteringsprocessen till en granskningsdito.

I samband med beskrivningen av BootMark identifieras fem praktiskt orienterade frågor vars gemensamma nämnare är att de kräver ett konkret sammanhang för att kunna besvaras. De fem frågorna rör: (1) namnigenkänningsuppgiftens och de därtill nyttjade maskininlärningsmetodernas karaktäristik, (2) sammansättningen av den mängd dokument som ska märkas upp i fas ett och ligga till grund för den aktiva inlärningsprocessen i fas två, (3) användandet av aktiv maskininlärning i fas två, (4) övervakningen av, och villkor för att automatiskt avbryta inlärningen i fas två, inklusive ett nytt intrinsiskt stoppvillkor för kommittébaserad aktiv inlärning samt (5) tillämpbarheten av namnigenkännaren som förprocessor i fas tre.

Utfallet av experimenten stöder tesen. Fas ett och två i BootMark bidrar till att reducera den mängd dokument en mänsklig annoterare behöver märka upp för att träna en namnigenkännare med prestanda som är lika bra eller bättre än en namnigenkännare som är tränad på ett slumpmässigt urval av dokument från samma korpus.

Resultaten tyder också på att även om namnigenkännaren som skapats i fas ett och två är lika lämplig att använda i ett förprocessningssteg som en namnigenkännare skapad genom träning på slumpmässigt utvalda dokument, så bör dess lämplighet undersökas genom användarstudier i vilka riktiga användare tar sig an en riktig namnigenkänningsuppgift.

ACKNOWLEDGEMENTS

These are the final lines I pin down in my dissertation. Undoubtedly they are the very first you'll read, and unless your name is mentioned below, this section is most probably the only one you'll ever care to digest. So here goes; shout outs and thank yous to:

Lars Borin. Supervisor. For immense knowledge, subtle puns and the willingness to share. For taking me in, setting things up, providing me with time, and seeing to it that my thesis actually ended up in print.

Björn Gambäck. Supervisor. For hiring me at SICS, finding time and funds, and encouraging me to enroll in graduate studies. For cheering, smearing, meticulously reading and commenting on whatever draft or idea I sent your way. For finding all those speling errors.

Roman Yangarber. Pre-doc seminar opponent. For forcing me to think again. For devoting time to read, comment on, and discuss what's in here.

Magnus Sahlgren. Prodigious peer. For mental and physical sparring; all those mornings in the gym meant more to the making of this thesis than you'd imagine. As you so wisely put it: "contradas!". Thanks buddy!

Magnus Boman. For back-watching, text-commenting, and music-providing.

Åsa Rudström. For specific comments and general support.

Jussi Karlgren. Inspirationalist. Just do it. Thank you.

Kristofer Franzén. For squibs, discussions, and the sharing of tax-free goods, jet-lag and cultural clashes in front of many hotel room TV-sets.

Oscar Täckström, Stina Nylander, Marie Sjölinder, Markus Bylund, Preben Hansen, Gunnar Eriksson, and the rest of the Userware lab at SICS for being such a cosy bunch.

Janusz Launberg. Provider extraordinaire. Thank you.

Vicki Carleson. For promptly finding and delivering whatever paper or book I asked for, whenever I asked for it, no matter how obscure it seemed.

Mikael Nehlsen (then) at SICS, Robert Andersson at GSLT, and Rudolf Rydstedt and Leif-Jöran Olsson at the Department of Swedish at University of Gothenburg for accommodating the constant hogging of the computers for far too long a time.

Katrin Tomanek for eye-opening and thought-provoking discussions and the sharing of ideas.

Joakim Nivre, Satoshi Sekine, Ralph Grishman, and Koji Murakami for comments and discussions regarding early ideas that I intended to formalize and challenge on the form of a thesis, but didn't; what's in this dissertation is really the prequel to what we discussed back then. The not-yet-implemented idea of using cost sensitive active learning at the document level for parallel annotation of multiple tasks (Section 13.3.3) is perhaps the most obvious connection between this thesis and my original idea on which you all provided much appreciated feedback.

Christine Körner and Andreas Vlachos for kindly answering my questions.

The teachers and fellow students at the Swedish national Graduate School of Language Technology for creating and nurturing a creative environment. I feel very fortunate and I'm very happy to have been a part of GSLT.

Shout outs to Connexor for providing me with a student's license to their linguistic analyzer.

The work on this thesis has been fuelled by the patience of my family and funded in part by GSLT, SICS, the University of Gothenburg, and the European Commission via the projects COMPANIONS (IST-FP6-034434) co-ordinated by Yorick Wilks and DUMAS (IST-2000-29452) co-ordinated by Kristiina Jokinen and Björn Gambäck.

A very special and particularly intense holler goes to
my wife Tove, my kids Leonard and Viola,
my father Göran, mother Birgitta, and sister Kristin.

Fredrik Olsson, Stockholm, October 2008

I’ma ninja this shit wit’ sugar in the fuel tank of a saucer

– Ian Matthias Bavitz

CONTENTS

Appendices

1 INTRODUCTION

Information extraction is the process of analyzing unrestricted text with the purpose of excerpting information about pre-specified types of entities, the events in which the entities are engaged, and the relationships between entities and events. The state-of-the-art of information extraction methods is manifested in the construction of extraction systems that are accurate, robust and fast enough to be deployed outside the realms of the research laboratories where they are developed. Still, some important challenges remain to be dealt with before such systems may become widely used. One challenge is that of adapting information extraction systems to handle new tasks and operate on new domains. For instance, a system that works well in a particular setting, such as the extraction of management succession information from news wire texts, is unlikely to work at all when faced with the task of extracting interactions between proteins from biomedical texts.

The heart of the problem lies in the fact that, at present, full text understanding cannot be carried out by means of computers. In an attempt to circumvent this problem, we typically specify, in advance, what pieces and types of information are of interest. Thus, our efforts can be concentrated on constructing theories, methods and techniques for finding and processing what is believed to satisfy a prototypical need for information with respect to the domain at hand. The key to information extraction is the information need; a well-specified need allows us to focus on the parts of the information that satisfy the need, while the rest can be ignored. Herein lies a tension. On the one hand, a specific and unambiguously defined information need is a prerequisite for successful information extraction. On the other hand, this very specificity of the information need definition causes problems in adapting and constructing information extraction systems; any piece of information that falls outside a given definition of an information need will not be recognized by the system, simply because it does not look for such pieces.

Partly to accommodate the necessary specificity, information needs are often defined in terms of examples of what should be covered by the information

extraction system fulfilling the need. Thus, the creation of state-of-the-art information extraction systems has come to rely increasingly on methods for automatically learning from examples. Such training examples are often provided to a machine learner in the form of a body of texts, a corpus, that has been annotated so as to make explicit the parts and types of the corpus constituting the focus of the information extraction task at hand. The assumption in the research community seems to be that the annotation of data, which is later used for machine learning, is better than manually writing rules. Nevertheless, the question of why we should opt for the annotation of what is important in a text, instead of directly addressing that knowledge by means of explicitly written rules remains one which clearly deserves a moment of contemplation. Addressing the issues pertaining to the creation of information extraction systems at the level of data instead of at the system's level directly arguably has several pivotal advantages. Decoupling the characteristics of the training data and the extraction system induced from the data facilitates, for instance, future extensions of the data by adding further details concerning already known information, or the re-creation of information extraction systems based on a novel machine learning technique that was not known at the time the data was collected and annotated.

In an investigation concerning the marking-up of data versus the manual construction of a system, Ngai and Yarowsky (2000) contrast annotation with rule writing for the task of base noun phrase chunking; the recognition of non-recursive noun phrases in text. They air a voice in favor of annotating over rule writing. Their investigation compares an annotation process based on active machine learning (introduced in chapter 4) for selecting the sentences to be annotated, with the process of manually specifying rules. Ngai and Yarowksy find that base noun phrase chunkers learned from the annotated data outperforms the chunkers based on manually constructed rules, even when considering the human effort spent. They point out that annotating data has a number of advantages over writing rules:

- Annotation-based learning can continue over a long period of time; the decisions needed to be made by the annotator concern information appearing in a relatively local context. Writing rules, on the other hand, requires the human to be aware of all potential rule interdependencies. Over time, the latter task may take precedence and obscure an initially transparent view of the task through the rules.

- The efforts of several annotators are easier to combine, than are the efforts of several rule writers. Given that the annotators use the same annotation guidelines, their relative performance may be measured and

corrective actions can be taken accordingly. The local contexts of the annotation decisions allow for isolation of hard cases and their deferral to, for instance, external reviewers. Rule interdependencies on the other hand, may cause the combination of rule sets to result in a set exhibiting undesired side effects when applied.

- Constructing rules require more of the human involved in terms of linguistic knowledge, familiarity with the language in which to specify the rules, and an eye for rule dependencies.

- Creating annotations facilitates data re-use. An annotated corpus can be used by learning schemes other than the initially envisioned ones, and the performance of a system may thus be improved without altering the mark-up in the underlying data.

Ngai and Yarowsky (2000) also point out that based on their empirical observations, rule writing tends to result in systems exhibiting more variance than the corresponding systems created by training on annotated text.

Although the above discussion on annotation versus rule writing may portray the task of annotation as a rather simple one, it should be pointed out that this is not necessarily the case. Depending on the task, marking up linguistic content in text may be quite complex. The comprehensiveness of available annotation guidelines may serve as indicators of the complexity of the annotation task. For instance, a seemingly simple task such as the detection and recognition of entity mentions[1] in English text, as outlined in the context of Automatic Content Extraction (Linguistic Data Consortium 2008), is accompanied by a document spanning more than 70 pages devoted solely to the mark-up of five classes of entities; persons, organizations, geographical/social/political entities, locations, and facilities.

Another way of illustrating the difficulties with annotating linguistic phenomena is by looking at the agreement (or lack thereof) between human annotators operating on the same task and texts. The inter-annotator agreement for a given task is of particular interest since the agreement provides the upper bound on the performance expected by an annotation system induced from the marked-up data. That is, a system that is created by means of machine learning will, at best, perform as good as the examples from which the system was learned. Generally, the more complex structures to mark up, the lower the inter-annotator agreement scores.

[1]The annotation guidelines by the Linguistic Data Consortium (2008) define an entity to be "... an object or set of objects in the world". Named entity recognition is a core sub-task in information extraction, and as such it is further elaborated on in chapter 2.

Furthermore, while annotation may facilitate the re-use of data, it does not mean that data re-use is guaranteed to be successful. For instance, data that has been selected and annotated to fit the characteristics of a particular machine learning algorithm may not at all be useful in conjunction with a different learning algorithm (this matter is discussed in section 5.6). That said, the issue of difficulty in producing high quality annotated data has been raised. A major bottleneck in machine learning is the acquisition of data from which to learn; this is an impediment due to the requirement of large resources in terms of time and human expertise when domain experts are to mark up data as needed in the learning process. Thus, obtaining good training data is a challenge in its own right.

This thesis describes the development of a method – BootMark – for the acquisition of annotated data from which to learn named entity recognizers. Names constitute references to real-world entities that participate in events, and are engaged in relations to other entities. As such, names provide viable ways of obtaining handles to information that may fit a given extraction task. If the acquisition of marked-up texts could be made easier, in some respect, we would be one step closer towards making information extraction available to a broader public. It is within this context that the present thesis should be understood.

1.1 Thesis

I present a method for bootstrapping the annotation process of named entities in textual documents. The method, called BootMark, is focused on the creation of annotated data, as opposed to the creation of classifiers, and the application of the method thus primarily results in a corpus of marked up textual documents. BootMark requires a human annotator to manually mark-up fewer documents in order to produce a named entity recognizer with a given performance, than would be needed if the documents forming the base for the recognizer were randomly drawn from the same corpus.

1.2 Method and organization of the dissertation

Part I contains the background needed to understand the rest of the dissertation. Named entity recognition is introduced in chapter 2. The necessary concepts in machine learning are presented in chapter 3, followed by an extensive literature survey of active machine learning with a focus on applications in computational linguistics in chapter 4, and a survey of support for annotation processes in chapter 5.

Part II constitutes the core of the dissertation. Chapter 6 introduces and elaborates on a three-phase method called BootMark for bootstrapping the annotation of named entities in textual documents. In the process of describing BootMark, five issues emerge that need to be empirically tested in order to assess the plausibility of the BootMark method. These emerging issues are the subject matter of part III. Chapter 6 concludes part II by relating the BootMark method to existing work.

Part III provides an account of the empirical work conducted, all related to the set of emerging issues outlined in part II. Chapter 7 introduces an experimental setting in which the major concerns raised in part II pertaining to the plausibility of the proposed annotation method are empirically tested.

Chapter 8 describes the first set of experiments, related to the first emerging issues outlined in chapter 6. The goal of the experiments is to provide a baseline for experiments to come. This is accomplished by an investigation of the characteristics of a number of base learners with respect to their training and testing time, as well as their accuracy on the named entity recognition task. The experiments also include parameter selection, the use of automatic feature set reduction methods, and, for the best base learner also the generation of learning curves visualizing its ability to learn as more data becomes available.

Chapter 9 provides an extensive empirical investigation into the applicability of active machine learning for the purpose of selecting the document to annotate next based on those that have been previously marked-up. The investigation pertains to the most crucial of the emerging issues outlined in chapter 6.

Chapter 10 addresses the issue of the constitution of the document set utilized for starting the bootstrapping process.

Chapter 11 examines ways to monitor the active learning process, as well as to define a stopping criterion for it having available an annotated, held-out test set.

Chapter 12 concludes part III with a discussion concerning the use of the named entity recognizer learned during the bootstrapping phase of BootMark for marking up the remainder of the documents in the corpus.

Finally, part IV ends the dissertation with a summary, conclusions, and future work.

It should be noted that the experiments introduced and carried out in the following are considered as *indicative* of the plausibility of the BootMark method. Thus, the empirical investigations do not constitute attempts at *proving* the method correct. Whereas the experiments indeed are instantiations of particular issues crucial to the realization of the method as such, their outcomes should be considered fairly loosely tied to the method proper. For instance, the fact that a particular base learner is shown to yield the best named entity recognizer in the particular setting described in chapter 8, should not be taken as

evidence of the base learner being the most suitable one for other settings as well. Due to the purpose of the investigations, it makes little sense in accompanying the results and related discussions by statistical tests for judging the significance of the findings; instead, the indications provided by the results are made visible on the form of graphs and tables containing performance results and variations, as well as learning curves.

A trade-off between the amount of data used for the experiments and the number of experiments conducted is in effect. I chose to explore more experiment configurations, such as the number of base learners involved in chapter 8 and, in particular, the number of uncertainty and selection metrics utilized in chapter 9, rather than using more data. As an example, the 216 base learner configurations used in chapter 8 required the better part of six months and several different machines to run to completion. If the amount of data involved would have been increased, it would have had severe effects on the execution time.

1.3 Contributions

Apart from the dissertation as a whole, some particular contributions merit attention in their own right since they may prove useful to other involved in field of active learning involving named entity recognition. The contributions include:

- The definition and evaluation of a number of metrics for quantifying the uncertainty of a single learner with respect to the classification of a document (section 9.2).
- The definition and evaluation of a number of metrics for quantifying decision committee disagreement with respect to the classification of a document, including the definition of Weighted Vote Entropy (section 9.3).
- A way of combining the results from two view classifiers in Co-testing in such a way that the contribution of each view classifier is weighted according to its classification performance on the training data, thus maintaining the relative compatibility of the views (section 9.3.4).
- An intrinsic stopping criterion for committee-based active learning. The realization of the stopping criterion is based on the intrinsic characteristics of the data, and does not require the definition, nor setting of any thresholds (section 11.3).

- A strategy for deciding whether the predicted label for a given instance (a token in the context of a document) should be suggested as a label to the human annotator during pre-tagging with revision. Employing the described selective strategy may allow for the use of pre-tagging with revision during the bootstrapping phase, something which otherwise appears volatile (section 12.2).

Part I

Background

2 NAMED ENTITY RECOGNITION

Named entity recognition is the task of identifying and categorizing textual references to objects in the world, such as persons, organizations, companies, and locations. Figure 2.1 contains an example sentence, taken from a corpus used in the Seventh Message Understanding Conference (MUC-7).[2] The names in the example sentence in the figure are marked-up according to four of the seven name categories used in MUC-7: organization, location, date, and time.

Named entity recognition constitutes an enabling technique in many application areas, such as question-answering, summarization, and machine translation. However, it was within information extraction that named entity recognition was first thoroughly researched. Thus, to understand named entity recognition, it is described here in the context of a prototypical information extraction system.

Information extraction is the process of analyzing unrestricted text with the purpose of picking out information about pre-specified types of entities, the events in which the entities are engaged, and the relationships between entities and events. In this context, the purpose of named entity recognition is to identify and classify the entities with which the information extraction task is concerned. As such, named entity recognition is arguably a well-researched and well-understood field; a good overview is given by Nadeau and Sekine (2007).

Introductions to information extraction are provided by, for instance, Cowie and Lehnert (1996), Grishman (1997), and Appelt and Israel (1999), while Kaiser and Miksch (2005) give a more recent survey of the field. The core definition of information extraction evolved during the MUC series which took place in the 1990's (Grishman and Sundheim 1996; Chinchor 1998).

Figure 2.2 illustrates the organization of a typical information extraction system. Usually, an extraction system is made up from a cascade of different modules, each carrying out a well-defined task and working on the output of previous modules. At the top end of figure 2.2, text is fed to the system and

[2] The MUC-7 corpus is further described in section 7.1.

<ENAMEX TYPE="ORGANIZATION">Massport</ENAMEX> officials said the replacement <ENAMEX TYPE="ORGANIZATION">Martinair</ENAMEX> jet was en route from <ENAMEX TYPE="LOCATION">Europe</ENAMEX> to <ENAMEX TYPE="LOCATION">New Jersey</ENAMEX>, but was diverted to <ENAMEX TYPE="LOCATION">Logan</ENAMEX> <TIMEX TYPE="DATE">Tuesday</TIMEX> <TIMEX TYPE="TIME">afternoon</TIMEX>.

Figure 2.1: An example sentence in which named entities are marked.

passed through a *lexical analysis* phase which involves segmenting the document into, for example, sentences and tokens. The tokens are then analyzed in terms of part-of-speech and syntactic functions. Next, the *name recognition* module harvests the text for name expressions referring to, for instance persons, organisations, places, monetary expressions, and dates. The *partial syntax* step includes identifying nominal and verbal groups as well as noun phrases. The *scenario patterns* module applies domain and scenario specific patterns to the text in order to resolve higher level constructs such as preposition phrase attachment. *Reference resolution* and *discourse analysis*, then, relate co-referring expressions to each other, and try to merge event structures found so far. Finally, templates expressing the structured version of the answer to the information need are generated. As depicted in figure 2.2, named entity recognition constitutes an integral and crucial part of a typical information extraction system since many subsequent modules depend on the output of the named entity recognizer.

The term *named entity recognition* was originally introduced in MUC-6 in 1995 (Grishman and Sundheim 1996). The task subsequently evolved during a number of different venues, including MUC-7 and the Second Multilingual Entity Task (MET-2) (Chinchor 1998), the HUB-4 Broadcast News technology evaluation (Chinchor, Robinson and Brown 1998), the Information Retrieval and Extraction Exercise (IREX) (Sekine and Ishara 2000), two shared tasks conducted within the Conference on Computational Natural Language Learning (CoNLL) (Tjong Kim Sang 2002a; Tjong Kim Sang and Meulder 2003), and the Automatic Content Extraction (ACE) program (Doddington et al. 2004).

Throughout the MUC series, the term *named entity* came to include seven categories; persons, organizations, locations (usually referred to as ENAMEX), temporal expressions, dates (TIMEX), percentages, and monetary expressions (NUMEX). Over time, the taxonomies used for named entity recognition have been re-defined. The seven name categories used in the MUCs were extended to include the types *facility* and *geo-political entity* in the ACE program, while

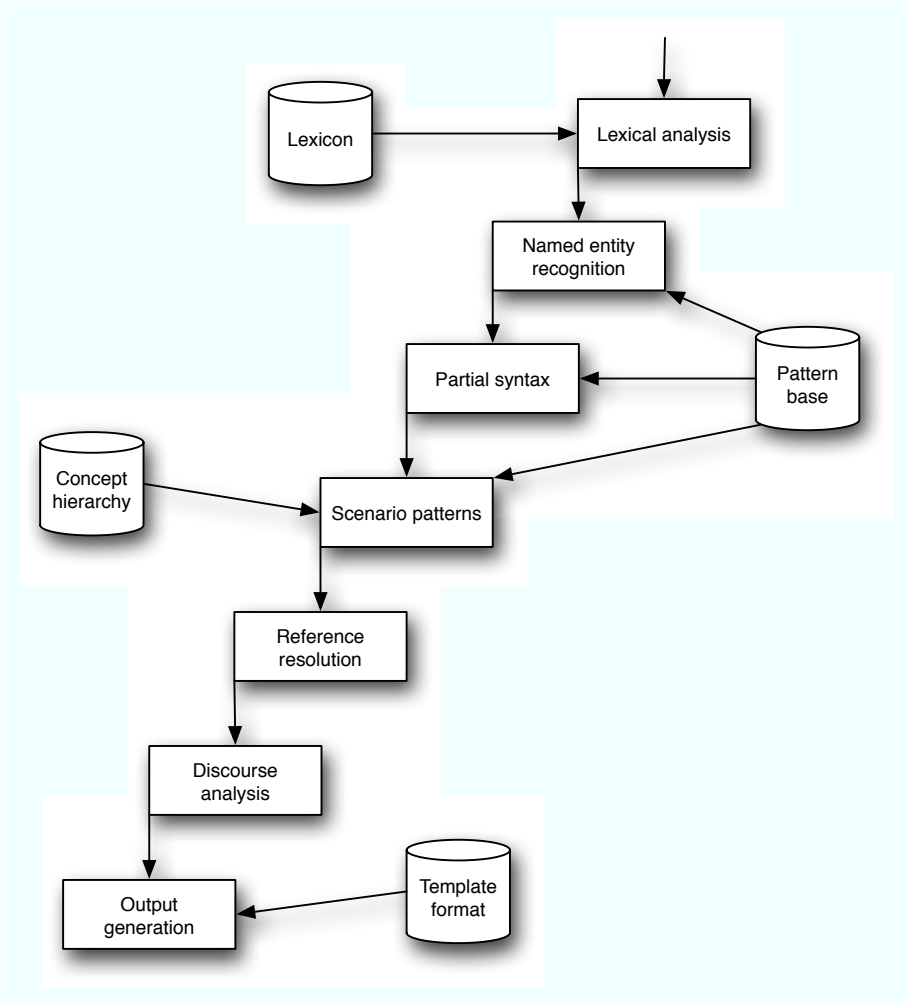

Figure 2.2: The organisation of a typical information extraction system, adopted from Yangarber and Grishman (1997).

types such as *protein* and *DNA* are part of the taxonomy used in the development of the GENIA corpus (Collier et al. 1999). More recently, Sekine and Nobata (2004) report ongoing work concerning what they refer to as *extended named entity recognition*, which comprises 200 categories of names.

Research on named entity recognition has been carried out for a number of languages other than English, for example, German, Spanish, and Dutch in the context of the CoNLL shared tasks (Tjong Kim Sang 2002a; Tjong Kim Sang and Meulder 2003), Japanese in IREX (Sekine and Ishara 2000), and

Swedish (Kokkinakis 2004; Borin, Kokkinakis and Olsson 2007). As Nadeau and Sekine (2007) point out, the domain and genre to which named entity recognition has been applied has not been varied to a great extent. The data sets used often consists of news wire texts, transcribed broadcast data, or scientific texts.

While the first systems for recognizing names were based on pattern matching rules and pre-compiled lists of information, the research community has since moved towards employing machine learning methods for creating such systems. The learning techniques applied include Decision Trees (Sekine 1998), Artificial Neural Networks (Carreras, Màrquez and Padró 2003), Hidden Markov Models (Bikel, Schwartz and Weischedel 1999), Maximum Entropy Models (Borthwick et al. 1998), Bayesian learning (Nobata, Collier and Tsujii 1999), Nearest Neighbor learning (Tjong Kim Sang 2002b), and Conditional Random Fields (McCallum and Li 2003).

A detailed description of a machine learning set-up used for named entity recognition is available in chapter 8, including the specification of the learning task, as well as the features used to represent training examples.

3 FUNDAMENTALS OF MACHINE LEARNING

This chapter introduces the concepts of machine learning methods used in the remainder of the thesis; as such, the chapter serves as a pointer to additional information, rather than a complete beginner's guide to the subject. Extensive introductions to machine learning are given by, for instance, Mitchell (1997) and Witten and Frank (2005). Mitchell (1997: 2) defines machine learning as:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

The definition naturally gives rise to additional questions. What is experience, and how can it be represented in a way beneficial to a computer program? How are the representations of experience and tasks related? What techniques are there to learn from experience? How can the performance of a learned computer program be measured? These questions are all addressed in the following.

3.1 Representation of task and experience

The way experience is represented is closely related to the way the task to be solved is expressed, both in terms of data structures used, and in terms of the granularity in which the experience – knowledge about the domain – is expressed. A common data structure to use for representation is a *vector* of *features* (often referred to as *attributes*). A feature denotes an aspect, an important piece of information, or a clue to how experience is best represented. Each feature can take on a *value*. As an example, the representation of an experience $e \in E$ from which a computer program is to learn can be written as:

$$e = (v_1, v_2, \ldots, v_{k-1}, v_k)$$

where v_1 to v_k are values of features $1, \ldots, k$ describing the experience e.

The difference between the representation of an experience from which to learn, and the task to be carried out is usually one attribute. In the example above, assume that the feature with value v_k is the feature that the computer program is to learn to predict. The range of possible values that v_k represents is called the *target class* of values. In the learning situation, the computer program is given the complete e as an example from which to learn. When the learned experience is to be applied, the example handed to the computer program is missing the value v_k such that

$$e = (v_1, v_2, \ldots, v_{k-1}, \_)$$

An experience from which to learn is often referred to as an *example* (training example), while the corresponding task of classifying (or predicting) the experience as belonging to a particular class (or being a particular value) is referred to as an*instance*. The terms *example* and *instance* are henceforth used interchangeably.

When the experience, or task, is such that the outcome can be categorized into discrete categories among which there is no relative order, the learning is said to pertain to *classification*. If, on the other hand, the outcome of the learning is predicting a numeric quantity, the learning is said to pertain to *regression*.

The *hypothesis space* is the space consisting of all possible feature values used for representing experience. The *version space* consists of all those combinations of feature values that are *consistent* with the training examples used for representing the experience. A hypothesis is said to be consistent with a set of training examples if the hypothesis predicts the same target class value as is represented by the training examples.

3.2 Ways of learning from experience

How can a program learn from experience? Traditionally, there are two strands to learning: *supervised* and *unsupervised* learning. In supervised learning, the experiences from which to learn are commonly presented as pairs consisting of an example and the correct class label (or value) associated with the example; this is the case in the above description of experience. In unsupervised learning on the other hand, the examples provided to the learner[3] are not associated with any class labels or values at all. Here, the task of the learner is to find interesting structures, or patterns, in the data. Between supervised and unsupervised learning is *semi-supervised* learning, in which the learner typically has access

[3]The terms *machine learner* and *learner* are henceforth used interchangeably.

to some labeled training examples and a lot of unlabeled examples. An introduction to semi-supervised learning is given in the book *Semi-Supervised Learning* by Chapelle, Schölkopf and Zien (2006).

Although this thesis is mainly concerned with supervised learning methods – the ones brought up in the present section are those subject to investigation in chapter 8 – one example of semi-supervised learning called Co-training is introduced in the context of active learning in chapter 4.

3.2.1 Decision tree learning

A decision tree is a directed acyclic graph in which the nodes constitute tests between features, the branches between a node and its children correspond to the values of the features, and leaves represent values of the target class.

The creation of a decision tree can be defined recursively. Initially, select the feature which best, on its own, predicts the correct classes of the training examples available. The first feature selected constitutes the root node of the tree. Branches corresponding to all possible values of the feature are created (one branch per value). In effect, the original set of training examples is now divided into sub-parts related to each value/branch. Continuing with the sub-parts, the process is repeated; for each sub-part, select the feature that best predicts the target classes of the training examples in the set. This process is repeated until all training examples corresponding to a node have the same target class. The decision tree learning approach is called *divide-and-conquer*.

When classifying an instance by means of a decision tree, the tree is traversed from the root, going towards the leaves, comparing the feature values in the instance with those available at the nodes in the tree until a leaf is reached. The instance is then assigned the class of the leaf at which the traversal of the tree ends.

Decision trees are robust with respect to noise in the input data, they are also relatively easy to interpret by a human, and can be used for classification as well as regression.

Two decision tree learners are used in the present thesis, J48 and REPTree, described in Witten and Frank 2005. The former is a re-implementation of the well-known C4.5 (Quinlan 1993).

3.2.2 Lazy learning

Lazy learning is also known as instance-based learning. The name lazy learning refers to the way that the learning is carried out. In the learning phase,

training examples are merely collected, while the bulk of the work is carried out during the application phase. The lazy learning method employed in this thesis is called *k-nearest neighbor*, or kNN for short. The idea is that an instance to classify is compared to its nearest neighbors – already collected examples – and the classification of the instance is made based on the classes of the neighbors. The *k* in kNN refers to the number of neighbors to consider when calculating the class or value of a given instance. kNN can be used for classification as well as regression.

The approach taken makes nearest neighbor fast in the learning phase, but slow to classify new data, as most of the computations required are made at that point. The benefits of using kNN include that when classifying a given instance, only the examples close to the instance have to be taken into consideration; the classification is based on local characteristics of the hypothesis space. This means that kNN can be used to model complex phenomena by using fairly simple, local, approximations. The kNN implementation used in this thesis is called IBk (Witten and Frank 2005).

3.2.3 Artificial neural networks

Artificial neural networks are non-linear statistical data modeling tools, used for classification or regression by modelling the relationships between input and output data. An artificial neural network can be described as a graph in which the nodes, the artificial neurons, are connected by arcs. A neural network as a whole models a function for mapping the network's input to its output. That function, in turn, is represented as the combination of sub-functions, each of which is manifested as the mapping between the input and output of a node in the network. The strength, or influence, of a sub-function is modeled as a weight on the arc leading to the node representing the function. Training an artificial neural network essentially involves first designing the network in accordance with the task and data at hand, and then deciding the weights of the arcs based on observations of the training examples.

There is a multitude of artificial neural networks available. The type of network used in this thesis is called *Radial Basis Function network*, RBF network for short (Powell 1987). An RBF network is a *feedforward network*, meaning that it is a directed acyclic graph. An RBF network typically consists of three layers of nodes; the input layer, a hidden layer, and the output layer. The two latter layers are referred to as processing layers. In the hidden processing layer, the input is mapped onto the radial basis functions representing the nodes. Each node in the hidden layer can be thought of as representing a point in the space made up by the training examples. The output from a hidden node can thus be

conceptualized as depending on the distance between the instance to be classified and the point in space represented by the node. The closer the two are, the stronger the output from the node (the more influence the node has on the final classification of the instance). The distance between an instance and a point represented by a node is measured by means of a nonlinear transformation function that converts the distance into a similarity measure. The hidden nodes are called RBFs since the points for which the strength of the output of the node is at the same level form a hypersphere or hyperellipsoid. In the case of regression, the network output is realized as a linear combination of the output of the nodes in the hidden layer. In the case of classification, the output is obtained by applying a *sigmoid function* to the output of the hidden nodes. The sigmoid function, also known as *logistic function* or *squashing function*, maps a potentially very large input domain to a small range of outputs.

RBF networks allow for efficient training, since the nodes in the hidden layer and the nodes in the output layers can be trained independently.

3.2.4 Rule learning

In rule learning, the goal is to learn sets of if-then rules that describe the training examples in a way that facilitates the decision making required to classify instances. For each target class in the training examples, rule sets are usually learned by finding a rule that *covers* the class in the sense that the rule classifies the examples correctly. Covering algorithms work by separating the training examples pertaining to one class from those of other classes, and continuously adding constraints – tests – to the rules under development in order to obtain rules with the highest possible accuracy for the given class. The approach is referred to as *separate-and-conquer* (in contrast to the divide-and-conquer approach taken in decision tree learning).

Two different rule learning algorithms are used in this thesis, JRip and PART, both of which are described by Witten and Frank (2005).

JRip is an implementation of RIPPER, short for *Repeated Incremental Pruning to Produce Error Reduction* (Cohen 1995). RIPPER is a separate-and-conquer algorithm that employs *incremental reduced-error pruning* to come to terms with potentially *overfitting* the learned set of rules to the training examples, as well as a global optimization strategy to increase the accuracy of the rule set. Overfitting means that the classifier learned is too specific to the training data at hand, and consequently does not generalize well to previously unseen data. In incremental reduced-error pruning, the rule learner divides the set of training examples into two sub-sets. The first set (the growing set) is used for learning rules, while the second set (the pruning set) is used for testing the

accuracy of the rules as the learning algorithm tries to remove tests from the rules, that is, prunes them. A pruned rule is preferred over an un-pruned rule if it performs better on the pruning set. *Incremental* reduced-error pruning means that each rule is pruned directly after being created, as opposed to deferring the pruning process until all rules have been created.

A global optimization step is used by RIPPER to increase the overall accuracy of the rule set by addressing the performance of individual rules. Once the complete rule set has been generated for a class, two variants of each rule are produced by using reduced-error pruning. This time, the error pruning phase is a bit different from the incremental one used to prune rules the first time around; the difference lies in that instances of the class that are covered by rules other than the one which is currently being considered for optimization are removed from the pruning set. The accuracy of the rule measured on the remaining instances in the pruning set is used as the pruning criterion. This procedure is repeated for each rule in the original rule set.

The other rule learning algorithm utilized in the thesis is called PART (Frank and Witten 1998). The way PART operates makes it possible to avoid the global optimization step used by RIPPER, and still obtain accurate rules. Essentially, PART combines the separate-and-conquer approach used in RIPPER with the divide-and-conquer approach used in decision tree learning. The former is realized as PART builds a rule, and subsequently removes the instances covered by the rule, thus separating the positive examples from the negative ones. The rule learning then proceeds recursively with the remaining instances. The divide-and-conquer approach is realized in that PART builds a pruned C4.5 decision tree for the set of instances currently in focus. The path leading to the leaf with best coverage is then used to formulate a rule, and the tree is discarded.

3.2.5 Naïve Bayesian learning

Naïve Bayesian learning is a special case of Bayesian learning, which in turn is a member of a family of statistical methods called graphical models.

Naïve Bayes constitutes a way of estimating the conditional probability distribution of the values of the target class, given the values of the features used for representing the experience. Naïve Bayes builds on applying *Bayes theorem* with strong (naïve) independence assumptions concerning the relations between the features used for representing experience. Bayes theorem provides a way of calculating the probability of a hypothesis concerning the classification of a given instance based on the *prior probability* that the hypothesis being correct, the probabilities of making various observations once

the hypothesis is believed to be true, and based on the observed data itself. The prior probability of a hypothesis reflects any background knowledge about the chance of the hypothesis being correct, for instance obtained from observations supporting the hypothesis in the training data. The independence assumption facilitates the calculation of the estimated probability of an instance belonging to a given class based on observations of each feature value in isolation in the training data, and relating that information to the sought for class using Bayes theorem.

In the learning phase, a naïve Bayesian learner calculates the frequencies of the feature values given each possible value of the target class. The frequencies are then normalized to sum to one, so as to obtain the corresponding estimated probabilities of the target classes.

In the classification phase, a naïve Bayesian classifier assigns the value of the target class that has the highest estimated probability, based on information regarding the feature values used for representing the instance obtained in the training phase.

In the experiments carried out in part III, two methods based on naïve Bayes are used; *Naïve Bayes* and *Naïve Bayes Updateable* (see, for instance, Witten and Frank 2005). The latter is able to accommodate learning by digesting new training examples as they are provided, in an incremental fashion, while the former method does not.

3.2.6 Logistic regression

Despite the fact that the name contains the term regression, previously introduced as pertaining to the prediction of numeric values, logistic regression can be used for classification. Logistic regression is a *linear classification* method suitable for domains in which the features used to describe experience take on numeric values. The most basic form of linear classification involves combining, by addition, the numeric features, with pre-determined weights indicating the importance of a particular feature to a given class.

Logistic regression makes use of a function for transforming the values of the target class into something that is suitable for numeric prediction. Since, in classification, the target class assumes discrete values, predicting the discrete values by means of regression necessitates a mapping from numeric intervals to the target class values. The function used for transforming the target class values – the transformation function – used in logistic regression is called the logistic function. The key to the logistic function is that it is able to map any numbers onto the interval ranging from 0 to 1 (the logistic function is a common sigmoid function, previously introduced in section 3.2.3 for RBF networks).

Training a logistic regression classifier, also known as a maximum entropy classifier, involves fitting the weights of each feature value for a particular class to the available training data. A good fit of the weights to the data is obtained by selecting weights to maximize the *log-likelihood* of the learned classification model. The log-likelihood is a way of expressing the values of weights based on the values of the target class.

Usually, logistic regression used in a multi-class setting consists of several classifiers each of which is trained to tell one class apart from another (pairwise classification). In classifying an instance, the instance is assigned the class that receives the most votes by the sub-classifiers.

The logistic regression approach used in part III of the present thesis is called *Logistic*, or *multinomial logistic regression* (le Cessie and van Houwelingen 1992).

3.3 Evaluating performance

Once a classifier has been learned, how is its performance to be evaluated? For a number of reasons, it is not common practice to evaluate a classifier on the same data that was used for training. Instead, the training and testing data should be kept separate in order to, for example, avoid overfitting the classifier. Among other things, overfitting may cause overly optimistic performance figures that most probably do not reflect the true behaviour of the classifier when it is facing previously unseen data. At the same time, it is desirable to use as much of the available data as possible in training; there is clearly a trade-off between the amount of data used for training, and the amount used for evaluating the learned classifier. One way to strike a balance is to divide the available data into n parts equal in size, train on parts $1, \ldots, (n-1)$, and evaluate the result on the remaining part. The procedure is then repeated for as many parts there are. This approach is called n-fold cross-validation. Usually n is set to 10, and the evaluation is then called 10-fold cross-validation.

The way to evaluate the coverage performance of a classifier depends on the task at hand. Throughout the thesis, four metrics are used: accuracy, precision, recall, and F-score. Precision, recall and F-score are commonly used in information retrieval and information extraction. The performance metrics can be defined in terms of the number of *true positives* (TP), *true negatives* (TN), *false positives* (FP), and *false negatives* (FN) returned by a classifier when classifying a set of instances. A true positive is an instance correctly classified as belonging to a given class. Conversely, a true negative is an instance correctly classified as not belonging to a given class. A false positive is an instance erroneously classified as belonging to a given class, while a false negative is an instance erroneously classified as not belonging to a class.

- The accuracy is simply the amount of correctly classified instances, usually given as a percentage:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

- Precision, P, is defined as the ratio between the number of correctly classified instances and the number of classified instances:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

- Recall, R, is defined as the ratio between the number of correctly classified instances and the total number of instances:

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

- The F-score is the harmonic mean of *precision* and *recall* such that

$$F = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R} \tag{4}$$

  where β is a constant used for determining the influence of precision over recall, or vice-versa. In the remainder of the thesis β is set to 1, which is commonly referred to as F_1 or $F_{\beta=1}$.

Precision, recall, and F-score assume values in the interval of 0 to 1, where higher values are better. Values are commonly reported in percentages, for instance, an F-score of 0.85 is often written as 85%.

4 ACTIVE MACHINE LEARNING

Active machine learning is a supervised learning method in which the learner is in control of the data from which it learns. That control is used by the learner to ask an oracle, a teacher, typically a human with extensive knowledge of the domain at hand, about the classes of the instances for which the model learned so far makes unreliable predictions. The active learning process takes as input a set of labeled examples, as well as a larger set of unlabeled examples, and produces a classifier and a relatively small set of newly labeled data. The overall goal is to produce as good a classifier as possible, without having to mark-up and supply the learner with more data than necessary. The learning process aims at keeping the human annotation effort to a minimum, only asking for advice where the training utility of the result of such a query is high.

On those occasions where it is necessary to distinguish between "ordinary" machine learning and active learning, the former is sometimes referred to as *passive learning* or learning by *random sampling* from the available set of labeled training data.

A prototypical active learning algorithm is outlined in figure 4.1. Active learning has been successfully applied to a number of language technology tasks, such as

- information extraction (Scheffer, Decomain and Wrobel 2001; Finn and Kushmerick 2003; Jones et al. 2003; Culotta et al. 2006);
- named entity recognition (Shen et al. 2004; Hachey, Alex and Becker 2005; Becker et al. 2005; Vlachos 2006; Kim et al. 2006);
- text categorization (Lewis and Gale 1994; Lewis 1995; Liere and Tadepalli 1997; McCallum and Nigam 1998; Nigam and Ghani 2000; Schohn and Cohn 2000; Tong and Koller 2002; Hoi, Jin and Lyu 2006);
- part-of-speech tagging (Dagan and Engelson 1995; Argamon-Engelson and Dagan 1999; Ringger et al. 2007);

- parsing (Thompson, Califf and Mooney 1999; Hwa 2000; Tang, Luo and Roukos 2002; Steedman et al. 2003; Hwa et al. 2003; Osborne and Baldridge 2004; Becker and Osborne 2005; Reichart and Rappoport 2007);

- word sense disambiguation (Chen et al. 2006; Chan and Ng 2007; Zhu and Hovy 2007; Zhu, Wang and Hovy 2008a);

- spoken language understanding (Tur, Hakkani-Tür and Schapire 2005; Wu et al. 2006);

- phone sequence recognition (Douglas 2003);

- automatic transliteration (Kuo, Li and Yang 2006); and

- sequence segmentation (Sassano 2002).

One of the first attempts to make expert knowledge an integral part of learning is that of query construction (Angluin 1988). Angluin introduces a range of queries that the learner is allowed to ask the teacher, such as queries regarding *membership* ("Is this concept an example of the target concept?"), *equivalence* ("Is X equivalent to Y?"), and *disjointness* ("Are X and Y disjoint?"). Besides a simple *yes* or *no*, the full answer from the teacher can contain counterexamples, except in the case of membership queries. The learner constructs queries by altering the attribute values of instances in such a way that the answer to the query is as informative as possible. Adopting this generative approach to active learning leads to problems in domains where changing the values of attributes are not guaranteed to make sense to the human expert; consider the example of text categorization using a bag-of-word approach. If the learner first replaces some of the words in the representation, and then asks the teacher whether the new artificially created document is a member of a certain class, it is not likely that the new document makes sense to the teacher.

In contrast to the theoretically interesting generative approach to active learning, current practices are based on example-driven means to incorporate the teacher into the learning process; the instances that the learner asks (queries) the teacher to classify all stem from existing, unlabeled data. The *selective sampling* method introduced by Cohn, Atlas and Ladner (1994) builds on the concept of membership queries, albeit from an example-driven perspective; the learner queries the teacher about the data at hand for which it is uncertain, that is, for which it believes misclassifications are possible.

1. Initialize the process by applying base learner B to labeled training data set D_L to obtain classifier C.
2. Apply C to unlabeled data set D_U to obtain $D_U{}'$.
3. From $D_U{}'$, select the most informative n instances to learn from, I.
4. Ask the teacher for classifications of the instances in I.
5. Move I, with supplied classifications, from $D_U{}'$ to D_L.
6. Re-train using B on D_L to obtain a new classifier, C'.
7. Repeat steps 2 through 6, until D_U is empty or until some stopping criterion is met.
8. Output a classifier that is trained on D_L.

Figure 4.1: A prototypical active learning algorithm.

4.1 Query by uncertainty

Building on the ideas introduced by Cohn and colleagues concerning selective sampling (Cohn, Atlas and Ladner 1994), in particular the way the learner selects what instances to ask the teacher about, *query by uncertainty* (*uncertainty sampling, uncertainty reduction*) queries the learning instances for which the current hypothesis is least confident. In query by uncertainty, a single classifier is learned from labeled data and subsequently utilized for examining the unlabeled data. Those instances in the unlabeled data set that the classifier is least certain about are subject to classification by a human annotator. The use of confidence scores pertains to the third step in figure 4.1. This straightforward method requires the base learner to provide a score indicating how confident it is in each prediction it performs.

Query by uncertainty has been realized using a range of base learners, such as logistic regression (Lewis and Gale 1994), Support Vector Machines (Schohn and Cohn 2000), and Markov Models (Scheffer, Decomain and Wrobel 2001). They all report results indicating that the amount of data that require annotation in order to reach a given performance, compared to passively learning from examples provided in a random order, is heavily reduced using query by uncertainty.

Becker and Osborne (2005) report on a two-stage model for actively learning statistical grammars. They use uncertainty sampling for selecting the sentences for which the parser provides the lowest confidence scores. The problem

1. Initialize the process by applying *EnsembleGenerationMethod* using base learner B on labeled training data set D_L to obtain a committee of classifiers C.
2. Have each classifier in C predict a label for every instance in the unlabeled data set D_U, obtaining labeled set $D_U{}'$.
3. From $D_U{}'$, select the most informative n instances to learn from, obtaining $D_U{}''$.
4. Ask the teacher for classifications of the instances I in $D_U{}''$.
5. Move I, with supplied classifications, from $D_U{}''$ to D_L.
6. Re-train using *EnsembleGenerationMethod* and base learner B on D_L to obtain a new committee, C.
7. Repeat steps 2 through 6 until D_U is empty or some stopping criterion is met.
8. Output a classifier learned using *EnsembleGenerationMethod* and base learner B on D_L.

Figure 4.2: A prototypical query by committee algorithm.

with this approach, they claim, is that the confidence score says nothing about the state of the statistical model itself; if the estimate of the parser's confidence in a certain parse tree is based on rarely occurring information in the underlying data, the confidence in the confidence score is low, and should thus be avoided. The first stage in Becker and Osborne's two-stage method aims at identifying and singling out those instances (sentences) for which the parser cannot provide reliable confidence measures. In the second stage, query by uncertainty is applied to the remaining set of instances. Becker and Osborne (2005) report that their method performs better than the original form of uncertainty sampling, and that it exhibits results competitive with a standard query by committee method.

4.2 Query by committee

Query by committee, like query by uncertainty, is a selective sampling method, the fundamental difference between the two being that query by committee is a multi-classifier approach. In the original conception of query by committee, several hypotheses are randomly sampled from the version space (Seung, Opper and Sompolinsky 1992). The committee thus obtained is used to examine the set of unlabeled data, and the disagreement between the hypotheses with re-

spect to the class of a given instance is utilized to decide whether that instance is to be classified by the human annotator. The idea with using a decision committee relies on the assumption that in order for approaches combining several classifiers to work, the ensemble needs to be made up from diverse classifiers. If all classifiers are identical, there will be no disagreement between them as to how a given instance should be classified, and the whole idea of voting (or averaging) is invalidated. Query by committee, in the original sense, is possible only with base learners for which it is feasible to access and sample from the version space; learners reported to work in such a setting include Winnow (Liere and Tadepalli 1997), and perceptrons (Freund et al. 1997). A prototypical query by committee algorithm is shown in figure 4.2.

4.2.1 Query by bagging and boosting

Abe and Mamitsuka (1998) introduce an alternative way of generating multiple hypotheses; they build on *bagging* and *boosting* to generate committees of classifiers from the same underlying data set.

Bagging, short for bootstrap aggregating (Breiman 1996), is a technique exploiting the bias-variance decomposition of classification errors (see, for instance, Domingos 2000 for an overview of the decomposition problem). Bagging aims at minimizing the variance part of the error by randomly sampling – with replacement – from the data set, thus creating several data sets from the original one. The same base learner is then applied to each data set in order to create a committee of classifiers. In the case of classification, an instance is assigned the label that the majority of the classifiers predicted (majority vote). In the case of regression, the value assigned to an instance is the average of the predictions made by the classifiers.

Like bagging, boosting (Freund and Schapire 1997) is a way of combining classifiers obtained from the same base learner. Instead of building classifiers independently, boosting allows for classifiers to influence each other during training. Boosting is based on the assumption that several classifiers learned using a weak[4] base learner, over a varying distribution of the target classes in the training data, can be combined into one strong classifier. The basic idea is to let classifiers concentrate on the cases in which previously built classifiers failed to correctly classify data. Furthermore, in classifying data, boosting assigns weights to the classifiers according to their performance; the better the performance, the higher valued is the classifier's contribution in voting (or averaging). Schapire (2003) provides an overview of boosting.

[4] A learner is *weak* if it produces a classifier that is only slightly better than random guessing, while a learner is said to be *strong* if it produces a classifier that achieves a low error with high confidence for a given concept (Schapire 1990).

Abe and Mamitsuka (1998) claim that query by committee, query by bagging, and query by boosting form a natural progression; in query by committee, the variance in performance among the hypotheses is due to the randomness exhibited by the base learner. In query by bagging, the variance is a result of the randomization introduced when sampling from the data set. Finally, the variance in query by boosting is a result of altering the sampling according to the weighting of the votes given by the hypotheses involved. A generalized variant of query by bagging is obtained if the *EnsembleGenerationMethod* in figure 4.2 is substituted with bagging. Essentially, query by bagging applies bagging in order to generate a set of hypotheses that is then used to decide whether it is worth querying the teacher for classification of a given unlabeled instance. Query by boosting proceeds similarly to query by bagging, with boosting applied to the labeled data set in order to generate a committee of classifiers instead of bagging, that is, boosting is used as *EnsembleGenerationMethod* in figure 4.2.

Abe and Mamitsuka (1998) report results from experiments using the decision tree learner C4.5 as base learner and eight data sets from the UCI Machine Learning Repository, the latest release of which is described in (Asuncion and Newman 2007). They find that query by bagging and query by boosting significantly outperformed a single C4.5 decision tree, as well as boosting using C4.5.

4.2.2 ActiveDecorate

Melville and Mooney (2004) introduce ActiveDecorate, an extension to the Decorate method (Melville and Mooney 2003) for constructing diverse committees by enhancing available data with artificially generated training examples. Decorate – short for *Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples* – is an iterative method generating one classifier at a time. In each iteration, artificial training data is generated in such a way that the labels of the data are maximally different from the predictions made by the current committee of classifiers. A strong base learner is then used to train a classifier on the union of the artificial data set and the available labeled set. If the resulting classifier increases the prediction error on the training set, it is rejected as a member of the committee, and added otherwise. In ActiveDecorate, the Decorate method is utilized for generating the committee of classifiers, which is then used to decide which instances from the unlabeled data set are up for annotation by the human oracle. In terms of the prototypical query by committee algorithm in figure 4.2, ActiveDecorate is used as *EnsembleGenerationMethod*.

Melville and Mooney (2004) carry out experiments on 15 data sets from the UCI repository (Asuncion and Newman 2007). They show that their algorithm outperforms query by bagging and query by boosting as introduced by Abe and Mamitsuka (1998) both in terms of accuracy reached, and in terms of the amount of data needed to reach top accuracy. Melville and Mooney conclude that the superiority of ActiveDecorate is due to the diversity of the generated ensembles.

4.3 Active learning with redundant views

Roughly speaking, utilizing redundant views is similar to the query by committee approach described above. The essential difference is that instead of randomly sampling the version space, or otherwise tamper with the existing training data with the purpose of extending it to obtain a committee, using redundant views involves splitting the feature set into several sub-sets or *views*, each of which is enough, to some extent, to describe the underlying problem.

Blum and Mitchell (1998) introduce a semi-supervised bootstrapping technique called *Co-training* in which two classifiers are trained on the same data, but utilizing different views of it. The example of views provided by Blum and Mitchell (1998) is from the task of categorizing texts on the web. One way of learning how to do that is by looking at the links to the target document from other documents on the web, another way is to consider the contents of the target document alone. These two ways correspond to two separate views of learning the same target concept.

As in active learning, Co-training starts off with a small set of labeled data, and a large set of unlabeled data. The classifiers are first trained on the labeled part, and subsequently used to tag an unlabeled set. The idea is then that during the learning process, the predictions made by the first classifier on the unlabeled data set, and for which it has the highest confidence, are added to the training set of the second classifier, and vice-versa. The classifiers are then retrained on the newly extended training set, and the bootstrapping process continues with the remainder of the unlabeled data.

A drawback with the Co-training method as it is originally described by Blum and Mitchell (1998) is that it requires the views of data to be conditionally independent and compatible given the class, that is, each view should be enough for producing a strong learner compatible with the target concept. In practice, however, finding such a split of features may be hard; the problem is further discussed in section 4.3.1.

Co-training *per se* is not within the active learning paradigm since it does not involve a teacher, but the work by Blum and Mitchell (1998) forms the ba-

1. Initialize the process by applying base learner B using each v in views V to labeled training set D_L to obtain a committee of classifiers C.
2. Have each classifier in C predict a label for every instance in the unlabeled data set D_U, obtaining labeled set $D_U{}'$.
3. From $D_U{}'$, select those instances for which the classifiers in C predicted different labels to obtain the *contention set*[5] $D_U{}''$.
4. Select instances I from $D_U{}''$ and ask the teacher for their labels.
5. Move instances I, with supplied classifications, from $D_U{}''$ to D_L.
6. Re-train by applying base learner B using each v in views V to D_L to obtain committe C'.
7. Repeat steps 2 through 6 until D_U is empty or some stopping criterion is met.
8. Output the final classifier learned by combining base learner B, views in V, and data D_L.

Figure 4.3: A prototypical multiple view active learning algorithm.

sis for other approaches. One such approach is that of *Corrected Co-training* (Pierce and Cardie 2001). Corrected Co-training is a way of remedying the degradation in performance that can occur when applying Co-training to large data sets. The concerns of Pierce and Cardie (2001) include that of scalability of the original Co-training method. Pierce and Cardie investigate the task of noun phrase chunking, and they find that when hundreds of thousands of examples instead of hundreds, are needed to learn a target concept, the successive degradation of the quality of the bootstrapped data set becomes an issue. When increasing the amount of unlabeled data, and thus also increasing the number of iterations during which Co-training will be in effect, the risk of errors introduced by the classifiers into each view increases. In Corrected Co-training a human annotator reviews and edits, as found appropriate, the data produced by both view classifiers in each iteration, prior to adding the data to the pool of labeled training data. This way, Pierce and Cardie point out, the quality of the labeled data is maintained with only a moderate effort needed on behalf of the human annotator. Figure 4.3 shows a prototypical algorithm for multi-view active learning. It is easy to see how Corrected Co-training fits into it; if, instead of having the classifiers select the instances on which they disagree (step 3 in figure 4.3), each classifier selects the instances for which it makes highly confident predictions, and have the teacher correct them in step 4, the algorithm in figure 4.3 would describe Corrected Co-training.

[5] The instance or set of instances for which the view classifiers disagree is called the *contention point*, and *contention set*, respectively.

Hwa et al. (2003) adopt a Corrected Co-training approach to statistical parsing. In pursuing their goal – to further decrease the amount of corrections of parse trees a human annotator has to perform – they introduce *single-sided corrected Co-training*. Single-sided Corrected Co-training is like Corrected Co-training, with the difference that the annotator only reviews the data, parse trees, produced by one of the view classifiers. Hwa et al. (2003) conclude that in terms of parsing performance, parsers trained using some form of sample selection technique are better off than parsers trained in a pure Co-training setting, given the cost of human annotation. Furthermore, Hwa and colleagues point out that even though parsing performance achieved using single-sided Corrected Co-training is not as good as that resulting from Corrected Co-training, some corrections are better than none.

In their work, Pierce and Cardie (2001) note that corrected Co-training does not help their noun phrase chunker to reach the expected performance. Their hypothesis as to why the performance gap occurs, is that Co-training does not lend itself to finding the most informative examples available in the unlabeled data set. Since each classifier selects the examples it is most confident in, the examples are likely to represent aspects of the task at hand already familiar to the classifiers, rather than representing potentially new and more informative ones. Thus, where Co-training promotes confidence in the selected examples over finding examples that would help incorporating new information about the task, active learning works the other way around. A method closely related to Co-training, but which is more exploratory by nature, is *Co-testing* (Muslea, Minton and Knoblock 2000, 2006). Co-testing is an iterative process that works under the same premises as active learning in general, that is, it has access to a small set of labeled data, as well as a large set of unlabeled data. Co-testing proceeds by first learning a hypothesis using each view of the data, then asking a human annotator to label the unlabeled instances for which the view classifiers' predictions disagree on labels. Such instances are called the *contention set* or *contention point*. The newly annotated instances are then added to the set of labeled training data.

Muslea, Minton and Knoblock (2006) introduce a number of variants of Co-testing. The variations are due to choices of how to select the instances to query the human annotator about, as well as how the final hypothesis is to be created. The former choice pertains to step 4 in figure 4.3, and the options are:

Naïve – Randomly choose an example from the contention set. This strategy is suitable when using a base learner that does not provide confidence estimates for the predictions it makes.

Aggressive – Choose to query the example in the contention set for which the least confident classifier makes the most confident prediction. This strategy is suitable for situations where there is (almost) no noise.

Conservative – Choose to query the example in the contention set for which the classifiers makes predictions that are as close as possible. This strategy is suitable for noisy domains.

Muslea, Minton and Knoblock (2006) also present three ways of forming the final hypothesis in Co-testing, that is, the classifier to output at the end of the process. These ways concern step 8 in figure 4.3:

Weighted vote – Combine the votes of all view classifiers, weighted according to each classifier's confidence estimate of its own prediction.

Majority vote – Combine the votes of all view classifiers so that the label predicted by the majority of the classifiers is used.

Winner-takes-all – The final classifier is the one learned in the view that made the least amount of mistakes throughout the learning process.

Previously described multi-view approaches to learning all relied on the views being *strong*. Analogously to the notion of a strong learner in ensemble-based methods, a strong view is a view which provides enough information about the data for a learner to learn a given target concept. Conversely, there are *weak* views, that is, views that are not by themselves enough to learn a given target concept, but rather a concept more general or more specific than the concept of interest. In the light of weak views, Muslea, Minton and Knoblock (2006) redefine the notion of contention point, or contention set, to be the set of examples, from the unlabeled data, for which the strong view classifiers disagree. Muslea and colleagues introduce two ways of making use of weak views in Co-testing. The first is as tie-breakers when two strong views predict a different label for an unlabeled instance, and the second is by using a weak view in conjunction with two strong views in such a way that the weak view would indicate a mistake made by both strong views. The latter is done by detecting the set of contention points for which the weak view disagrees with both strong views. Then the next example to ask the human annotator to label, is the one for which the weak view makes the most confident prediction. This example is likely to represent a mistake made by both strong views, Muslea, Minton and Knoblock (2006) claim, and leads to faster convergence of the classifiers learned.

The experimental set-up in used by Muslea, Minton and Knoblock (2006) is targeted at testing whether Co-testing converges faster than the corresponding

single-view active learning methods when applied to problems in which there exist several views. The tasks are of two types: classification, including text classification, advertisement removal, and discourse tree parsing; and wrapper induction. For all tasks in their empirical validation, Muslea, Minton and Knoblock (2006) show that the Co-testing variants employed outperform the single-view, state-of-the art approaches to active learning that were also part of the investigation.

The advantages of using Co-testing include its ability to use any base learner suitable for the particular problem at hand. This seems to be a rather unique feature among the active learning methods reviewed in this chapter. Nevertheless, there are a couple of concerns regarding the shortcomings of Co-testing aired by Muslea and colleagues that need to be mentioned. Both concerns relate to the use of multiple views. The first is that Co-testing can obviously only be applied to tasks where there exist two views. The other of their concerns is that the views of data have to be uncorrelated (independent) and compatible, that is, the same assumption brought up by Blum and Mitchell (1998) in their original work on Co-training. If the views are correlated, the classifier learned in each view may turn out so similar that no contention set is generated when both view classifiers are run on the unlabeled data. In this case, there is no way of selecting an example for which to query the human annotator. If the views are incompatible, the view classifiers will learn two different tasks and the process will not converge.

Just as with committee-based methods, utilizing multiple views seems like a viable way to make the most of a situation that is caused by having access to a small amount of labeled data. Though, the question remains of how one should proceed in order to define multiple views in a way so that the they are uncorrelated and compatible with the target concept.

4.3.1 How to split a feature set

Acquiring a feature set split adhering to the assumptions underlying the multi-view learning paradigm is a non-trivial task requiring knowledge about the learning situation, the data, and the domain. Two approaches to the view detection and validation problem form the extreme ends of a scale; randomly splitting a given feature set and hope for the best at one end, and adopting a very cautions view on the matter by computing the correlation and compatibility for every combination of the features in a given set at the other end.

Nigam and Ghani (2000) report on randomly splitting the feature set for tasks where there exists no natural division of the features into separate views. The task is text categorization, using Naïve Bayes as base learner. Nigam and

Ghani argue that, if the features are sufficiently redundant, and one can identify a reasonable division of the feature set, the application of Co-training using such a non-natural feature set split should exhibit the same advantages as applying Co-training to a task in which there exists natural views.

Concerning the ability to learn a desired target concept in each view, Collins and Singer (1999) introduce a Co-training algorithm that utilizes a boosting-like step to optimize the compatibility between the views. The algorithm, called CoBoost, favors hypotheses that predict the same label for most of the unlabeled examples.

Muslea, Minton and Knoblock (2002a) suggest a method for validating the compatibility of views, that is, given two views, the method should provide an answer to whether each view is enough to learn the target concept. The way Muslea and colleagues go about is by collecting information about a number of tasks solved using the same views as the ones under investigation. Given this information, a classifier for discriminating between the tasks in which the views were compatible, and the tasks in which they were not, is trained and applied. The obvious drawback of this approach is that the first time the question of whether a set of views is compatible with a desired concept, the method by Muslea, Minton and Knoblock (2002a) is not applicable. In all fairness, it should be noted that the authors clearly state the proposed view validation method to be but one step towards automatic view detection.

Muslea, Minton and Knoblock (2002b) investigate view dependence and compatibility for several semi-supervised algorithms along with one algorithm combining semi-supervised and active learning (Co-testing), CoEMT. The conclusions made by Muslea and colleagues are interesting, albeit perhaps not surprising. For instance, the performance of all multi-view algorithms under investigation degrades as the views used become less compatible, that is, when the target concept learned by view classifiers are not the same in each view. A second, very important point made in (Muslea, Minton and Knoblock 2002a) is that the robustness of the active learning algorithm with respect to view correlation is suggested to be due to the usage of an active learning component; being able to ask a teacher for advice seems to compensate for the views not being entirely uncorrelated.

Balcan, Blum and Yang (2005) argue that, for the kind of Co-training presented by Blum and Mitchell (1998), the original assumption of conditional independence between views is overly strong. Balcan and colleagues claim that the views do not have to denote conditionally independent ways of representing the task to be useful to Co-training, if the base learner is able to correctly learn the target concept using positive training examples only.

Zhang et al. (2005) present an algorithm called *Correlation and Compatibility based Feature Partitioner, CCFP* for computing, from a given set of

features, independent and compatible views. CCFP makes use of feature pairwise symmetric uncertainty and feature-wise information gain to detect the views. Zhang and colleagues point out that in order to employ CCFP, a fairly large number of labeled examples are needed. Exactly how large a number is required is undisclosed. CCFP is empirically tested and Zhang et al. (2005) report on somewhat satisfactory results.

Finally, one way of circumventing the assumptions of view independence and compatibility is simply not to employ different views at all. Goldman and Zhou (2000) propose a variant of Co-training which assumes no redundant views of the data; instead, a single view is used by differently biased base learners. Chawla and Karakoulas (2005) make empirical studies on this version of Co-training. Since the methods of interest to the present thesis are those containing elements of active learning, which the original Co-training approach does not, the single-view multiple-learner approach to Co-training will not be further elaborated.

In the literature, there is to my knowledge no report on automatic means to discover, from a given set of features, views that satisfy the original Co-training assumptions concerning independence and compatibility. Although the Co-training method as such is not of primary interest to this thesis, offsprings of the method are. The main approach to active multi-view learning, Co-testing and its variants rely on the same assumptions as does Co-training. Muslea, Minton and Knoblock (2002b) show that violating the compatibility assumption in the context of an active learning component, does not necessarily lead to failure; the active learner might have a stabilizing effect on the divergence of the target concept learned in each view. As regards the conditional independence assumption made by Blum and Mitchell (1998), subsequent work (Balcan, Blum and Yang 2005) shows that the independence assumption is too strong, and that iterative Co-training, and thus also Co-testing, works under a less rigid assumption concerning the expansion of the data in the learning process.

4.4 Quantifying disagreement

So far, the issue of disagreement has been mentioned but deliberately not elaborated on. The algorithms for query by committee and its variants (figure 4.2) as well as those utilizing multiple views of data (figure 4.3) all contain steps in which the disagreement between classifiers concerning instances has to be quantified. In a two-class case, such quantification is simply the difference between the positive and negative votes given by the classifiers. Typically, instances for which the distribution of votes is homogeneous is selected for

querying. Generalizing disagreement to a multi-class case is not trivial. Körner and Wrobel (2006) empirically test four approaches to measuring disagreement between members of a committee of classifiers in a multi-class setting. The active learning approaches they consider are query by bagging, query by boosting, ActiveDecorate, and Co-testing. The disagreement measures investigated are *margin-based disagreement*, *uncertainty sampling-based disagreement*, *entropy-based disagreement*, and finally a measure of their own dubbed *specific disagreement*. Körner and Wrobel (2006) strongly advocate the use of margin-based disagreement as a standard approach to quantifying disagreement in an ensemble-based setting.

Sections 4.4.1 through 4.4.4 deal with the different measures used by Körner and Wrobel (2006), followed by the treatment of the commonly used *Kullback-Leibler divergence*, *Jensen-Shannon divergence*, *vote entropy*, and *F-complement* in sections 4.4.5 to 4.4.8.

4.4.1 Margin-based disagreement

Margin, as introduced by Abe and Mamitsuka (1998) for binary classification in query by boosting, is defined as the difference between the number of votes given to the two labels. Abe and Mamitsuka base their notion of margins on the finding that a classifier exhibiting a large margin when trained on labeled data, performs better on unseen data than does a classifier that has a smaller margin on the training data (Schapire et al. 1998). Melville and Mooney (2004) extend Abe and Mamitsuka's definition of margin to include class probabilities given by the individual committee members. Körner and Wrobel (2006), in turn, generalize Melville and Mooney's definition of margin to account for the multi-class setting as well. The margin-based disagreement for a given instance is the difference between the first and second highest probabilities with which an ensemble of classifiers assigns different class labels to the instance.

For example, if an instance X is classified by committee member 1 as belonging to class A with a probability of 0.7, by member 2 as belonging class B with a probability of 0.2, and by member 3 to class C with 0.3, then the margin for X is $A - C = 0.4$. If instance Y is classified by member 1 as class A with a probability of 0.8, by member 2 as class B with a probability of 0.9, and by member 3 as class C with 0.6, then the margin for Y is $B - A = 0.1$. A low value on the margin indicates that the ensemble disagree regarding the classification of the instance, while a high value signals agreement. Thus, in the above example, instance Y is more informative than instance X.

4.4.2 Uncertainty sampling-based disagreement

Originally, uncertainty sampling is a method used in conjunction with single classifiers, rather than ensembles of classifiers (see section 4.1). Körner and Wrobel (2006), though, prefer to view it as another way of generalizing the binary margin approach introduced in the previous section. In uncertainty sampling, instances are preferred that receives the lowest *class probability* estimate by the ensemble of classifiers. The class probability is the highest probability with which an instance is assigned a class label.

4.4.3 Entropy-based disagreement

The entropy-based disagreement used in (Körner and Wrobel 2006) is what they refer to as the ordinary entropy measure (*information entropy* or *Shannon entropy*) first introduced by Shannon (1948). The entropy H of a random variable X is defined in equation 5 in the case of a c class problem, that is, where X can take on values $x_1, \ldots, x_c$.

$$H(X) = -\sum_{i=1}^{c} p(x_i)\log_2 p(x_i) \tag{5}$$

where $p(x_i)$ denotes the probability of x_i. A lower value on $H(X)$ indicates less confusion or less uncertainty concerning the outcome of the value of X.

4.4.4 The Körner-Wrobel disagreement measure

The *specific disagreement* measure, here referred to as the *Körner-Wrobel disagreement measure* is a combination of margin-based disagreement M and the maximal class probability P over classes C in order to indicate disagreement on a narrow subset of class values. The Körner-Wrobel disagreement measure, R, is defined in equation 6.

$$R = M + 0.5\frac{1}{(|C|P)^3} \tag{6}$$

Körner and Wrobel (2006) find that the success of the specific disagreement measure is closely related to which active learning method is used. Throughout the experiments conducted by Körner and Wrobel, those configurations utilizing specific disagreement as selection metric perform less well than the margin-based and entropy-based disagreement measures investigated.

4.4.5 Kullback-Leibler divergence

The Kullback-Leibler divergence (*KL-divergence, information divergence*) is a non-negative measure of the divergence between two probability distributions p and q in the same event space $X = \{x_1, \ldots, x_c\}$. The KL-divergence, denoted $D(\cdot \parallel \cdot)$, between two probability distributions p and q is defined in equation 7.

$$D(p \parallel q) = \sum_{i=1}^{c} p(x_i) \log \frac{p(x_i)}{q(x_i)} \tag{7}$$

A high value on the KL-divergence indicates a large difference between the distributions p and q. A zero-valued KL-divergence signals full agreement, that is p and q are equivalent.

Kullback-Leibler divergence to the mean (Pereira, Tishby and Lee 1993) quantifies the disagreement between committee members; it is the average KL-divergence between each distribution and the mean of all distributions. KL-divergence to the mean, D_{mean} for an instance x is defined in equation 8.

$$D_{mean}(x) = \frac{1}{k} \sum_{i=1}^{k} D(p_i(x) \parallel p_{mean}(x)) \tag{8}$$

where k is the number of classifiers involved, $p_i(x)$ is the probability distribution for x given by the i-th classifier, $p_{mean}(x)$ is the mean probability distribution of all k classifiers for x, and $D(\cdot \parallel \cdot)$ is the KL-divergence as defined in equation 7.

KL-divergence, as well as KL-divergence to the mean, has been used for detecting and measuring disagreement in active learning, see for instance (McCallum and Nigam 1998; Becker et al. 2005; Becker and Osborne 2005)

4.4.6 Jensen-Shannon divergence

The *Jensen-Shannon divergence*, (*JSD*) is a symmetrized and smoothed version of KL-divergence, which essentially means that it can be used to measure the distance between two probability distributions (Lin 1991). The Jensen-Shannon divergence for two distributions p and q is defined in equation 9.

$$JSD(p,q) = H(w_1 p + w_2 q) - w_1 H(p) - w_2 H(q) \tag{9}$$

where w_1 and w_2 are the weights of the probability distributions such that $w_1, w_2 \geq 0$ and $w_1 + w_2 = 1$, and H is the Shannon entropy as defined in equation 5.

Lin (1991) defines the Jensen-Shannon divergence for k distributions as in equation 10.

$$JSD(p_1,\ldots,p_k) = H(\sum_{i=1}^{k} w_i p_i) - \sum_{i=1}^{k} w_i H(p_i) \tag{10}$$

where p_i is the class probability distribution given by the i-th classifier for a given instance, w_i is the vote weight of the i-th classifier among the k classifiers in the set, and $H(p)$ is the entropy as defined in equation 5. A Jensen-Shannon divergence value of zero signals complete agreement among the classifiers in the committee, while correspondingly, increasingly larger JSD values indicate larger disagreement.

4.4.7 Vote entropy

Engelson and Dagan (1996) use *vote entropy* for quantifying the disagreement within a committee of classifiers used for active learning in a part-of-speech tagging task. Disagreement VE for an instance e based on vote entropy is defined as in equation 11.

$$VE(e) = -\frac{1}{\log k} \sum_{i=0}^{|l|} \frac{V(l_i,e)}{k} \log \frac{V(l_i,e)}{k} \tag{11}$$

where k is the number of members in the committee, and $V(l_i,e)$ is the number of members assigning label l_i to instance e. Vote entropy is computed per tagged unit, for instance per token. In tasks where the smallest tagged unit is but a part of the construction under consideration, for instance in phrase chunking where each phrase may contain one or more tokens, the vote entropy of the larger unit is computed as the mean of the vote entropy of its parts (Ngai and Yarowsky 2000; Tomanek, Wermter and Hahn 2007a).

4.4.8 F-complement

Ngai and Yarowsky (2000) compare the vote entropy measure, as introduced by Engelson and Dagan, with their own measure called *F-complement* (*F-score complement*). Disagreement FC concerning the classification of data e among a committee based on the F-complement is defined as in equation 12.

$$FC(e) = \frac{1}{2} \sum_{k_i,k_j \in K} (1 - F_{\beta=1}(k_i(e),k_j(e))) \tag{12}$$

where K is the committee of classifiers, k_i and k_j are members of K, and $F_{\beta=1}(k_i(e), k_j(e))$ is the F-score, $F_{\beta=1}$ (defined in equation 4), of the classifier k_i's labelling of the data e relative to the evaluation of k_j on e.

In calculating the F-complement, the output of one of the classifiers in the committee is used as the answer key, against which all other committee members' results are compared and measured (in terms of F-score).

Ngai and Yarowsky (2000) find that the task they are interested in, base noun phrase chunking, using F-complement to select instances to annotate performs slightly better than using vote entropy. Hachey, Alex and Becker (2005) use F-complement to select sentences for named entity annotation; they point out that the F-complement is equivalent to the inter-annotator agreement between $|K|$ classifiers.

4.5 Selecting the seed set

The initial set of labeled data used in active learning should be representative with respect to the classes that the learning process is to handle. Omitting a class from the initial seed set might result in trouble further down the road when the learner fits the classes it knows of with the unlabeled data it sees. Instances that would have been informative to the learner can go unnoticed simply because the learner, when selecting informative instances, treat instances from several classes as if they belong to one and the same class.

A related issue is that of instance distribution. Given that the learner is fed a seed set of data in which all classes are represented, the number of examples of each class plays a crucial role in whether the learner is able to properly learn how to distinguish between the classes. Should the distribution of instances in the seed set mirror the (expected) distribution of instances in the unlabeled set?

In the context of text categorization, McCallum and Nigam (1998) report on a method that allows for starting the active learning process without any labeled examples at all. They select instances (documents) from the region of the pool of unlabeled data that has the highest density. A dense region is one in which the distance (based on Kullback-Leibler divergence, defined in equation 7) between documents is small. McCallum and Nigam (1998) combine expectation-maximization (Dempster, Laird and Rubin 1977) and active learning in a pool-based setting (section 4.6); their results show that the learning in this particular setting might in fact benefit from being initiated without the use of a labeled seed set of documents.

Tomanek, Wermter and Hahn (2007b) describe a three-step approach to compiling a seed set for the task of named entity recognition in the biomedical domain. In the first step, a list of as many named entities as possible is gath-

ered, the source being either a human domain expert, or some other trusted source. The second step involves matching the listed named entities against the sentences in the unlabeled document pool. Third, the sentences are ranked according to the number of diverse matches of named entities to include in the seed set. Tomanek, Wermter and Hahn (2007b) report results from running the same active learning experiment with three different seed sets; a randomly selected set, a set tuned according to the above mentioned method, and no seed set at all. Though the learning curves seem to converge, initially the tuned seed set clearly contributes to a better progression of learning.

In experimental settings, a work-around to the seed set selection problem is to run the active learning process several times, and then present the average of the results achieved in each round. Averaging rounds, combined with randomly selecting a fairly large initial seed set – where its size is possibly related to the number of classes – might prove enough to circumvent the seed set problem when conducting controlled experiments. How the issue is best addressed in a live setting is not clear.

4.6 Stream-based and pool-based data access

There are two ways in which a learner is provided access to data, either from a stream, or by selecting from a pool. In stream-based selection used by, among others, Liere and Tadepalli (1997) and McCallum and Nigam (1998), unlabeled instances are presented one by one. For each instance, the learner has to decide whether the instance is so informative that is should be annotated by the teacher. In the pool-based case – used by for example Lewis and Gale (1994), and McCallum and Nigam (1998) – the learner has access to a set of instances and has the opportunity to compare and select instances regardless of their individual order.

4.7 Processing singletons and batches

The issue of whether the learner should process a single instance or a batch of instances in each iteration has impact on the speed of the active learning process. Since in each iteration, the base learner generates classifiers based on the labeled training data available, adding only one instance at a time slows the overall learning process down. If, on the other hand, a batch of instances is added, the amount of data added to the training set in each iteration increases, and the learning process progresses faster. The prototypical active learning algorithms presented previously, see figures 4.1, 4.2 and 4.3, respectively, do not

advocate one approach over the other. In practice though, it is clearly easier to fit singleton instance processing with the algorithms. Selecting a good batch of instances is non-trivial since each instance in the batch needs to be informative, both with respect to the other instances in the batch, as well as with respect to the set of unlabeled data as a whole.

While investigating active learning for named entity recognition, Shen et al. (2004) use the notions of *informativeness*, *representativeness*, and *diversity*, and propose scoring functions for incorporating these measures when selecting batches of examples from the pool of unlabeled data. Informativeness relates to the uncertainty of an instance, representativeness relates an instance to the majority of instances, while diversity is a means to avoid repetition among instances, and thus maximize the training utility of a batch.

The pool-based approach to text classification adopted by McCallum and Nigam (1998) facilitates the use of what they refer to as *density-weighted pool-based sampling*. The density in a region around a given document – to be understood as representativeness in the vocabulary of Shen et al. (2004) – is quantified as the average distance between that document and all other documents. McCallum and Nigam (1998) combine density with disagreement, calculated as the Kullback-Leibler divergence (equation 7), such that the document with the largest product of density and Kullback-Leibler divergence is selected as a representative of many other documents, while retaining a confident committee disagreement. McCallum and Nigam show that density-weighted pool-based sampling used in conjunction with Kullback-Leibler divergence yields significantly better results than the same experiments conducted with pool-based Kullback-Leibler divergence, stream-based Kullback-Leibler divergence, stream-based vote entropy, and random sampling.

Tang, Luo and Roukos (2002) also experiment with representativeness, or density, albeit in a different setting; that of statistical parsing. They propose to use clustering of the unlabeled data set based on the distance between sentences, the resulting clusters are then used to compute the density of examples. Tang and colleagues define the distance between two sentences based on the parse trees corresponding to the sentences. A parse tree can be uniquely represented by a series of events, each of which is constituted by a parse action and its context. Sentence similarity is calculated as the Hamming edit distance between two sequences of events. The Hamming distance measures the number of substitutions (or errors) required to turn one sequence into the other (Hamming 1950). The results reported by Tang, Luo and Roukos (2002) show that taking density into account helps in keeping the amount of training data needed down, compared to random sampling.

Brinker (2003) addresses the issue of incorporating a diversity measure when selecting batches of instances. Brinker's work is carried out with Support

Vector Machines, and his batch selection method is accordingly described in terms of feature vectors in a high-dimensional space. When selecting single instances for querying, an instance with a minimal distance to the classification hyperplane is usually favored, since choosing such an instance will result in halving the version space. When selecting several unlabeled instances, Brinker (2003) argue that picking instances such that their angles are maximal with respect to each other, rather than relative to the decision hyperplane, is a batch selection technique which is both computationally cheap and scalable to large data sets. Brinker (2003) conducts empirical investigations using a number of UCI data sets (Asuncion and Newman 2007), and reports results indicating that previously approaches to active learning with Support Vector Machines are outperformed by his batch selection strategy.

Hoi, Jin and Lyu (2006) present work on large-scale text categorization in which a batch of documents is selected in each learning iteration. Hoi and colleagues report on the development of an active learning algorithm utilizing logistic regression as base learner, capable of selecting several documents at a time, while minimizing the redundancy in the selected batch. The uncertainty of the logistic regression model is measured using Fisher matrix information, something which is claimed to allow for the batch selection problem to be recast as an optimization problem in which instances from the unlabeled pool are selected in such as way that the Fisher information is maximized. The notion of Fisher information and Fisher matrix is described by Hoi, Jin and Lyu (2006). Hoi and colleagues carry out experiments on several document collections, using a range of learning methods, and conclude that their active learning approach equipped with the batch selection method is more effective than the margin-based active learning methods tested.

4.8 Knowing when to stop

A number of different approaches for knowing when to stop the active learning process have been suggested in the literature. These approaches include to decide on a target accuracy and stop when it has been reached, to go on for a given number of active learning iterations, or to exhaust the pool of unlabeled data.

Some more elaborate methods monitor the accuracy as the learning process progresses and stop when accuracy deterioration is detected. Schohn and Cohn (2000) observe, while working with Support Vector Machines for document classification, that when instances are drawn at random from the pool of unlabeled data, the classifier performance increases monotonically. However, when Schohn and Cohn add instances according to their active learning selec-

tion metric, classifier performance peaks at a level above that achieved when using all available data. Thus, they obtain better performance by training on a subset of data, than when using all data available. Schohn and Cohn (2000) use this observation to form the basis for a stopping criterion; if the best, most informative instance is no closer to the decision hyperplane than any of the support vectors, the margin has been exhausted and learning is terminated. This is an approximation of true peak performance that seem to work well, Schohn and Cohn (2000) claim.

Zhu and Hovy (2007) investigate two strategies for deciding when to stop learning – *max-confidence* and *min-error* – in a word sense disambiguation task. Max-confidence relies on an entropy-based uncertainty measure of unlabeled instances, while min-error is based on the classification accuracy of predicted labels for instances when compared to the labels provided by the human annotator. Thresholds for max-confidence and min-error are set such that when the two conditions are met, the current classifier is assumed to provide high confidence in the classification of the remaining unlabeled data. The experiments carried out by Zhu and Hovy indicate that min-error is a good choice of stopping criterion, and that the max-confidence approach is not as good as min-error.

Zhu, Wang and Hovy (2008a) extend the work presented in (Zhu and Hovy 2007) and introduce an approach called *minimum expected error strategy*. The strategy involves estimating the classification error on future unlabeled instances in the active learning process. Zhu and colleagues test their stopping criterion on two tasks; word sense disambiguation, and text classification. Zhu, Wang and Hovy (2008a) conclude that the minimum error strategy achieves promising results.

In addition to the max-confidence and min-error strategies, Zhu, Wang and Hovy (2008b) introduce and evaluate *overall-uncertainty* and *classification-change*. Overall-uncertainty is similar to max-confidence, but instead of taking only the most informative instances into consideration, overall-uncertainty is calculated using all data remaining in the unlabeled pool. Classification-change builds on the assumption that the most informative instance is the one which causes the classifier to change the predicted label of the instance. Zhu and colleagues realize the classification-change-based stopping criterion such that the learning process is terminated once no predicted label of the instances in the unlabeled pool changes during two consecutive active learning iterations. Zhu, Wang and Hovy (2008b) propose ways of combining max-confidence, min-error, and overall-uncertainty with classification-change in order to come to terms with the problem of pre-defining the required thresholds. Zhu and colleagues conclude that the proposed criteria work well, and that the combination strategies can achieve even better results.

Vlachos (2008) suggests to use classifier confidence as a means to define a stopping criterion for uncertainty based sampling. Roughly, the idea is to stop learning when the confidence of the classifier, on an external possibly unannotated test set, remains at the same level or drops for a number of consecutive iterations during the learning process. Vlachos shows that the criterion indeed is applicable to the two tasks he investigates; text classification and named entity recognition carried out using Support Vector Machines, maximum entropy models, and Bayesian logistic regression.

Laws and Schütze (2008) investigate three ways of terminating uncertainty-based active learning for named entity recognition; *minimal absolute performance*, *maximum possible performance*, and *convergence*. The minimal absolute performance of the system is set by the user prior to starting the active learning process. The classifier then estimates its own performance using a held-out unlabeled data set. Once the desired performance is reached, the learning is terminated. The maximum possible performance strategy refers to the optimal performance of the classifier given the data. Once the optimal performance is achieved, the process is aborted. Finally, the convergence criterion aims to stop the learning process when the pool of available data does not contribute to the classifier's performance. The convergence is calculated as the gradient of the classifier's estimated performance or uncertainty. Laws and Schütze (2008) conclude that both gradient-based approaches, that is, convergence, can be used as stopping criteria relative to the optimal performance achievable on a given pool of data. Laws and Schütze also show that while their method lend itself to acceptable estimates of accuracy, it is much harder to estimate the recall of the classifier. Thus, the stopping criteria based on minimal absolute performance as well as maximum possible performance are not reliable.

Tomanek and Hahn (2008) examine two ways of monitoring the progression of learning in the context of a query by committee setting for training named entity recognizers. Their first approach relies on the assumption that the agreement within the decision committee concerning the most informative instance selected in each active learning iteration approaches one as the learning process progresses. Tomanek and Hahn refer to this as the *selection agreement*, originally introduced in Tomanek, Wermter and Hahn 2007a. The motivation for using the selection agreement score is that active learning should be aborted when it no longer contributes to increasing the performance of the classifier; at that time, active learning is nothing more than a computationally expensive way of random sampling from the remaining data.

The second approach taken by Tomanek and Hahn is to calculate the agreement within the committee regarding a held-out, unannotated test set. This is referred to as the *validation set agreement*. The idea is to calculate the agree-

ment on a test set with a distribution of names that reflects that of the data set on which the active learning takes place. In doing so, Tomanek and Hahn (2008) aim at obtaining an image of the learning progression that is more true than that obtained by calculating the selection agreement, simply because the distribution of the held-out set, and thus also the validation set agreement score, is not affected by the progression of the learning process in the same manner as the selection agreement score is. Tomanek and Hahn (2008) carry out two types of experiments. In the first type, the human annotator is simulated in the sense that the active learning utilizes pre-annotated data; the annotated training examples supplied to the system are in fact not annotated by a human at the time the system requests assistance in classifying them, but comes from the pre-annotated corpus. In this type of experiment, the amount of data is typically limited. The second type of experiment conducted by Tomanek and Hahn (2008) involves real human annotators who operate on a substantially larger amount of data, approximately 2 million sentences, as opposed to the at most 14 000 sentences used in the experiments with simulated annotators.

Tomanek and Hahn (2008) find that, for the experiments with simulated annotators (using relatively small amounts of data), both the selection agreement curves and the validation set agreement curves can be useful for approximating a learning curve, thus indicating the progression of the learning process. However, for the experiments employing human annotators and large amounts of unlabelled data, the selection agreement does not work at all. Tomanek and Hahn conclude that monitoring the progress of active learning should always be based on a separate validation set instead of the data directly affected by the learning process. Thus, validation set agreement is preferred over selection agreement.

Of the approaches to defining a stopping criterion for active learning reviewed, only the work described by Tomanek and colleagues is explicitly directed towards committee-based active learning. The other approaches involve single classifier active learning strategies.

4.9 Monitoring progress

A very common way of visualizing how an active learner behaves is by plotting a learning curve, typically with the classification error or F-score along one axis, commonly the Y-axis, and something else along the other axis. It is that *something else* that is of interest here. The X-axis is usually indicating the amount of data seen – depending on the granularity of choice for that particular learning task it can be for instance tokens, named entities, or sentences – or the number of iterations made while learning. The purpose of a learning curve is

to depict the progress in the learning process; few variations of how to measure progress exist, and consequently there are few differences in how the axes of a graph illustrating a learning curve are labeled.

There are times when the graphical nature of learning curves is not an appropriate means to describe the learning process. Abe and Mamitsuka (1998) calculate the *data efficiency* achieved by using an active learning approach as the ratio between the number of iterations required by a base learner to reach top performance when data is drawn at random, and the number of iterations required for the base learner in an active learning setting to reach the same performance.

Melville and Mooney (2004) defines the *data utilization ratio* – which is similar to the data efficiency introduced by Abe and Mamitsuka (1998) – as the number of instances an active learner requires to reach a target error rate divided by the number that the base learner – Decorate – requires to reach the same error rate. Both *data efficiency* and *data utilization ratio* reflect how good an active learner is at making use of the data.

Baram, El-Yaniv and Luz (2004) propose to use a quantification of the *deficiency* of the querying function with respect to randomly selecting instances from which to learn. The deficiency is defined in equation 13.

$$Deficiency_n(A) = \frac{\sum_{t=1}^{n}(Acc_n(L) - Acc_t(A))}{\sum_{t=1}^{n}(Acc_n(L) - Acc_t(L))} \tag{13}$$

where t is the training set size, $Acc_n(L)$ is the maximal achievable accuracy when using algorithm L and all available training data, $Acc_t(A)$ is the average accuracy achieved by active learning algorithm A and t amount of training data, and $Acc_t(L)$ is the average accuracy achieved using random sampling and learning algorithm L and t amount of training data. The deficiency measure captures the global performance of active learner A throughout the learning process. Smaller values indicate more efficient learning.

There are, of course, more parameters than data-related ones to consider when using active learning in a practical setting, such as time, money, cognitive load on the user, all of which relates to situations beyond the scope of active learning proper as discussed in this chapter. Section 5.4 brings up a number of issues relating to the cost of annotation.

5 ANNOTATION SUPPORT

This chapter concerns methods and tools for the annotation of linguistic content that, in some way or another, utilize knowledge about what is being annotated in order to ease the workload for the annotator. Four dimensions along which the annotation process can be supported are described in the following: *static*, *dynamic*, *intra-container*, and *inter-container* support, respectively.

The support of the annotation process can be described in terms of the ability of the method or tool to dynamically exploit previously made annotations as the process proceeds. Along this dimension, annotation support can be either *static* or *dynamic*. The source of static support does not change within the scope of the annotation process. Dynamic support, on the other hand, takes into account the annotations made previously by the annotator when providing assistance in marking-up data.

Another way of characterizing the nature of annotation support pertains to the level at which the support is available. To be able to describe such support, it is necessary to first introduce the concepts of *markable* and *container*. Exactly what the term "markable" refers to depends on the annotation task. For instance, in part-of-speech tagging, markable refers to tokens; in named entity recognition, markable refers to sequences of tokens; while in text categorization, the term markable refers to documents.

A container contains markables. Analogously to the description of what constitutes a markable, the explanation of the term "container" depends on the annotation task. For instance, in part-of-speech tagging, the container is usually a phrase or a sentence; in named entity recognition, the container is a sentence; in text categorization, containers and markables refer to the same level of abstraction, that is, documents.

The annotation process can be assisted in terms of suggestions regarding the whereabouts and categorization of markables within a container; this type of support is called *intra-container* support. The support of the annotation process can also be described in terms of suggestions about which container to operate on next; this kind of support is called *inter-container* support. Note

that while *intra-container* support does not conflict with *inter-container* support, the notions of *static* and *dynamic* support do. Also note that there is no need to distinguish between intra-container and inter-container support if the container and the markable are described at the same linguistic level, such as is the case in text categorization.

In *inter-container* support, new markables to be annotated are suggested, for instance sentences to mark-up for named entities, or sequences of words to annotate for part-of-speech. System-provided help at the inter-container stage is usually based on active learning for container selection. There is, however, a fundamental difference between active learning as such, and annotation support based on active learning; where active learning focuses on increasing the performance of a learner without having to supply more than necessary training data, the active learning-based annotation support focuses on increasing the quality of the data at the lowest possible cost. These two views are not necessarily incompatible, but there are issues to be dealt with at their intersection, for instance *annotation cost* and *data re-use* (see sections 5.4, and 5.6, respectively).

Focusing on the annotation process brings forward an issue which is as crucial in active learning as in active learning-based annotation support, but which is more often than not neglected in research concerning the former; the actual interaction between the human annotator and the system. Current research on active learning commonly simulates the human oracle by using pre-annotated data. Consequently, active learning research rarely address the issue of user-system interaction. In active learning as annotation support, on the other hand, the interaction between the annotator and his tool is a major concern. To this end, several systems described in the literature incorporate some form of interaction model. Regardless of, for instance, what kind of annotation task is to be carried out, and whether it is carried out by a single annotator in isolation, or by a team of annotators, there are some critical elements in the interaction that are likely to be present in all annotation tasks. Interaction issues are further discussed in section 5.5.

5.1 Static intra-container support

By and large, the static approach to pre-tagging, outlined in figure 5.1, has been successfully applied when, for instance, creating the Penn Treebank corpus (Marcus, Santorini and Marcinkiewicz 1993), assigning part-of-speech tags to the GENIA corpus (Tateisi and Tsujii 2004), building a biomedical proposition bank (Chou et al. 2006), and marking up corpora for named entities (Vlachos 2006; Ganchev, Pereira and Mandel 2007). Here, the knowledge source – the software performing the pre-tagging – is usually static.

1. The corpus is annotated by means of an existing tagger.
2. Human manually corrects suggestions made by the tagger.

Figure 5.1: Outline of a system-initiated intra-container pre-tagging scheme.

Marcus, Santorini and Marcinkiewicz (1993) describe two experiments conducted when constructing the Penn Treebank corpus consisting of one million words of English coming from a number of different domains. The first experiment concerns the part-of-speech tagging of the corpus, while the second deals with marking up parse trees in the text. For the part-of-speech tagging, Marcus and colleagues employ the method outlined in figure 5.1. Pre-tagging was made by means of a combination of stochastic and rule-based taggers based on the Penn Treebank tag set; the error rates obtained are reported to range from 2–6%. The correction stage is facilitated by a common text editor – Emacs – equipped with specially written software allowing the annotators to correct part-of-speech errors as easily as possible. Marcus, Santorini and Marcinkiewicz (1993) report on an experiment examining two modes of annotation; tagging and correction. In tagging mode, the unannotated texts were hand-tagged, while in the correction mode, the human annotators corrected the suggestions made by a tagger in a pre-tagging step. Marcus and colleagues show that the semi-automatic approach to part-of-speech tagging is superior to annotating the texts from scratch in all of the three ways of measuring performance: tagging speed, consistency between annotators, and tagging accuracy.

Tateisi and Tsujii (2004) use the Penn Treebank tag scheme when marking up the GENIA corpus with part-of-speech information. GENIA consists of research abstracts from the biomedical domain. The approach adopted by Tateisi and Tsujii (2004) differs from that of Marcus, Santorini and Marcinkiewicz (1993) in that Tateisi and Tsujii first construct a list of frequent terms specific for the GENIA corpus. The list is then manually inspected by domain experts and each term is assigned the appropriate part-of-speech tags. This effort is called for by the nature of the text in this domain; Tateisi and Tsujii identify a number of characteristics that cause problems for the annotators, the most prominent being that capitalized names and abbreviations make the distinction between proper and common nouns hard. The information contained in the list of technical terms manually excerpted from the corpus forms the basis for the pre-tagging process; a part-of-speech tagger considers that information when assigning tags to the words in the surrounding context. During the correction phase, the human annotator is made aware of which tags in the text stem from

the list of terms, such tags are less likely to require correction, and the annotator can concentrate on the ones introduced by the tagger. Tateisi and Tsujii (2004) conclude that the corpus is consistently annotated, by non-domain experts, using the proposed method. Their error analysis shows that the technical terms still constitute the problematic cases.

While the pre-tagging stage in the efforts of Marcus and colleagues, and Tateisi and Tsujii, relies on a part of their respective corpora being manually tagged, the approach adopted by Chou et al. (2006) is fundamentally different. Chou and colleagues utilize an existing corpus from one domain to train a semantic role labeling classifier intended to operate on another domain. The classifier is trained on predicate argument structures superimposed on news wire text in the Penn Treebank, while the target texts are taken from the GENIA corpus, that is, in the biomedical domain. The annotation process consists of four steps, approximately following the pre-tagging method outlined in figure 5.1; identify predicate candidates constituting relations between named entities in the target corpus, automatically mark-up the semantic roles of the predicate candidates using the classifier trained on news wire text, transform the output to a format suitable for the WordFreak tool (Morton and LaCivita 2003), and finally use WordFreak to manually correct the annotations introduced by the classifier. Chou et al. (2006) claim that their method can significantly reduce the annotation effort required.

The differences in characteristics between domains, regarding some linguistic phenomenon, can to some extent be seen as an introduction of noise. That is, the change in how, for instance, names are expressed in domain *B* with respect to how they are expressed in domain *A*. This observation is exploited by Vlachos (2006) when he uses pre-tagging for marking up named entities. The main idea in Vlachos's approach is to pre-tag the corpus using an existing named entity recognizer, and then use active learning techniques to select those automatically tagged names that need to be corrected by the user. In order to facilitate this approach, Vlachos (2006) first experiments with various types and amounts of artificially created noise in the training corpus forming the base for the pre-tagger. The rationale behind this approach is that if automatic pre-tagging is to be used, but not all potential errors introduced in that step are to be corrected, it is imperative to gain knowledge about how the various kinds of errors affect the quality of the final corpus, and thus also how machine learning schemes applied to the corpus may or may not pick up on the errors. An error that does not propagate to the output of a classifier trained on the erroneous corpus need not be corrected. Vlachos used the LingPipe toolkit (Alias-I 2008) to train named entity recognizers on the noisy texts, and evaluation then took place on a designated test set. Vlachos (2006) makes two specific observations. The first is that limited noise does not significantly affect

the tagging performance, this, in turn, suggests that not all mistakes introduced in the pre-tagging stage need to be corrected. The second observation is that the number of errors in the training text is not a good indicator of the accuracy obtained by a tagger trained on the text. Vlachos's work includes selecting the erroneous instances of names for the human annotator to correct, something which will be further discussed in section 5.3.

In the scenario envisioned by Ganchev, Pereira and Mandel (2007), half the corpus has already been manually annotated, which would entail high quality mark-up. The goal is to annotate the remaining, untagged half of the corpus with minimal human effort. The task at hand is that of identifying and categorizing names of genes in text. Ganchev and colleagues propose the following method: use the manually tagged part of the corpus to train a high-recall tagger; then use the tagger to mark up the unannotated part of the corpus; finally, have a human annotator filter the suggested annotations. The filtering is set up as binary decisions; either a suggested named entity is correct, or it is not. In the latter case, the annotation is easily discarded. Regarding the effectiveness of their approach, Ganchev, Pereira and Mandel (2007) point out that they need to process *more* data in order to reach a given accuracy than they would have had if they had refrained from the binary decision approach and instead annotated all data from scratch. However, if the human effort is factored in, the cost of annotating the data needed is lower using the method proposed by Ganchev and colleagues, than manually annotating the data.

The work presented in Ganchev, Pereira and Mandel 2007 accentuates a crucial issue present in the forms of pre-tagging introduced here. It concerns the correction of suggested annotations. There are two strands. Either the use of pre-tagging is embraced and the correction phase is viewed as a necessity which cannot be avoided, or everything is done in order to prevent a situation in which corrections have to be made. Although the use of pre-tagging appears to be beneficial to tagging speed and quality of the marked-up corpora, the question remains of the amount of human labor required in correcting suggested annotations versus that required for inserting them from scratch.

5.2 Dynamic intra-container support

The dynamic approach to providing intra-container support is described in figure 5.2. Examples of systems utilizing this include the Alembic Workbench (Day et al. 1997), Annotate (Brants and Plaehn 2000), and Melita (Ciravegna, Petrelli and Wilks 2002). These systems all have the capability to learn from the mark-up made by the user as the annotation process proceeds. In effect, this means that as the system picks up on the way the user annotates data,

1. The human manually annotates a portion of the corpus.
2. The system learns from the annotations.
3. The system presents, to the human, suggestions as to what annotations can be inserted in the text.
4. Human manually corrects the suggestions made by the system.
5. Repeat 2 to 4 until some stopping criterion is met, for instance that all available data has been annotated.

Figure 5.2: Outline of a human-initiated intra-container pre-tagging scheme.

the annotation process will eventually turn into one where the user reviews system-made suggestions instead of creating annotations from scratch.

The Alembic Workbench mixes hand-coded rules with machine-learned ones in order to bootstrap corpus development for use in information extraction (Day et al. 1997, 1998). This is achieved by using a mixed initiative approach and keeping the human in the loop when annotating, largely following the method outlined in figure 5.2. As a side effect, if somewhere in the process of corpus development, the automatically tagged corpus is of such high quality that the human annotator does not have to correct the machine, the learned rules constitute a domain-specific tagging procedure themselves. The sequence of rules can then be applied to new, previously unseen, documents from which information is to be extracted.

In the Annotate system (Brants and Plaehn 2000), the user has access to a graphical interface for manipulating syntactic structures. Annotate has access to a part-of-speech tagger as well as a parser, and all changes made by the user in the interface are immediately reflected in the analysis provided by the interactive tools. In the method presented by Brants and Plaehn, the part-of-speech tagger is first used to assign tags to each word in the current sentence. Based on the probabilities of the tags, Annotate distinguishes between reliable and unreliable tags, the unreliable ones are presented in such a way that the human annotator is made aware of them and either confirms or corrects them. Then the parser is activated and it incrementally suggests analyses of phrases based on the previously analyzed, partial syntactic structures. If the user does not find any of the suggested analyses appropriate, he has the option of providing a correct one himself. Brants and Plaehn (2000) point out that their approach has two advantages over manual annotation (and thus also over the more general approach described in figure 5.2). The first one is that the human annotator is

guided, step-by-step, through the structure assigned to the input, meaning that the user actively looks at each phrase. The second advantage is that the parser is able to immediately exploit corrections made by the user.

Melita (Ciravegna, Petrelli and Wilks 2002) incorporates an information extraction system into the annotation process for the semantic web. As such, the extraction system would automatically annotate things in web pages it has learned by observing a human annotator. Melita operates in two phases: *training* and *active annotation with revision*. The training phase, in turn, is divided into two sub-phases: *bootstrapping* and *training with verification*. During the former, the system learns from the annotations made by the human. During the latter, the system silently competes with the user, utilizing the manually created annotations as answer key. At this stage, the user may choose to use the system as an support by adjusting the level of pro-activity, causing it to enter the second main phase of operation – active annotation with revision – where the system explicitly suggests annotations, and where any corrections made by the user are used as training data by the system. Ciravegna, Petrelli and Wilks (2002) claim that the supervision required at this stage is faster, and also likely to be less error prone than producing new annotations from scratch. The main concern of Ciravegna and colleagues, is to control the timeliness and intrusiveness of the system with respect to the human annotator and the annotation process. These issues will be discussed in section 5.5.

Similar to the Annotate system, the work devised by Culotta et al. (2006) takes into account any corrections made by the user. Culotta and colleagues use the terms *corrective feedback* and *persistent learning* to refer to the ability to make use of user-provided feedback, and the ability to update the learned model with the corrections, respectively. Their argument is that by designing more efficient mechanisms for soliciting feedback from the user, a more effective use of persistent learning is enabled. The application area addressed by Culotta et al. (2006) is that of interactive information extraction concerning the recognition of contact information available in various textual sources. Culotta and colleagues present a method for propagating user corrections to other parts of the analysis under consideration; if a part of the suggested annotation is corrected by the user, the other parts of the predicted analysis are automatically updated. Furthermore, Culotta et al. (2006) show how to determine which parts of the automatically assigned structure should be corrected by the user first in order to, for instance, maximize the effect of the error propagation. Using these findings in an experimental active learning-based set-up, Culotta and colleagues demonstrate that by using the proposed corrective feedback techniques, the effort needed for labeling in order to train a extraction classifier is decreased.

5.3 Active learning as inter-container support

Recall the experiment by Ngai and Yarowsky (2000) introduced in chapter 1. They compared active learning-based annotation for learning a base noun phrase chunker to manual construction of rules for the same task based on the same corpus. Note that Ngai and Yarowsky did not compare rule writing to a sequential, randomized annotation process as a baseline, but to what is arguably the best way of achieving informative examples to annotate; selective sampling. In active learning, the data produced is often viewed as a side effect of an annotation process that is really intended to produce a classifier with as little human effort as possible. During the past decade, researchers have begun exploiting active learning with a focus on producing annotations, examples include corpora marked-up for parse trees (Hwa 2000), word senses (Chklovski and Mihalcea 2002), part-of-speech information (Ringger et al. 2007), and named entities (Vlachos 2006; Tomanek, Wermter and Hahn 2007b, a).

Hwa (2000) uses active learning to select sentences to be marked-up with parse trees. The corpus constructed is then used to induce a statistical grammar. The sample selection method used by Hwa is based on a single learner using sentence length and tree entropy as means to select the sentences to annotate. Hwa points out that creating a statistical grammar (parser) is a complex task which differs from the kind of classification commonly addressed by active learning in two significant respects; where a classifier selects labels from a fixed set of categories for each instance, in parsing, every instance has a different set of possible parse trees. While most classification problems concern a rather limited set of classes, the number of parse trees may be exponential with respect to the length of the sentence to parse. These two characteristics have bearing towards the complexity of the task faced by the human annotator acting as oracle in the learning process. Hwa's aim is to minimize the amount of annotation required by the human in terms of the number of sentences processed, as well as in terms of the number of brackets denoting the structure of each sentence. Hwa (2000) thus acknowledge that the success of an active learner may not be as simple to measure as the data required to reach a certain level of accuracy. Issues pertaining to the human effort needed to produce annotated data will be further discussed in section 5.4.

Chklovski and Mihalcea (2002) direct their efforts towards collecting a word sense-tagged corpus involving the general public as annotators by using the World Wide Web as communications channel. The active learning component used to select instances to tag is made up from two classifiers created by two different base learners. An instance is picked out for annotation if the labels assigned to it by the classifiers are not equal.

In setting up the experiment, Chklovski and Mihalcea faced two rather uncommon challenges; that of ensuring the quality of annotations provided by a potentially very large and uncontrolled selection of annotators, and that of drawing attention to their task in order to bring in enough annotators. Chklovski and Mihalcea dealt with the first challenge by limiting the number of tags of an item to two, and also by limiting the number of tags assigned to an item to one per contributor. The authors proposed to make the tagging task "game-like", including awarding prizes to the winners, in order for people to be inclined to contribute. Mihalcea and Chklovski (2003) report that after the first nine months of being available on the web, their system had collected 90 000 high-quality tagged items.

Ringger et al. (2007) approach the task of part-of-speech tagging of poetry from the British National Corpus, and prose from the Penn Treebank. In their scenario, the constraining factor is the budget – they can only afford to annotate so many sentences – causing Ringger and colleagues to examine active learning as a way of carefully selecting the sentences to be processed by the human annotator. Ringger and colleagues try out query by uncertainty as well as query by committee, both using a Maximum entropy conditional Markov model (McCallum, Freitag and Pereira 2000) as base learner. In query by uncertainty, the uncertainty is derived from an approximation of the per-sentence tag sequence distribution entropy. The committee members used in query by committee divide the training set evenly, and disagreement is settled using the total number of tag sequence differences among the committee members, when predictions are compared pair-wise. Ringger and colleagues find that one of their variants of query by uncertainty yields the best results. Once the active learning procedure is complete, that is, when the budgetary limit is reached, the idea is that the part-of-speech tagger trained on the data annotated so far is to be used to mark up the rest of the unannotated data. Ringger et al. (2007) do not report on how they assess the quality of the automatically inserted part-of-speech tags; a correction phase does not seem to fit into their initial scenario, rather, the performance of the generated tagger on a designated test set serves as an indirect quality assessment of the resulting corpus.

Vlachos (2006) approaches the production of marked-up data slightly different than the rest; instead of employing active learning for the purpose of selecting sentences to annotate with names, Vlachos uses it to select the automatically inserted named entity annotations that need to be corrected. In his experiments, the active learning follows a phase where an unsupervised named entity tagger is used to pre-tag the texts; the use of such a tagger is one of two points distinguishing Vlachos's method from the prototypical active learning outlined in figure 4.1. The second difference is the way sample selection is carried out; its purpose is not to identify probably informative unannotated

sentences, but rather to identify likely errors introduced in the data by the pre-tagger, and bring those to the attention of the human annotator. Active learning is realized by using a Hidden Markov Model base learner in a query by uncertainty setting where the uncertainty of a sentence is represented either as the average uncertainty of the tokens, or as the most uncertain token in the sentence. To assess the effects of errors introduced by the pre-tagger, a number of taggers is created by training on a corpus contaminated with various kinds and degrees of noise (as mentioned in section 5.1). The error detection executed in the active learning step is based on the assumption that errors are instances that are hard for the classifier to predict, and at the same time inconsistent with the rest of the data. Vlachos (2006) claim that the inconsistency of an instance can be indicated by a mismatch between the label assigned to the instance by the pre-tagger, and the label assigned to it by the model learned from the data corrected so far. As a measure of "hardness", the uncertainty of the current model's prediction of the instance label is used. Vlachos finds that his approach to use active learning to select errors to correct outperforms active learning for selecting instances to annotate in all cases except for the one where very noisy data had been used to train the initial pre-tagger.

Tomanek et al. (2007a; 2007b) describe the Jena annotation environment (Jane), a client-server-based workbench for accommodating and managing the labeling of texts. The task described by Tomanek and colleagues is that of named entity recognition in the immunogenetics domain, although they point out that other types of tasks are possible too. Jane contains tools supporting a single annotator, as well as for managing teams of annotators. The administrative tool facilitating the latter include modules for managing users, creating and editing projects, monitoring the annotation progress and inter-annotator agreement, and deploying the data once the annotation of it is completed. Single annotators have the option to choose focused annotation, that is, being supported by a server-side active annotation module that selects the sentences to be marked-up. Active learning is realized as query by committee employing three different base learners – conditional random fields (Lafferty, McCallum and Pereira 2001), maximum entropy, and Naïve Bayes – with vote entropy as disagreement quantification. Tomanek et al. (2007a; 2007b) perform a real-world annotation experiment, indicating a reduction in the number of annotations with between 50% and 75%. One conclusion drawn from the experiment is that active learning is particularly advantageous when the instances of interest are sparsely distributed in the texts. The density of named entities, that is, the number of entities per sentence, in the corpus produced in the experiment is almost 15 times greater than the density of names in the test corpus. Another set of conclusions is realized as a list of requirements for facilitating deployment of active learning in practical circumstances:

- the turn-around time for selecting what to annotate needs to be kept short;
- the data produced in the annotation process need to be re-usable by other machine learners; and,
- the criterion for stopping the active learning needs to be sensitive to the progress of the performance in the annotation process.

The requirements list presented by Tomanek and colleagues is one of several manifestations of increasing awareness in the research community of the conditions under which the human annotators operate. Concerns are raised regarding the usefulness of the resulting data sets for tasks other than that which originally created the data. Questions regarding the cost of annotation, interaction, and re-usability of data are dealt with in the sections 5.4, 5.5, and 5.6, respectively. The issue of stopping criteria is elaborated on in section 4.8.

5.4 The cost of annotation

Methods for providing the human annotator with support within containers, as well as between containers, raise concerns regarding the effort needed to produce correct annotations. Cost in this sense is typically derived from monetary, temporal, or effort-based issues. The present section is related to section 4.9, but whereas that section concentrates on ways of measuring and illustrating the success of active learning in terms of classifier performance versus amount of data used, the purpose of this section is to shift the focus from learner/data to learner/teacher.

Opting to use active learning in the first place is mainly due to the possibility it opens up to obtain high performing classifiers with less data. But what if the data comes at a cost that is not accommodated for in a practical setting? A cost model should reflect the constraints currently in effect; for instance, if annotator time is more important than the presumed cognitive load put on the user, then the overall time should take precedence in the evaluation of the plausibility of the method under consideration. If on the other hand a high cognitive load causes the users to produce annotations with too high a variance, resulting in poor data quality, then the user situation may have to take precedence over monetary issues in order to allow for the recruitment and training of more personnel.

Using a scale mimicking the actions made by the user when annotating the data is one way of facilitating a finer grained measurement of the learning progress. For instance, Hwa (2000) uses the number of brackets required for

marking up parse trees in the training data as a measure of cost, rather than using the sheer number of sentences available.

Osborne and Baldridge (Osborne and Baldridge 2004; Baldridge and Osborne 2004) distinguish between *unit cost* and *discriminant cost* in their work on ensemble-based active learning for selecting parses. In their setting, unit cost is the absolute number of sentences selected in the learning process. Discriminant cost assigns a variable cost per sentence and concerns the decision an annotator has to make concerning the example parse trees provided by the system.

Culotta et al. (2006) design their system so that segmentation decisions are converted into classification tasks. They use what they refer to as Expected Number of User Actions (ENUA) to measure the effort required by the user to label each example in an information extraction setting. The ENUA is computed as a function of four atomic labeling actions, corresponding to annotating the start and end boundaries, the type of a field to be extracted, as well as to an option for the user to select the correct annotation among k predicted ones. The use of ENUA reflects the authors' goal with the system; to reduce the total number of actions required by the user. Culotta et al. (2006) notice that there is a trade-off between how large k is, that is, how many choices the user is presented with, and the reduction in amount of required user actions caused by introducing the multiple-choice selection.

It is easy to see that, on average, it must be harder to annotate the units provided by active learning, than it is annotating units randomly drawn from the same corpus simply because the former is, by definition, more informative. Along these lines, Hachey, Alex and Becker (2005) find that the three selection metrics they used in a live annotation experiment yield three distinct annotation time per token/data size curves. Hachey and colleagues measure maximum Kullback-Leibler divergence, average Kullback-Leibler divergence and F-complement for selecting the sentences in which the annotators are to mark up named entities. Hachey, Alex and Becker (2005) demonstrate that the time spent on marking up an example is correlated with its informativeness. Similarly, in the experiments conducted by Ringger et al. (2007), the selection metric resulting in the best performance in terms of amount of data needed to reach a given accuracy, is also the one demanding the most attention from the user; the amount of corrections made by the oracle is clearly larger for the most complex selection method used.

Haertel et al. (2008) show that the active learning strategy which performs the best depends on how the annotation cost is measured. They examine the performance of query by uncertainty and query by committee for the task of part-of-speech tagging under a number of cost models. The models used include what Haertel and colleagues refer to as an hourly cost model. As an

example of how a cost model can be used in their particular setting, Haertel et al. (2008) show that when a low tag accuracy is required, random selection of what to annotate is cheapest according to the hourly cost model. On the other hand, query by uncertainty is cost-effective (compared to a random baseline) starting at around 91% tag accuracy, while query by committee is more effective than both the baseline and query by uncertainty at tag accuracies starting at around 80%.

So far, this section has been all about inter-container support. What about the various forms of pre-tagging used as intra-container support, what kind of cost do they imply? It seems as if the main concern relates to the correction phase usually taking place after the automatic insertion of annotations in the data. The types of corrections, or actions, needed are more often than not the focus for researchers. For instance, Brants and Plaehn (2000) designed their system, Annotate, so that the user can assemble an annotation from system suggested parts instead of having the user simply correct complete annotations suggested by their system. Brants and Plaehn claim that their way of doing it is much faster and less error prone.

Sometimes failure is success, depending on along which scale the results are measured. Ganchev, Pereira and Mandel (2007) design their process so as to require binary decisions only, as opposed to full annotations/corrections, from the user. They show that the effectiveness of their approach is inferior to that of learning from fully manually annotated data. Their semi-automatic method requires the annotator to process more data than he would have had if he had chosen to manually annotate it. This is an effect of reducing the load on the user to binary decisions. On the other hand, Ganchev and colleagues show that less effort is required by the annotator to produce annotations of a quality superior to that of manually tagging. In all, Ganchev, Pereira and Mandel (2007) conclude that, in the conducted experiments, the suggested semi-automatic method reduced annotation time by 58%.

5.5 Interaction issues

None of the things described in the previous section as pertaining to the cost of annotation is separable from the pivotal role of user-system interaction. From the user's point of view, annotating is all about interacting. It is clearly not possible to contrive a general interaction model applicable to all sorts of annotation tasks, since such a model unavoidably depends on task specific parameters such as: the knowledge, training, and numbers of participating annotators; the monetary, and temporal frames of the task; and, the purpose of the creation of the data. Nonetheless, a number of interaction-related things emerge as be-

ing universally important and ought to receive attention, regardless of task; *timeliness*, *intrusiveness*, and *degrees of freedom*.

Timeliness (turn-around time, idle time) and *system intrusiveness* are related in that one entail the other. If the user's work flow is interrupted due to long turn-around times on behalf of the system, the system is often perceived as being intrusive. Ciravegna, Petrelli and Wilks (2002) describe an experiment involving an adaptive information extraction system that accommodates these two issues. Intrusiveness as a side effect of system proactivity is dealt with by presenting the user with a control in the form of a single slider allowing the user to set thresholds indicating when the system should suggest annotations, and when it should not. Ciravegna and colleagues find that the notion of intrusiveness evolves, and the user finds himself understanding it as an effect of using the system. They also point out that the acceptable level of intrusiveness is subjective, and that the interaction design should empower the user to select a suitable level of intrusiveness, without having to fully understand the underlying mechanisms involved. Drawing the observation a step further, it is easy to envision that a given user may perceive the same proactive behavior by the system as moving from non-disturbing to disturbing.

In the case of active learning-based annotation, timeliness refers to the ability of the system to ask the user to annotate pieces of data in a timely manner. That is, without interrupting the users flow. Ciravegna et al. (2002) address this matter by using two annotation systems in parallel such that while the user annotates document n, in which annotations are suggested by a system trained on annotated documents $1,...,n-2$, the other system trains on documents $1,...,n-1$. Adopting this approach results in a "blind" spot of the systems, where the user can face two very similar consecutive documents to annotate. Obviously, the number of parallel systems to use depends on the time it takes for one system to train, related to the time the user needs to annotate one document.

The options concerning the number of seemingly valid choices for the user is referred to as the *degrees of freedom* available in the interaction process. Efforts by researchers to reduce the freedom in order to guide the user to making correct decisions include reducing the annotation task to that of classification, for instance in marking up phrase boundaries, or assigning parse trees to sentences (Osborne and Baldridge 2004; Culotta et al. 2006; Ganchev, Pereira and Mandel 2007). Another way is to have the user making informed choices by stimulating him to assemble a complete annotation from parts suggested by the system, and thereby encourage him to carefully consider each choice possible (Brants and Plaehn 2000).

5.6 Re-use of annotated data

Under some circumstances, active learning can evidently be used to identify the most informative units in a corpus. What really takes place is the reordering, and elicitation, of available examples highly biased towards the preferences of the base learner and task in effect. The re-usability of the data created in the process is the subject matter of a number of research efforts.

Baldridge and Osborne (2004) use active learning to create a corpus on which to train parsers. In doing that, their principal worry is whether the selected material will be useful if used with a base learner different than the one used to select the data. Indeed, for their particular task, they find that the gains of using active learning may turn out minimal or even negative. The reason lies in how complex it is for a human annotator to assign parse trees to the selected sentences. In response to this finding, Baldridge and Osborne formulate a strategy involving semi-automatic labeling to operate in concert with active learning. The semi-automatic technique makes use of the fact that the parser used can provide ranked partial parses, of which the ones with higher probability than chance is presented to the user in a drop-down list. Baldridge and Osborne (2004) conclude that *n*-best automation can be used to increase the possibility of the annotations produced being re-usable.

Tomanek, Wermter and Hahn (2007a) conduct two experiments addressing the re-usability of training material obtained by employing active learning to annotate named entities in a biomedical domain. In their first experiment, the base learners used for selecting data – called *selectors* – and the learning schemes used for testing the data – called *testers* – are varied. By keeping the feature set fixed and using the best selector, generated by a conditional random field base learner, Tomanek and colleagues show that feeding the tester with data generated by faster, albeit worse performing selectors based on maximum entropy and naïve Bayes, still yield results far better than passive learning. Comparable results are reported for the variation of the tester's base learner.

In the second experiment outlined in Tomanek, Wermter and Hahn 2007a, the feature sets used by the selectors are reduced, while that of the tester remain fixed and full. The three reduced feature sets contain, in turn, all but the syntactic features, all but the syntactic and morphological features, and finally, only orthographic features. A committee of conditional random field selectors employs each of the three reduced feature sets. Tomanek, Wermter and Hahn (2007a) show that, even when using selectors in concert with the most reduced feature set, a tester (also based on conditional random fields) still can make use of the data and generate results better than those resulting from passive learning.

Vlachos (2006) points out that his approach – pre-tagging followed by an active learning phase in which erroneously marked-up examples are selected for correction by an oracle – is likely to yield data more re-usable than the data created using "ordinary" active learning. This claim is based on the observation that the corpus produced by Vlachos's method contains all data that the initial corpus does, and although only parts of the data is manually corrected, the errors in the uncorrected parts are possibly non-significant to a machine learner. Since the distribution of the data in the resulting corpus is the same as in the original one, the former is likely to be as re-usable as the latter.

Part II

BootMark – A bootstrapping method for marking up named entities in textual documents

6 BOOTSTRAPPING THE MARK-UP OF NAMED ENTITIES IN DOCUMENTS

This chapter presents a three-phase method – BootMark – for bootstrapping the annotation of named entities in textual documents. The first phase serves as a seeding stage for the method, while the second phase constitutes the bootstrapping proper, and in the third and final phase, human annotation efforts take on the form of revising system-suggested mark-up instead of creating annotations from scratch.

When applicable, the active learning paradigm has the desirable effect of creating high-performing classifiers using less data than required by competitive classifiers trained on a random selection of data. The BootMark method is an attempt to orchestrate active learning in such a way that the focus of the overall process is on creating high quality annotated corpora instead of on creating optimal classifiers; obtaining a classifier able to operate on the introduced annotations is but a secondary goal. BootMark is primarily set out to facilitate the creation of corpora annotated with named entities, and uses the named entity recognizers created in intermediate steps as means to achieve that goal. The main idea is to favor the creation of data over the creation of classifiers by selecting whole documents for annotation with named entities, rather than selecting sentences; the principal result from applying this method is a corpus of documents annotated with named entities, instead of a collection of annotated but possibly non-related sentences. The motivation in BootMark for focusing on the document level and choosing the document as the smallest unit for building a corpus pertains to the need for annotated documents in building or adapting information extraction systems.

6.1 What this method description is not

The BootMark method as presented in this chapter is a high-level description of a method for marking up linguistic content in textual documents. As such, it unavoidably comes with some constraints.

First, despite the intention for the present chapter to be applicable as an implementation blue-print, some steps in the description of the method will by necessity be underspecified. All issues that require more information to be resolved than is available in this method description are identified as such as they are encountered. These matters are collected in section 6.6 as a list of emerging issues in need of further elaboration.

Second, for BootMark to be applicable, it *must* be possible to distinguish between documents by means of the linguistic level at which the annotation takes place. Thus, it must be possible to distinguish between documents, using calculations at the named entity level, in order to find those cases of named entities deemed most informative by the active learner. Depending on, for instance, how the task of named entity recognition is addressed, and the characteristics of the data to work with, it may not be possible to make this distinction between documents. In that case, the performance of BootMark will be nothing more than a computationally expensive way of random selection of documents to annotate.

Third, BootMark does not entail an interaction design or otherwise address the issue of what constitutes an appropriate annotation cost model. These matters are beyond the scope of the current incarnation of the BootMark method, since they do not generalize well between tasks. On the other hand, the BootMark method description as it stands does not in any way hinder the realization of an arbitrary cost model. The way the interaction between a user and a system implementing the BootMark method will be realized is due to details not part of the present description. Such details ultimately refer to, among other things, the specifics of the task, as well as the intended user's expertise and goals, none of which is part of the method as such.

6.2 Prerequisites

To further constrain the description of the proposed annotation method, a number of prerequisites need to be made explicit prior to embarking on the elaboration of BootMark. It is required that:

The unannotated corpus is relevant. The texts comprising the corpus to be annotated are assumed to be of such relevance that none of them is thought of as being an outlier. That is, the goal of applying BootMark is to mark up all unannotated texts available.

Pre-processing tools are available. The unannotated corpus is assumed to be appropriately pre-processed. The pre-processing necessary depends on how the named entity recognition task is addressed, and might include,

for instance, tokenization, part-of-speech tagging, phrase chunking, or parsing. Any resources needed for the conversion between text and formats suitable for machine learning are also assumed to be in place, such as gazetteers or external lexica.

Annotation guidelines are available. The user is assumed to be familiar with, and have readily at hand, any annotation guidelines applicable to the annotation task at hand. The annotation guidelines should also contain definitions of how the quality of annotations should be evaluated.

6.3 Phase one – seeding

The purpose of the first phase of BootMark is to produce a set of annotated documents to be used as a seed for the active learning process in phase two. Phase one is outlined in figure 6.1 and described in the subsequent sections. The legend to the symbols used to describe the control and data flow in the BootMark method is available in figure 6.2.

The prerequisites of phase one include the ones listed above in section 6.2, that is, the presence of a relevant, appropriately pre-processed, unannotated corpus of textual documents, as well as the resources needed to convert the annotated text into a format suitable for whatever machine learning set-up is used, including a specification of features to use for representing annotated material, and the parameters to use in conjunction with a given base learner. A skilled human annotator equipped with the appropriate annotation guidelines is also a prerequisite. The post condition, that is, the result of phase one is primarily a set of annotated documents, initially taken from the unannotated corpus.

6.3.1 Select seed set

The first thing that needs attention is the selection of the seed set, represented as the box labeled *Select seed set* in figure 6.1. The seed set is the set of documents that will be manually annotated by the human annotator. A good seed set is one in which all classes of the information to be marked up are represented in a way beneficial to the base learner, that is, preferably with a distribution between classes as even as possible. Of course, distributional information is unlikely to be available beforehand, and measures have to be taken to approximate such information if it is found to be crucial to the task. Two things concerning the seed set need to be decided on: its size and the constituent documents.

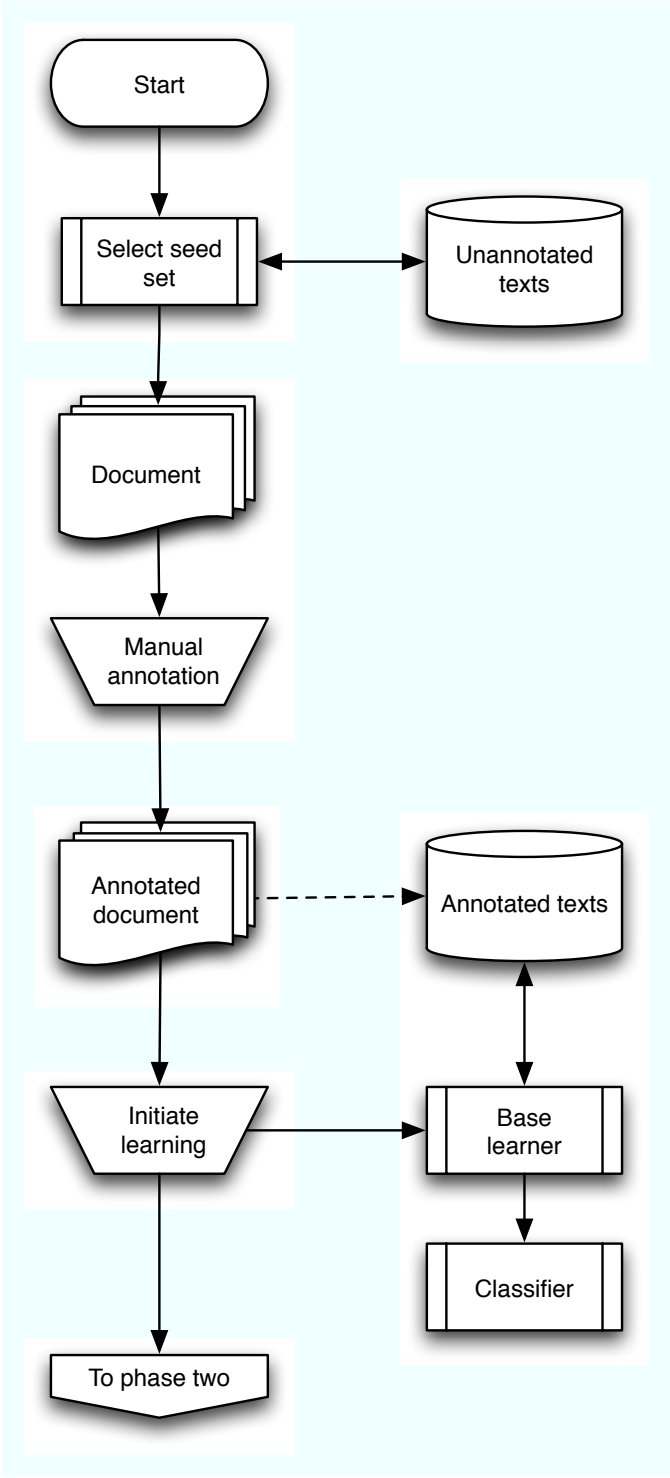

Figure 6.1: Outline of the first phase of the BootMark method.

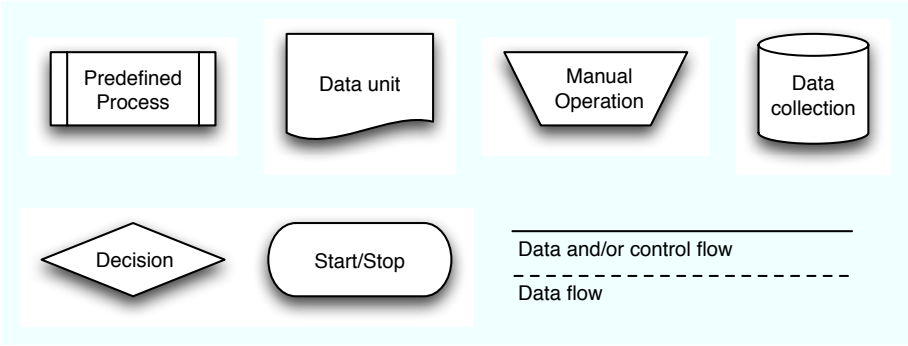

Figure 6.2: Legend to figures 6.1, 6.3, and 6.4.

There is a trade-off between the number of documents in the seed set and the effort required to annotate them. If the effort is not an issue, it could be argued that the number of documents in the seed set should be as large as possible. Now, effort is a constraining factor (and a motivation for the BootMark method proper), and thus should be taken into consideration. The exact number of documents in the seed set depends on the annotation task at hand, as well as on the ability of the human annotator.

There are primarily two ways in which the documents comprising the seed set can be selected: either automatically or by manual inspection. In the literature both ways have their proponents. For instance, Tomanek, Wermter and Hahn (2007b) advocate careful manual selection of a seed set of sentences to be used in active learning for named entity recognition, while McCallum and Nigam (1998) describe a method for learning in the context of text categorization without any seed. Both methods, as well as others, are described in section 4.5.

The issue of how to select a seed set cannot be settled on in the general case, it has to be decided on a per task basis and the issue is therefore deferred to section 6.6 where it is listed as issue E–2: *The constitution of the seed set*.

6.3.2 Manual annotation

The annotation of the selected seed set, represented by the box labeled *Manual annotation* in figure 6.1, is to be made manually, without explicit support. Note that the BootMark method does not presuppose, nor impose a specific way of creating the annotations; an implementation of the method is required to provide this functionality. Thus, the way manual annotation is to be carried out is not within the scope of the present method description.

6.3.3 Initiation of learning and transition between phases one and two

It is not strictly necessary to have the human annotator explicitly initiate the learning process, represented by the box labeled *Initiate learning* in figure 6.1. The initiation may well be coupled to the completion of the annotation of the seed set. The initiation includes the transformation of the annotated material to a data representation suitable for machine learning.

Although it is listed as a prerequisite to phase one, the question of desirable characteristics exhibited by the base machine learner surfaces at this point in the method description. The characteristics are as important and task specific as they are impossible to specify for the general case. Thus, the issue is deferred to section 6.6 in which it is listed as E–1: *Base learner and task characteristics*.

Once the learning from the annotated documents has been completed, the BootMark method proceeds to phase two. This should require no explicit action on behalf of the user.

6.4 Phase two – selecting documents

The second phase, depicted in figure 6.3, constitutes the core of the BootMark method. This is where the bootstrapping of the annotation process takes place. Essentially, phase two corresponds to the prototypical active learning algorithm outlined in figure 4.1, and its purpose is to select in each iteration, from the unannotated corpus, the document that would be most beneficial to mark up in order to increase the performance of the classifier used to select documents.

The prerequisites of phase two are an annotated corpus resulting from phase one, a representational scheme suitable for machine learning of the annotated information, means to convert annotations into that scheme, and an appropriately set-up base learner. Additionally, and most importantly, phase two requires the active learning to be set up (further elaborated on in section 6.4.1). The post conditions of phase two are more annotated data and a classifier for classifying annotations.

6.4.1 Automatic document selection

The automatic selection of the next document to annotate, represented by the box labeled *Select document* in figure 6.3, is the point on which the entire BootMark method hinges. If it is not possible, the purpose of the method is invalidated.

At this point, the classifier trained in previous iterations, or on the seed set if this step is taken for the first time, is used to select the most informative document in the unannotated corpus to be marked-up.

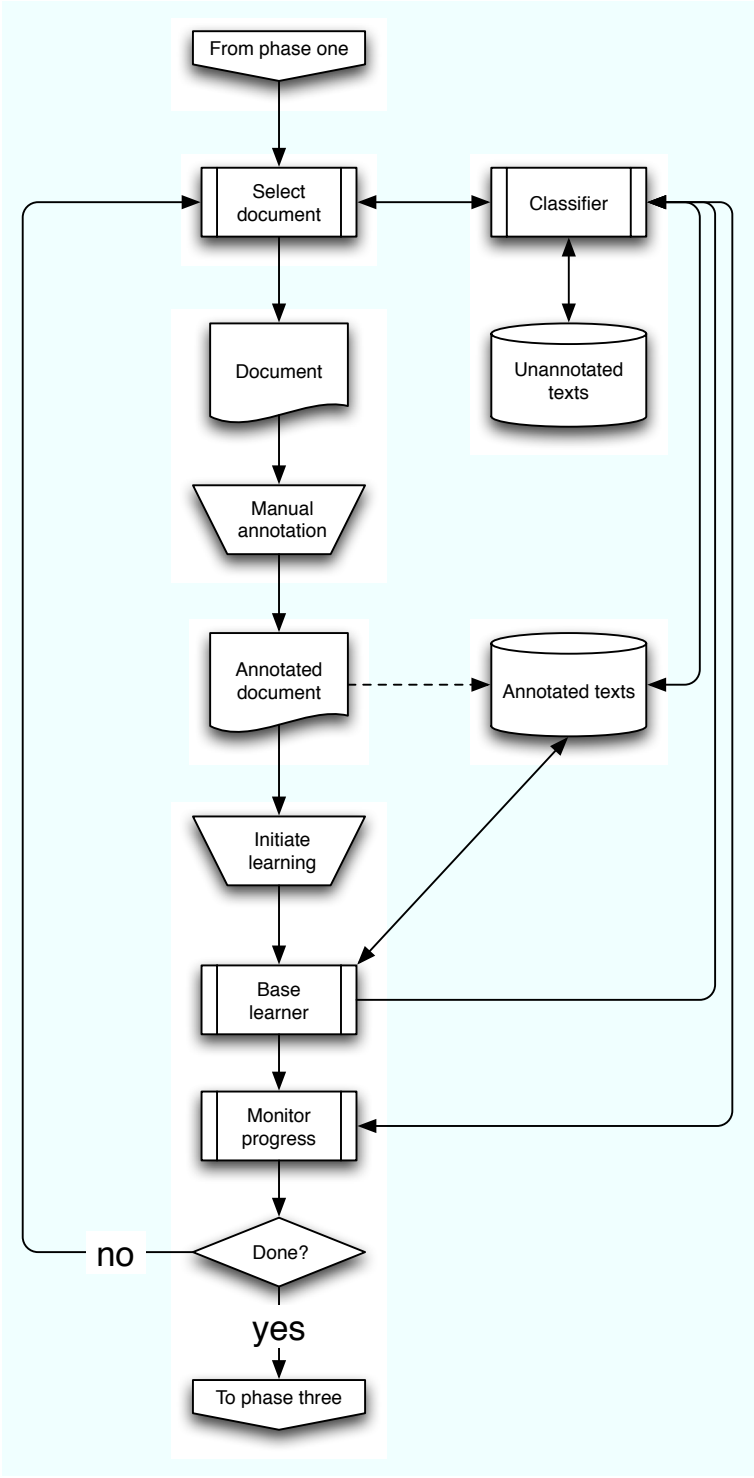

Figure 6.3: Outline of the second phase of the BootMark method.

Before getting to the actual selection, there are a number of decisions to be made, all of which are task specific. Thus, this issue is yet another one that will be need to be resolved at the time of implementation of the method. It is listed in section 6.6 as emerging issue E–3: *Actively selecting documents*.

6.4.2 Manual annotation

The manual annotation taking place in phase two, represented by the box labeled *Manual annotation* in figure 6.3, adheres to the same assumptions as the corresponding step in phase one, described in section 6.3.2.

6.4.3 Initiate learning

As with the initialisation of learning in phase one, section 6.3.3, the initialisation in phase two does not necessarily need to be made explicit; it may well be a consequence of a completed manual annotation. The learning initialization is represented by the box labeled *Initiate learning* in figure 6.3.

6.4.4 Monitoring progress

Although the description of the BootMark method does not include an interaction design proper, some capabilities of a method implementation are believed to be fundamental to the successful annotation of documents. One such basal function is to provide the user with ways of monitoring the progress of the annotation process, represented by the box labeled *Monitor progress* in figure 6.3. There are several ways to think of progress, for instance in terms of the change in performance of the classifier selecting documents, how much data has been annotated, how much data remains to be marked-up, the average time it takes to annotate a document, how long it takes for the system to select a new document to annotate, and so on.

Depending on how the named entity annotation task is realized, and the set-up used for actively selecting documents, some of these ways of monitoring the learning progress are harder to realize than others. For instance, keeping track of the amount of annotated text, as well as timing the annotation process, is likely to be fairly easy. On the other hand, tracking the classifier's performance might turn be a completely different matter. The most straightforward way of accomplishing the latter is to devise a designated test set; in each iteration, the trained classifier is then evaluated on the test set. This is the way that the progress of experiments in active learning usually is scored. The problem with

using a designated test set is that it has to be made readily available, either by using an existing annotated corpus, or more likely, by having the user mark-up documents, possibly while working on the seed set. It is possible that there are other ways of monitoring the learning progress, without using a test set. Information originating from the active learning process may possibly be used to approximate a learning curve as it would have looked had a test set been used, either by resemblance in shape, or by actual performance values.

Since the matter of monitoring the progress is an open one, it is listed in the section concerning emerging issues in section 6.6 as E–4: *Monitoring and terminating the learning process*.

6.4.5 Transition between phase two and three

The matter of going from phase two to phase three pertains to the issues of a stopping criterion for actively selecting documents, represented by the combination of the box labeled *Monitor progress* and the decision whether the bootstrapping phase is completed in figure 6.3. This is an open issue which cannot be given a definite description prior to defining the specifics of the annotation task. It is listed as a part of issue E–4 in section 6.6.

A number of different stopping strategies are outlined in section 4.8. Basically, the learning can either be stopped based on a criterion beyond the control of the human annotator, or the human annotator can himself decide on an appropriate time to stop the learning. If the former scheme is in effect, it seems reasonable to stop learning when the learning does not contribute to selecting documents. In a query by uncertainty setting, such a situation may be implied by the decrease in learning performance such as described by, for instance, Vlachos (2008). In query by committee, the decrease in disagreement among the committee members concerning the classification of the most informative instance selected in each active learning iteration may serve as an indicator of a situation in which the active learning is no longer beneficial to the annotation process. Another indicator is the decrease in disagreement among the members of the committee regarding an unannotated, held-out test set of the same distribution as the data on which the active learning takes place. Both of these indicators are introduced and elaborated on by Tomanek and Hahn (2008). The stopping criterion depends on the active learning paradigm applied, as well as on the selection metric used.

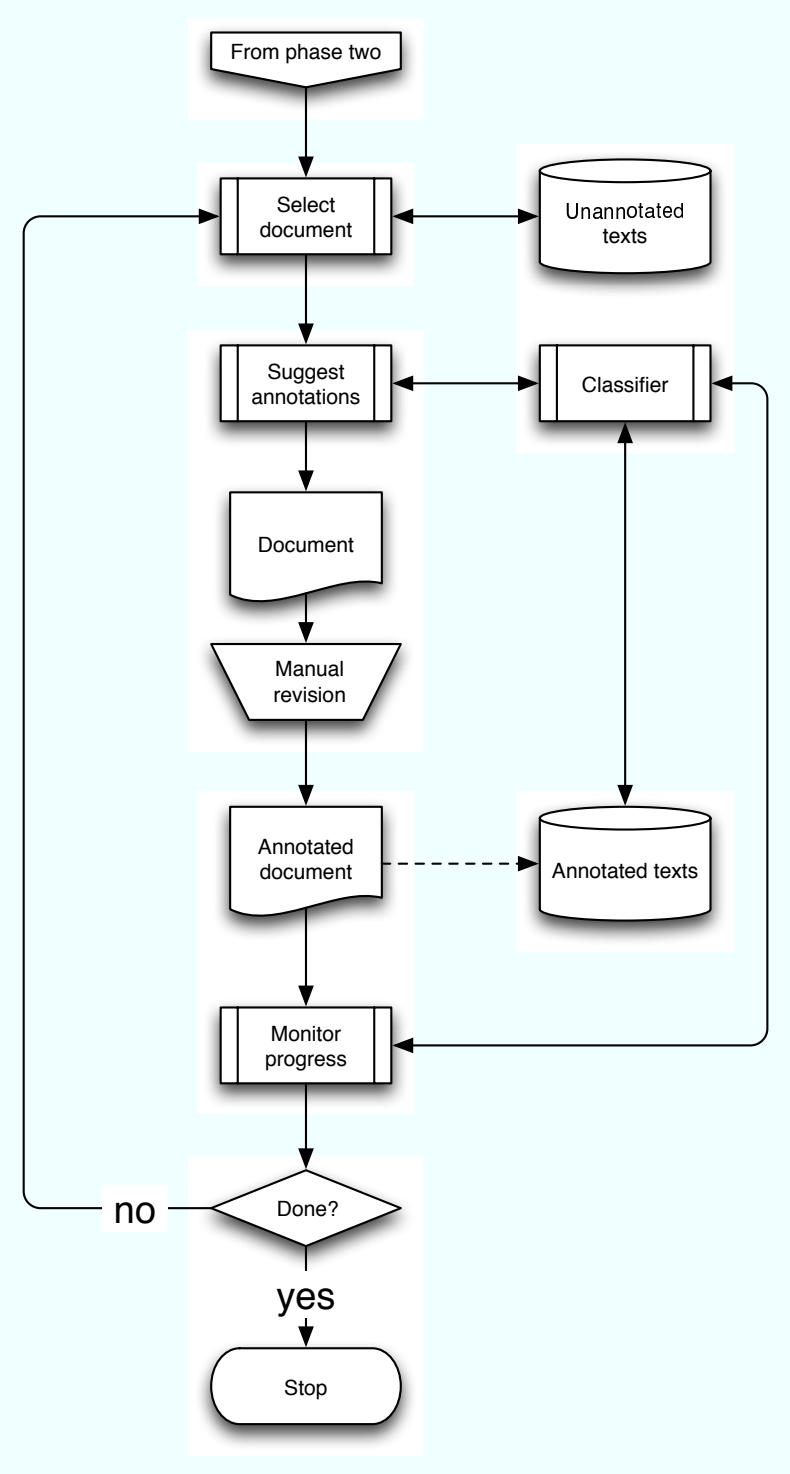

Figure 6.4: Outline of the third phase in the BootMark method.

6.5 Phase three – revising

The third phase of the BootMark method is a stage where the manual annotation process is turned into one of manual revising of the annotations suggested by the system, outlined in figure 6.4. At this point, the active learning that dominated the bootstrapping process in phase two has been terminated and the documents to annotate are selected at random from the corpus of unlabelled documents. A prerequisite of the third phase is the classifier created during the second phase. The idea is to use the classifier as a pre-tagger of the documents yet to be processed, and then have the human annotator manually correct the automatically introduced annotations. The post-conditions of the third phase include a completely marked-up version of the entire original corpus, as well as the classifier originating from phase two.

6.5.1 Selecting documents and suggesting annotations

After having made the transition from phase two, the system selects a document at random from the unannotated corpus. The selected document is then processed using the classifier from phase two in order to generate mark-up suggestions. This stage is represented by the boxes labeled *Select document* and *Suggest annotations* in figure 6.4.

6.5.2 Revising system-suggested annotations

Revising the automatically introduced annotations in a document resembles the manual annotation of a document as described in phase one (section 6.3.2) in that the specifics of how and what to revise is left underspecified. At this stage, represented by the box labeled *Manual revision* in figure 6.4, the human annotator has at his disposal a set of system-suggested annotations to which to react. It is assumed that the classifier used performs so well that it makes sense to use it as a pre-tagger; one of the major objections towards employing pre-tagging in the first place is that it biases the user set out to revise the annotations too much in favor of the automatically inserted mark-ups so that he misses out on true annotations. There is no clear answer to whether pre-tagging is good or bad, thus it is to be resolved on a per task basis. See issue E–5: *Revision of system-suggested annotations* in section 6.6.

6.5.3 Monitoring progress

The purpose of monitoring the annotation progress in phase three is slightly different from that of phase two. Here, it is not the performance of the classifier, and thus the progress of the learning process, that is the primary concern; rather, the goal of monitoring the annotation process in phase three is about assessing and maintaining the consistency of the annotations inserted specifically in this phase. To this end, there are at least two ways of gathering information: either by directly counting the number and types of changes made by the human annotator, compared to the annotations produced by the pre-tagger; or by evaluating the classifier from phase two, at appropriate intervals, on the newly annotated documents. The former way provides direct means of measuring the consistency of the automatically produced annotations with respect to the ones introduced by the human annotator. If the number of human made alterations of annotations increases, then this is a sign that the pre-tagger performance has decreased. The latter way of assessing the consistency of automatically suggested annotations, that is to evaluate the pre-tagger on newly annotated documents and keep track of the performance, provides an indirect way of monitoring the consistency. Both ways can be visualized by plotting the number of changes, or the tagging performance, respectively, against the amount of data processed. The question is what such a curve should look like. It is not a learning curve, and since it is the degradation of the consistency of the system-suggested annotations that is of interest, it seems fair to assume that a flat curve is a good one. Any major deviations from a flat curve should prompt the user to investigate the quality of the newly made annotations with respect to the ones obtained as a result of the bootstrapping process in phase two. A plausible action in a situation where the consistency is found to be inadequate is to re-train the classifier on all available data in order to incorporate recent changes into future predictions.

6.5.4 When to stop annotating

The termination of the BootMark method is a data-centric decision. The obvious point at which to stop the annotation process is when there is enough marked-up material; either when the unannotated corpus is exhausted, or when the user determines the amount of annotated data is enough for whatever purposes it is to be used.

6.6 Emerging issues

The method described thus far admittedly is underspecified in some respects that it should not be. In fact, the implementation of most of the steps in the different phases is open-ended with respect to the task at hand. The items in this section are aggregations of the issues in need of further investigation by anyone interested in implementing the BootMark method.

E–1 Base learner and task characteristics. The realization of the named entity task, including the representation of instances, influences what machine learning schemes are applicable. The behavior of the base learner when applied to randomly selected documents constitutes a baseline with which it is possible to assess the progress of the bootstrapping process; the active selection of documents should yield results better than those accomplished in the baseline case.

How can the target concept be represented? What candidate base learners are there? What are good parameter settings for those base learners with respect to the data available? What are the characteristics of the base learners with respect to the task at hand in terms of time to train it, time to apply it, and accuracy?

E–2 The constitution of the seed set. The compilation of the set of documents constituting the initial seed from which to start the annotation process is crucial.

How many documents should be in it? How are the documents best selected?

E–3 Actively selecting documents. As previously pointed out, the applicability of the BootMark method hinges on the ability to actively select documents with respect to the linguistic level at which the annotation is to be made.

What active learning paradigms are applicable? What disagreement or uncertainty metrics are available for those learning paradigms?

E–4 Monitoring and terminating the learning process. Are there other means to make visible the learning progress than that of providing a designated held-out, annotated test set? Also, the question of when, and how, the active learning process constituting the core of the bootstrapping phase should be terminated is closely related

to the issue of monitoring the learning progress. Hence, this issue also covers the questions: When should the active selection of documents terminate? Is it possible to define such a stopping criterion beforehand, or should the learning process be terminated by means of the user's good judgement?

E–5 Revision of system-suggested annotations There are, as previously mentioned, two strands to the question of the soundness of pre-tagging with revision. The typical objection to such an approach raises the potential bias introduced by the automatically inserted annotations as a primary reason that it should be avoided. The praise for pre-tagging with revision, on the other hand, is often derivable from the fact, or assumption, that the approach can be used to speed the annotation process up, albeit possibly with repercussions on annotator workload.

There are primarily two questions concerning pre-tagging with revision pertaining to the BootMark method. What are the requirements on the classifier used as pre-tagger in terms of, for instance, accuracy, to be able to be fruitfully used as a pre-tagging device? Is pre-tagging with revision applicable during the bootstrapping taking place in phase two, or will the performance of the classifier be so bad that it hinders the user, rather than helps him?

6.7 Relation to the work by others

After having introduced and explained the BootMark method in the previous sections, the time is ripe to relate it to the work presented by others. The BootMark method is novel and unique in the way active learning is used for bootstrapping. More specifically, the difference between the abstraction level of the container (document) and the markable (name) is what makes BootMark special.

According to the taxonomy introduced in chapter 5, the BootMark method would fall under the categories *active learning as inter-container support* (section 5.3), and *dynamic intra-container support* (section 5.2). It thus makes sense to compare BootMark to the efforts presented in each of those sections.

The approaches described as being active learning for inter-container support in section 5.3 all differ from BootMark in one important respect; that of the granularity of the markable versus that of the container. Typically, when active learning is involved, the markables (chunks of data selected) is at approximately the same level of linguistic abstraction as the containers (chunks

which the classifier finds informative). This makes sense since the task, in such situations, is most likely to train a classifier exhibiting the best performance attainable with the least human annotation effort possible. For instance, when using active learning for text categorization, the document level is often the one targeted when selecting examples, which is also the level of labeling; hence, there is no discrepancy between selection level and operation level see, for instance, the work by Lewis and Gale (1994), and Hoi, Jin and Lyu (2006).

In active learning for other types of linguistic information, it may not be feasible to assume a one-to-one correspondence between the linguistic level of the target concept and the abstraction level of the markables selected to realize the target concept. For example, when learning to classify part-of-speech information, it will be very hard, and in some cases impossible, for the human annotator to accurately decide on a proper part-of-speech tag for a word if that word is selected and shown out of context. In such cases, the selected chunks of data are usually comprised by sentences, as described by, for example, Ringger et al. (2007).

The inter-container support presented in section 5.3 all utilizes active learning in order to select sentences to mark-up with, for example, parse trees (Hwa 2000; Baldridge and Osborne 2004), word senses (Chklovski and Mihalcea 2002), part-of-speech information (Ringger et al. 2007), and named entities (Vlachos 2006; Tomanek, Wermter and Hahn 2007b, a).

In the category dynamic intra-container support, the BootMark method has a lot in common with the other approaches described as such in section 5.2; all benefits and drawbacks of any other attempt at realizing pre-tagging with revision are applicable to the BootMark method.

Perhaps most notable is BootMark's resemblance to Melita (Ciravegna, Petrelli and Wilks 2002; Ciravegna et al. 2002). In one of its phases, Melita employs a named entity recognizer which is dynamically trained on a per-document basis using the annotations made by the human annotator, to propose new annotations to be inserted by the user. The purpose of Melita, at that stage, is to turn the annotation process into the revising of proposed annotations. The significant difference between Melita and BootMark is that the former is not concerned with how the documents to annotate are selected; it is implied that Melita is not at all involved in selecting documents to annotate, instead the decision is made by the human annotator. Thus, Melita does not use an active learning strategy.

Part III

Empirically testing the BootMark method

7 EXPERIMENT DESIDERATA

To be able to judge the plausibility of the BootMark method introduced in chapter 6, the emerging issues outlined in section 6.6 have to be investigated and elaborated on in detail with respect to a specific task. The purpose of this chapter is to provide a rationale for such a task in order to allow for empirically testing the emerging issues.

Recall that the overall motivation for the work presented in this dissertation is to facilitate the creation of annotated corpora intended to be used for creating and adapting information extraction systems to meet new information needs. Consequently, a task suitable as context for investigating the emerging issues should be firmly rooted in the information extraction domain. Named entity recognition is a task fundamental to information extraction and is the task selected to serve as the backdrop of inquires relating to the five emerging issues listed in section 6.6. Here, named entity recognition is taken to include both identification and categorization of named entities.

7.1 The data

The data at hand is the training portion of the named entity part of the MUC-7 corpus regarding air crashes (Linguistic Data Consortium 2001).[6] The part of the MUC-7 corpus used contains 100 documents, 3480 sentences, and 90 790 tokens. The number of sentences and tokens was calculated after the corpus had been processed with the functional dependency grammar introduced in section 7.2.1; the grammar affect the number of tokens in that some multi-word units are considered one token, an example of which is *en route* in figure 7.3.

There are 6336 names, of which 4958 are of type ENAMEX, 1238 of type TIMEX, and 140 of type NUMEX. A further breakdown of the corpus into numbers is provided in table 7.1. As can be seen from the figures in the table, the sizes of the documents in the corpus vary quite a bit, both in terms of

[6] The training part of the MUC-7 corpus is henceforth referred to as *the MUC-7 corpus*.

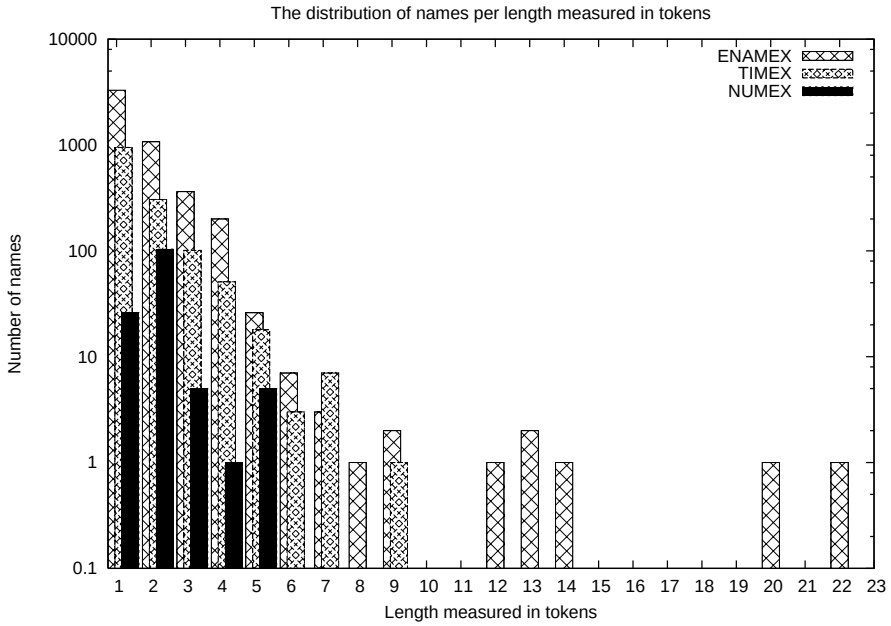

Figure 7.1: The distribution of name types per length in tokens.

		TOKENS	ENAMEX	TIMEX	NUMEX	NAMES	TYPES	SENT
SENT	Min	1.00	0.00	0.00	0.00	0.00	0.00	—
	Max	116.00	17.00	7.00	3.00	21.00	3.00	—
	Avg	25.91	1.42	0.36	0.04	1.82	0.90	—
	Sd	14.75	1.84	0.73	0.24	2.30	0.75	—
DOC	Min	280.00	8.00	2.00	0.00	10.00	2.00	9.00
	Max	3240.00	165.00	49.00	23.00	217.00	3.00	147.00
	Avg	907.90	49.58	12.38	1.40	63.36	2.41	34.80
	Sd	423.12	25.78	7.15	3.14	31.40	0.49	19.22

Table 7.1: Statistics about the MUC-7 corpus in terms of tokens, sentences, documents, and names. For sentences (SENT) and documents (DOC), the minimum (Min), maximum (Max), average (Avg), and standard deviation (Sd) are shown for tokens, names, and name types.

number of tokens and sentences; 907.90 tokens per document on average with a standard deviation of 423.12, and 34.80 sentences per document on average with 19.22 in standard deviation. Table 7.1 further shows that there are names in every document, but not in all sentences – the average number of names in a document is 63.36, while the average per sentence is 1.82 – and that each document contains at least two name types. ENAMEX is the largest groups

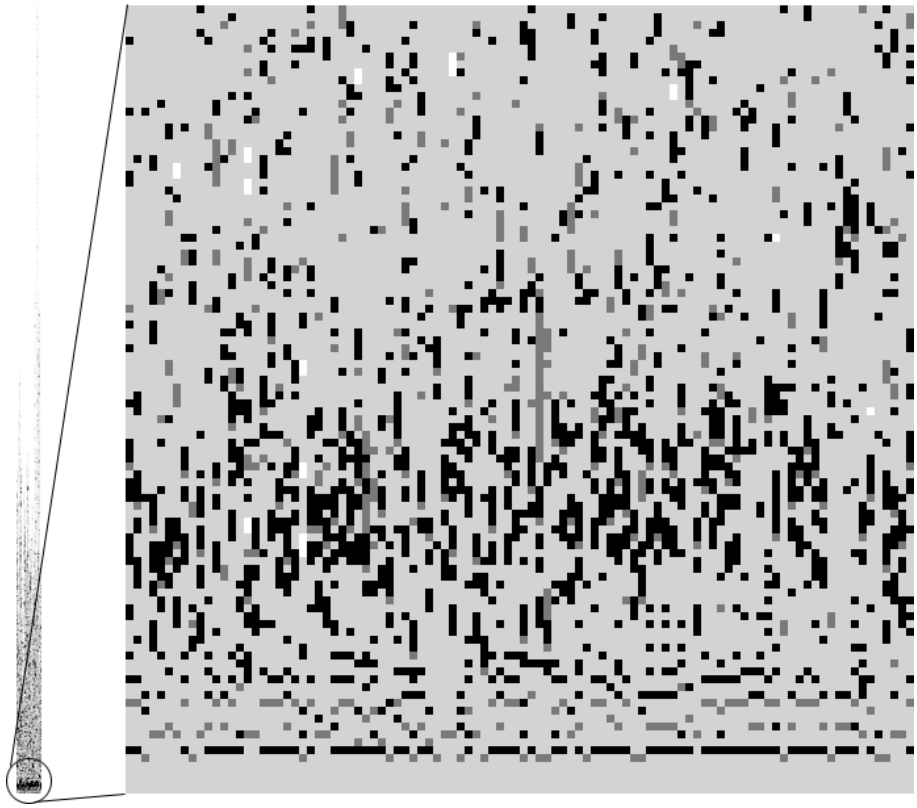

Figure 7.2: An illustration of the distribution of names in MUC-7. Columns denotes documents, rows denotes tokens. Black squares represent tokens of type ENAMEX, white represents NUMEX, while dark grey denotes TIMEX. Medium grey represents tokens that are not part of names. Note the long vertical line of dark grey squares approximately in the middle of the right hand side part of the figure. It is the 22 token long TIMEX expression, with an embedded ENAMEX, listed as the longest name in the corpus in figure 7.1: " Two harrowing hours after its crew lost much of its ability to navigate while at 35,000 feet over the North Atlantic,".

of names with, on average, 1.42 names per sentence, and 49.58 names per document. Only the NUMEX name type is not present in all documents.

The length of the names of each type is available in figure 7.1. The TIMEX type of name is distributed over the largest length span (1 - 22 tokens in length) follwed by ENAMEX (1 - 9 tokens), and NUMEX (1 - 5 tokens). The average length of a TIMEX name is 1.61 tokens (with a standard deviation of 1.35), while the average ENAMEX is made up from 1.52 tokens (standard deviation 0.89), and NUMEX is 1.95 tokens (standard deviation 0.71).

Figure 7.2 illustrates the distribution of names in the MUC-7 corpus. The smaller part on the left represents the whole corpus, while the larger part on the right is an enlargement of the circled portion on the left hand side. Each column – one square wide – denotes one document. The larger part of the image is 100 squares wide, which means that all documents in the corpus are represented. Each row – one square high – represents tokens. The larger part of the image is 100 squares high and thus represents the first 100 tokens in each document. A black square means that the token in the corresponding spot in the document is of type ENAMEX. White means NUMEX, while dark grey means TIMEX. The dominating medium grey color represents tokens that are neither type of names. As illustrated by figure 7.2, names tend to occur more frequently at the beginning of documents, while being decreasingly common as each document proceeds.

7.2 Technical set-up

When setting out to conduct experiments concerning named entity tagging, tools for linguistically analyzing, programatically handling, and automatically learning from textual documents are obviously needed. This section describes the tools used throughout the remainder of the dissertation.

7.2.1 The Functional Dependency Grammar

The English Functional Dependeny Grammar (EN-FDG, version 3.6) from Connexor Oy is used for linguistic analysis (Tapanainen and Järvinen 1997). The EN-FDG is a commercially available parser which carries out tokenization, part-of-speech tagging, lemmatization, the assignment of grammatical functions, as well as dependency parsing. The strengths of the EN-FDG include the ability to always deliver at least one (albeit possibly partial) analysis of the input.

Figure 7.3 contains the same example sentence as presented in figure 2.1, only this time the sentence has been processed by EN-FDG. The input to EN-FDG is plain text, possibly with very restricted XML tags present. All XML tags are passed through unaffected, unless they contain white space characters. As this is the case with any sensible mark-up, additional pre-processing measures are taken to prevent the EN-FDG to erroneously process XML tags. XML tags in the input text are percent-encoded[7] before the text is fed to the

[7] Percent-encoding (URL-encoding) is described in *Uniform Resource Identifier (URI): Generic Syntax* available at <http://tools.ietf.org/html/rfc3986>.

```
<ENAMEX TYPE="ORGANIZATION">
1    <s>    <s>
1    Massport    massport    attr:>2   @A> %>N <?> N NOM SG
</ENAMEX>
2    officials   official    subj:>3   @SUBJ %NH N NOM PL
3    said        say         main:>0   @+FMAINV %VA V PAST
4    the         the         det:>7    @DN> %>N DET SG/PL
5    replacement replacement attr:>6   @A> %>N N NOM SG
<ENAMEX TYPE="ORGANIZATION">
6    Martinair   martinair   attr:>7   @A> %>N N NOM SG
</ENAMEX>
7    jet         jet         subj:>8   @SUBJ %NH N NOM SG
8    was         be          obj:>3    @+FMAINV %VA V PAST SG1,3
9    en route    en route    >8        @ADVL %EH ADV
10   from        from        sou:>8    @ADVL %EH PREP
<ENAMEX TYPE="LOCATION">
11    Europe     europe      pcomp:>10 @<P %NH N NOM SG
</ENAMEX>
12    to         to          pth:>10   @ADVL %EH PREP
<ENAMEX TYPE="LOCATION">
13    New Jersey new jersey  pcomp:>12 @<P %NH N NOM SG
</ENAMEX>
14    ,          ,
15    but        but         cc:>8     @CC %CC CC
16    was        be          cc:>8     @+FMAINV %VA V PAST SG1,3
17    diverted   divert                @-FMAINV %VP EN
18    to         to          ha:>17    @ADVL %EH PREP
<ENAMEX TYPE="LOCATION">
19    Logan      logan       attr:>20  @A> %>N N NOM SG
</ENAMEX>
<TIMEX TYPE="DATE">
20    Tuesday    tuesday     pcomp:>18 @<P %NH N NOM SG
</TIMEX>
21    <p>    <p>
<TIMEX TYPE="TIME">
1    <s>    <s>
1    afternoon   afternoon   main:>0   @ADVL %EH N NOM SG
</TIMEX>
2    .    .
```

Figure 7.3: The example sentence from Figure 2.1 processed with the English Functional Dependency Grammar, FDG, from Connexor.

EN-FDG. For brevity, the XML tags in figure 7.3 have been decoded. The output of EN-FDG is in plain text, as illustrated by figure 7.3.

7.2.2 Kaba

To facilitate the programmatic handling of linguistically analyzed textual documents – for instance documents processed with the abovementioned EN-FDG – software has previously been developed in-house at SICS, the Swedish Institute of Computer Science AB (Olsson 2002). The software package, called Kaba, is a partial Java implementation of the TIPSTER document management architecture (Grishman et al. 1997). During the course of the experiments described in chapters 8 to 12, Kaba is further developed and adjusted to fit the needs of the particular experimental set-up used. In effect, Kaba acts as glue, integrating the document handling with the capabilities of the implementations of machine learners available in Weka (described below).

7.2.3 Weka

The machine learning toolbox of choice throughout the experiments is Weka (Witten and Frank 2005), which is a freely available, and large, collection of machine learning algorithms implemented in Java.[8] The algorithms can be utilized via a provided application programmer's interface (API) or via Weka's own graphical user interface. Weka is primarily selected due to the vast number of machine learning schemes available, as well as the swift support implied by Weka's large user community.

For the first set of experiments, concerning base learner characteristics described in chapter 8, the possibility of using the Weka Experimenter is particularly useful. The Weka Experimenter allows for defining machine learning experiments on one machine, and then run these experiments from the command line on other more powerful servers. The Experimenter allows for running sets of machine learning algorithms with different settings on various sets of training and test data, and then analyze the results using a number of different metrics.

For the experiments described in the remainder of the dissertation, the API is the primary way of using Weka; the object oriented nature of the API allows easy extension of existing base learners to accommodate for the special needs encountered in experimenting with active learning.

[8] Weka is available on the Internet at <http://www.cs.waikato.ac.nz/ml/weka/>.

```
@relation  illustrative-example

@attribute isAlphaNumeric     NUMERIC
@attribute isDigitsAndAlpha   NUMERIC
@attribute isDigitsAndPeriod  NUMERIC
@attribute isFirstInSentence  NUMERIC
@attribute isInitCaps         NUMERIC
@attribute containsSpace      NUMERIC
@attribute length             NUMERIC
@attribute targetClass  { OUT, LOCATION, ORGANIZATION, TIME, DATE }

@data

1 0 0 1 1 0 8  ORGANIZATION
1 0 0 0 0 0 9  OUT
1 0 0 0 0 0 4  OUT
1 0 0 0 0 0 3  OUT
1 0 0 0 0 0 11 OUT
1 0 0 0 1 0 9  ORGANIZATION
1 0 0 0 0 0 3  OUT
1 0 0 0 0 0 3  OUT
1 0 0 0 0 0 3  OUT
1 0 0 0 0 1 8  OUT
1 0 0 0 0 0 4  OUT
1 0 0 0 1 0 6  LOCATION
1 0 0 0 0 0 2  OUT
1 0 0 0 0 1 10 LOCATION
1 0 0 0 0 0 1  OUT
1 0 0 0 0 0 3  OUT
1 0 0 0 0 0 3  OUT
1 0 0 0 0 0 8  OUT
1 0 0 0 0 0 2  OUT
1 0 0 0 1 0 5  LOCATION
1 0 0 0 1 0 7  DATE
1 0 0 0 0 0 9  TIME
1 0 0 0 0 0 1  OUT
```

Figure 7.4: The example sentence from Figures 2.1 and 7.3 represented using the Attribute Relation File Format, ARFF.

7.2.4 ARFF

Weka requires the data it operates on to be in what is known as the Attribute Relation File Format, or ARFF for short (Witten and Frank 2005). An ARFF file consists of two parts; a header describing the attributes by which each instance is represented, and a data section where all instances are represented as feature vectors. Each feature can be one of the data types *numeric*, *nominal*, *string*, or *date*.

One of the steps taken to turn the textual data into ARFF is to decide on which features should represent the instances that the machine learning algorithms in Weka are to operate on (see section 8.2 for an elaboration on the features chosen).

Figure 7.4 shows the sentence from figure 2.1 represented using ARFF. The attributes are selected for illustrative purposes only and they will not make for a good distinction between the target classes. There are eight attributes declared in the `@relation` part. The first seven attributes are numeric, while the last one is nominal, reflecting the classes into which each instance will be classified. In the example, there are five possible classes. The `@data` section holds the actual representations of the instances. Each token in the sentence in figure 2.1, when processed with the EN-FDG as shown in figure 7.3, is represented by one instance in figure 7.4. The `targetClass` attribute is only used by the machine learners when training on the data; an instance to be classified is represented by all but the target attribute.

The ARFF does not impose an order among the instances. If one wishes to extend the example in figure 7.4 to incorporate context information for each token, that information has to be represented using additional attributes by adding, for instance, `classOfPrevious` to the `@relation` section and extend each instance accordingly. However, what is important is the order of the attributes, as they are treated by position internally in Weka, rather than by name; switching the place of two attributes may result in erroneous classifications, or inseparable training examples. Weka provides a range of Java classes to handle ARFF.

7.3 The BootMark prerequisites re-visited

The prerequisites made explicit prior to embarking on the description of the BootMark method in section 6.2 are met in the following ways by means of the present chapter:

The corpus is relevant. Given the task of named entity recognition (chapter 2), the training part of the MUC-7 named entity recognition sub-task is highly relevant, well-researched, and well-documented (section 7.1).

The availability of pre-processing tools. The Functional Dependency Grammar by Connexor will be used for linguistic analysis of the MUC-7 corpus (section 7.2.1). The WEKA machine learning libraries, and the Kaba platform will form the basis for the machine learning required to solve the task (sections 7.2.3 and 7.2.2).

Annotation guidelines. The training part of the MUC-7 data is annotated with named entity information, which is a prerequisite for the evaluation of the experiments to come. The guidelines are accompanied by definitions of appropriate performance metrics (elaborated on in section 8.7.1).

The scene is now set for empirically investigating the plausibility of the BootMark method as described in chapter 6 by means of addressing the emerging issues listed in section 6.6.

8 INVESTIGATING BASE LEARNERS FOR NAMED ENTITY RECOGNITION

This chapter is devoted to investigating the subject matter of the first emerging issue raised in chapter 6, section 6.6 concerning base learner characteristics and named entity recognition. The task of named entity recognition can be formulated as follows:

Definition Let D be a document, and L be the set of classes of names to recognize, then the goal of named entity recognition is to learn a classifier C such that $C : n \rightarrow l$ for all names $n \in D$, where $l \in L$.

Given this definition of the task, the following questions are examined:

- What information should be included in the representation of a name?
- What candidate base learners are there?
- What are good parameter settings for those base learners with respect to the data available?
- What are the characteristics of the base learners with respect to named entity recognition in terms of time required to train, time required to apply a classifier, and classification accuracy?

The answers to the above questions are used in chapters 9 to 12 to form the base for the design and implementation of the experiments addressing the remaining emerging issues. When the combinations of representational schemes and base learners have been evaluated with respect to the named entity recognition task, it will be possible to select the best combination to be used as a baseline.

8.1 Re-casting the learning problem

Names are often more than one token in length. For instance, the names in the ENAMEX class in the part of the corpus used in this experiment are on average

1.52 tokens long (section 7.1). This is a fact which complicates the named entity recognition task as it is defined on the previous page since it does not leave any clues as to where the names in the text should be sought for: Should each possible sub-string in the document be considered a potential name, or should perhaps each n-gram combination of tokens in the text be considered a name candidate? A solution is to re-define named entity recognition as a token classification task which purports to deciding whether each individual token in a text is a part of a name, instead of trying to recognize whole names all at once. Following the IOB labelling scheme proposed by Ramshaw and Marcus (1995), the initial named entity recognition task of assigning one of the seven classes of names to a chunk of text, can be turned into a 15-class task in which each token in the text is assigned a tag indicating if it is located in (I), outside (O), or at the beginning (B) of a name. Thus, the task of named entity recognition is that of assigning, to each token in the input, one of the 15 labels in table 8.1. The initial definition of the named entity recognition task is re-casted as follows.

Definition Let D be a document, and let L be the set of classes of names in table 8.1, then the goal of named entity recognition is to learn a classifier C such that $C : t \rightarrow l$ for all tokens $t \in D$, where $l \in L$.

B-PERSON	I-PERSON
B-ORGANIZATION	I-ORGANIZATION
B-LOCATION	I-LOCATION
B-MONEY	I-MONEY
B-PERCENT	I-PERCENT
B-TIME	I-TIME
B-DATE	I-DATE
OUT	

Table 8.1: Target classes in named entity recognition.

B-X is assigned to a token if it is the first token in a sequence of type X and if the immediately preceding token is the last one in a sequence of the same type. I-X is assigned to tokens otherwise appearing in a sequence of type X. OUT is assigned to tokens that are not part of a name.

8.2 Instance representation

Now that the task has been broken down to a finer grained and more manageable one, it is time to focus on how each token should best be represented to facilitate learning. In machine learning, the examples from which to learn, and

Cardinal/ordinal tag[1]	Is first name?[3]
Case tag[1]	Is four digit number?[2]
Class of previous previous[4]	Is hyphen?[2]
Class of previous[4]	Is initial caps?[2]
Comp tag[1]	Is location?[3]
Contains digits and dollar?[1]	Is name part?[3]
Contains dollar?[2]	Is roman number?[2]
Contains percent?[2]	Is salutation?[3]
Contains punctuation?[2]	Is sentence delimiter?[1]
Contains white space?[2]	Is single cap and period?[2]
Dependency function tag[1]	Is single character?[2]
Grammatical function tag[1]	Is single character and period?[2]
Is all caps?[2]	Is single lower case character and period?[2]
Is all digits?[2]	Is two digit number?[2]
Is all lower case?[2]	Morph tag[1]
Is alpha numeric?[2]	Num tag[1]
Is any or all digits?[2]	Part of speech tag[1]
Is company descriptor?[3]	Prefix[2]
Is digits and alpha?[2]	Suffix[2]
Is digits and comma?[2]	Surface form[1]
Is digits and dash?[2]	Surface lemma form[1]
Is digits and period?[2]	Syntactic tag[1]
Is digits and slash?[2]	Tense tag[1]
Is first in sentence?[1]	Length in characters[2]
	TARGET CLASS

Table 8.2: The superscript next to the feature names denote the origin of the feature. Features marked: (1) are calculated based on the linguistic pre-processing of the input made with the EN-FDG; (2) originate from the surface appearance of the text; (3) are fetched from pre-compiled lists of information; and (4) depend on predictions made concerning the context.

consequently also the unseen instances to classify, are commonly represented as vectors of features. Each feature corresponds to a piece of information that is believed to contribute to making possible the distinction between feature vectors belonging to different classes.

The task of named entity recognition has been thoroughly investigated, for example in MUC. Some of the important features used to represent names are described by, for instance, Bikel, Schwartz and Weischedel (1999), Borthwick et al. (1998), and Nadeau and Sekine (2007). The features used in this experiment are such features that have contributed to good results as reported by others, as well as a range of other features that are due to the use of the EN-FDG (section 7.2.1). The full list of features used to represent a single token is available in table 8.2. All features except for the target class are represented numerically. The target class is a nominal feature that can take any of

the values listed in table 8.1. The features can be categorized according to their origin. Some of the features are made possible by linguistically pre-processing the input text, others are drawn from the surface appearance of the words in the text, while yet others have their source in pre-compiled lists, or in the predicted class of a token's surroundings. Each feature, except for the target class, in table 8.2 is marked as belonging to one of the four feature categories. The pre-compiled lists used in this experiment stem from a wide variety of projects conducted during the course of several years at SICS; the lists have not been developed nor examined with respect to their coverage of the data used in the present experiment.

Each token can be represented in terms of its context. For instance, a context window size of 0 means that a token is represented by means of itself only. A context size of 1 means that a token is represented by means of itself *and* the token immediately to the left, and the token immediately to the right. In the current experiment, data sets corresponding to context window sizes of *intrinsic*, 0 to 5 (inclusive) are used. In the intrinsic representation of instances, a token is represented by the features outlined in table 8.2, except for the features *class of previous previous*, *class of previous*, and *is first in sentence*. Arguably, there are more features than the aforementioned three that can be considered extrinsic; the excluded features are such that clearly require information beyond that carried by the single token to compute. The reason to include the intrinsic token representation in the experiment is to examine how well the machine learning schemes used can classify a token by means of information stemming from mainly the token itself.

The values of the features listed in table 8.2 are calculated for each token within the context window. For instance, if a context size of two tokens on the left and two tokens on the right of the current main token is used, the number of features used for representing the main token is

$$2 \times (N-1) + N + 2 \times (N-3) = 237$$

where N is the number of features presented in table 8.3. The target class feature is only calculated for the main token, and the predicted class of a previous token is not available for the right hand context. As shown in table 8.3, the number of features used to represent a single token increases considerably as a wider context is used.

8.3 Automatic feature selection methods

The sheer amount of features used to represent each token lead to the question whether all those features really help in telling one class from another. There

CONTEXT	# FEATURES
intrinsic	46
0	49
1	143
2	237
3	331
4	425
5	519

Table 8.3: The number of features used to represent a token in each context size.

are means by which a given feature set can be reduced – feature selection – while still maintaining the expressiveness of the retained representation. The reason for reducing the number of features is primarily to speed up the learning process, but also to reduce the size of the resulting classifier.

The feature selection problem can be defined as one in which one wishes to find a minimum set of M relevant features that describes the data set at hand equally well as the original N features do, and where $M < N$. Feature selection is usually divided into two approaches; one is about making decisions about which features to select based solely on the characteristics of the data at hand (so called filter methods or intrinsic methods), while the other also involves the learning method under investigation (wrapper methods or extrinsic methods). An overview of feature selection schemes is given by, for instance, Witten and Frank (2005).

Hall and Holmes (2003) investigate the performance of a number of automatic feature selection techniques on a range of data sets. Following the conclusions made by Hall and Holmes, two intrinsic feature selection methods are chosen for inclusion in the present experiment; consistency-based feature subset evaluation (Liu and Setiono 1996), and correlation-based feature selection (Hall 1999).

The key to consistency-based feature selection is an inconsistency criterion for specifying whether dimensionally reduced representations are acceptable. The criterion is used for iteratively comparing randomly selected feature subsets in a way such that the subset which is deemed most consistent with respect to a pre-defined rate is used as a replacement for the original feature set. Liu and Setiono (1996) claim their method to be fast and unaffected by any bias introduced by a learning algorithm.

The main idea in correlation-based feature selection is to select those features that are highly correlated with the target class, but not with each other. Hall (1999) empirically shows that the method, in many cases, improves classifier accuracy.

ALGORITHM FAMILY	WEKA IMPLEMENTATION
Decision tree learner	J48
	REPTree
Rule learner	JRip
	PART
Bayesian learner	NaiveBayes
	NaiveBayesUpdateable
Lazy learner	IBk
Function	Radial Basis Function
	Logistic

Table 8.4: The machine learning families and Weka implementations used in the experiment.

Consistency-based feature selection as well as correlation-based feature selection are applied to the data set in which instances are represented with a 5 token context window, that is, the widest context, containing 519 features to choose from.

8.4 Candidate machine learning schemes

Although Weka, which is the machine learning platform of choice in this experiment as explained in section 7.2.3, contains numerous machine learning schemes, it does not comprise all the ones reported as successful in the literature concerning named entity recognition. Most notably, Conditional Random Fields and Hidden Markov Models are not included in Weka, and they will consequently not be included among the candidate base learners. The machine learning methods under scrutiny are chosen since they represent a fairly broad spectrum of the algorithms used for named entity recognition (see chapter 2), but the selection is by no means claimed to be an exhaustive one. The selected algorithms stem from different families of learners, as outlined in table 8.4. Introductions to each of these learning methods are available in chapter 3.

Given the investigations to come concerning active learning and document selection, machine learning methods that accommodate rapid learning are desirable. This is the case since, as evident by the prototypical active learning algorithms introduced in chapter 4 (figures 4.1, 4.2, and 4.3, respectively), active learning regularly enforces re-training of classifiers as more annotated data is made available. Intuitively, what is needed are methods for learning that allow for adding new training data as it appears, that is, methods for incremental machine learning. Two such methods are explored; a method for incremental Bayesian learning called NaiveBayesUpdateable (John and Langley 1995),

and a Nearest Neigbor learner, IBk (Aha, Kibler and Albert 1991). In addition, a naïve Bayesian learner – NaiveBayes – is included in the experiment to serve as reference by which the influence of the incremental learning performed by the NaiveBayesUpdateable is judged.

The family of decision tree learners is represented by J48, which is an implementation of Quinlan's C4.5 (Quinlan 1993), and REPTree, which is a comparatively fast but memory consuming learner.

For learning rules, JRip, and PART are used. The former is a propositional rule learner – an implementation of RIPPER (Cohen 1995) – while the latter builds partial C4.5 decision trees in each iteration and makes the "best" leaf a rule (Frank and Witten 1998).

There evidently is a very large range of Artificial Neural Network configurations to test. The one included here, RBFNetwork is an implementation of a Radial Basis Function (Powell 1987). RBFNetwork is a two-layer feedforward network that differs from a multilayer perceptron in the way that the hidden units perform computations. The RBFNetwork was chosen since it learned faster than the multilayer perceptron network on a subset of the training data. The other function family member included as a candidate base learner is Logistic, which is an implementation of Multinomial Logistic Regression with a ridge estimator (le Cessie and van Houwelingen 1992). Logistic regression is also known as Maximum Entropy classification.

8.5 Parameter settings

Each machine learning method has its own parameters that need to be tweaked to obtain the best result with respect to the task and data at hand. The importance of paying attention to parameter settings is emphasized by Daelemans and Hoste (2002), who raise general concerns regarding the reliability of reported machine learning results. Their hypothesis is that the differences in accuracy that can be observed between different machine learning methods applied to some problem is lower than the variability in accuracy originating from interactions between data selection, data representation, and algorithm parameter settings within one and the same machine learning algorithm. To test their hypothesis, Dealemans and Hoste employ memory-based learning and decision tree learning for three tasks – word sense disambiguation, diminutive suffixes, and part-of-speech tagging – and analyze the influence of algorithm parameter optimization, as well as the interaction between feature selection and parameter optimization. Based on the experiments, Daelemans and Hoste (2002) claim to confirm their hypothesis that the accuracy differences between different base learners will, in general, be lower than the accuracy range re-

sulting from interactions between parameter settings and selection information sources as used by a single base learner.

Although the results reported by Daelemans and Hoste (2002) are on tasks other than named entity recognition, there is no reason not to believe that the interaction of parameter settings, data representation and base learners is as important to the present experiment. Thus, an informed variation of algorithm parameters is conducted for each base learner. The parameter settings for the respective learning scheme are first tested using a subset of the entire training data. The influence of changes to parameter values on the classifier results is quicker to assess on small data sets, albeit it is admittedly also more prone to issues such as overfitting of the learned model with respect to the data at hand. Once the identification of parameters and the approximate parameter value range are decided, a classifier set-up for each such combination is defined in conjunction with the full data sets. Appendix A, section A.1 contains listings, per base learner, of all parameters and parameter values used in this experiment. Note that search conducted in parameter space is by no means exhaustive, but should rather be considered explorative.

8.6 Token classification results

All nine machine learners in the right column in table 8.4 were run on each one of the seven data sets representing the context sizes introduced in section 8.2, using various parameter settings specific to the individual learning methods as described in section 8.5. In addition, the two feature selection techniques introduced in section 8.3 were used with each machine learner on the data set representing the largest context size. In total 216 experiments were run, using 10-fold cross-validation for calculating the results.

Note that the results reported in this section are for the classification of single tokens as defined in section 8.1, not for recognizing entire multi-token names. The reason why the token classification task is at all conducted, instead of directly investigating the multi-token named entity recognition task, has to do with the way the task is re-casted from a 7-class task, to a 15-class task as described in section 8.2. Weka supports the execution and evaluation of classification of single instances. Due to the re-casting of the initial problem, from multi-token names, into single-token name parts, each token, not each name, is considered a separate instance in Weka. Thus, the re-casted problem is evaluated in terms of tokens, and it is assumed that the base learner configuration that performs the best on the single-token classification task, is also the one yielding the best results in the multi-token named entity recognition task. The latter is reported and elaborated on in section 8.7.

ABBREVIATION	EXPLANATION
Ctx	The context in which each token in the input is represented. Possible values are *intr* for *intrinsic*, and *c0 - c5* where the number indicates the size of the context, and *c5-fs* which denotes a context of size 5 which has been reduced by means of one of the automatic feature selection methods.
Train	Normalized CPU time in seconds for training a classifier.
Test	Normalized CPU time in seconds for testing a classifier.
PC	The percent correctly classified tokens.
Prec	The precision, as defined in Equation 2.
Rec	The recall, as defined in Equation 3.
F	The F-score, as defined in Equation 4.
NBUd	NaiveBayesUpdateable
NB	NaiveBayes
REPT	REPTree
CfsSE	Correlation-based feature-subset selection.
ConsSE	Consistency-based feature-subset selection.

Table 8.5: Legend to reading the results presented in tables 8.6, 8.7, 8.8, and 8.9.

The result tables included in this part, tables 8.6, 8.7, 8.8, and 8.9, contain only the classifiers learned by the best combination of base learner, parameter settings, and data representation for each of the learners in the right column in table 8.4.[9] That is, the results are only reported for one instance of, for example, the IBk nearest neighbor learner, although several base learner configurations were explored. The base learner configurations reported in each table are not necessarily the same. For instance, the IBk nearest neighbor reported as the fastest IBk to learn, is not learned using the same learner configuration as the one that is fastest to apply, or the one that produces the most accurate classifications. A combination of the results in terms of learning time, time required to test, and accuracy is available in table 8.9, which the most important table reporting the results of the token classification task. Recall, precision, and F-score are also included in the result listings, although it was found that the obtained results varied too little to be of any practical use; instead, the percentage of correctly classified tokens is compared between each classifier. Table 8.5 serves as a legend to reading the parts of the result tables that have been abbreviated.

[9]The combination of base learner, parameter settings, and data representation is referred to as *base learner configuration* or *learner configuration*.

8.6.1 A note on measuring time across machines

The results of the single-token classification task reported in sections 8.6.2, 8.6.3, 8.6.4, and 8.6.5 include the CPU time required to train and test each base learner configuration. The experiments were run on several different computers. Generally, it is not possible to accurately and with certainty measure exact execution times across different computers. What is important in the results reported is not the exact time it took to train, or test, a particular learner, but rather the relative rank order implied by the training and testing time. To obtain this relative order, the measured times were normalized with respect to each computer used. The factor for normalizing the times was obtained by running the same experiment on all machines, and then compare the resulting times obtained as part of each result. That way, each computer could be assigned a time normalization factor, which was subsequently used to re-calculate the training and testing times obtained in all experiments, on all machines. The times reported in the results are normalized.

8.6.2 Time to train

This section focuses on the time required to train each base learner configuration. As anticipated, the list of machine learning methods presented in table 8.6 requiring the least time to train on the training part of the MUC-7 data has the incremental learning methods in the top positions. The IBk nearest neighbor classifier is by far the fastest to learn, which is due to the fact that it does not really learn, but merely adds data as it goes. NaiveBayesUpdateable also hoards data as it becomes available, but it still is slower than the nearest neighbor approach. Incremental learning, or lazy learning, are fast to learn, but requires more time to classify unseen data, as will be evident in section 8.6.3.

Interestingly enough, the NaiveBayes learner is not all that much slower than the incremental Naïve Bayesian learner, a fact which might invalidate the inclusion of the latter, depending on its performance along the other two axes; time required to apply to unseen data, and accuracy.

In fourth place, the REPTree decision tree learner performs closer to the top three learners, than it does the Logistic and J48 learners, which are in fifth and sixth place, respectively. The REPTree learner requires approximately double the amount of time required by the Naïve Bayesian learners, but only one tenth of the time required by the J48 decision tree learner. Then follows a range of learning methods that have been invoked after a correlation-based feature selection method has been applied; what is worth noticing here is that some of the machine learning methods are faster learners if the feature space has been

CLASSIFIER	CTX	TRAIN	TEST	PC	PREC	REC	F
IBk	intr	0.13	709.80	96.49	0.98	0.99	0.99
NBUd	intr	9.30	5.28	69.66	0.99	0.73	0.84
NB	intr	11.75	18.80	91.18	0.98	0.95	0.97
REPT	intr	20.01	0.02	96.15	0.98	0.99	0.99
Logistic	intr	145.69	0.38	92.60	0.97	0.98	0.97
J48	intr	180.85	0.08	96.72	0.98	0.99	0.99
CfsSE IBk	c5-fs	284.92	340.04	97.17	0.99	0.99	0.99
CfsSE NBUd	c5-fs	360.75	1.67	85.34	0.99	0.89	0.94
CfsSE J48	c5-fs	367.69	0.11	97.28	0.99	0.99	0.99
CfsSE Logistic	c5-fs	408.81	0.38	91.81	0.93	0.99	0.96
CfsSE NB	c5-fs	435.46	0.17	93.13	0.99	0.95	0.97
CfsSE REPT	c5-fs	446.37	0.12	97.32	0.99	0.99	0.99
CfsSE PART	c5-fs	504.79	0.23	97.19	0.99	0.99	0.99
RBF	c1	1040.80	6.88	90.56	0.94	0.98	0.96
PART	c0	1295.23	0.3	97.43	0.99	0.99	0.99
CfsSE Jrip	c5-fs	1858.39	0.19	96.33	0.97	0.99	0.98
Jrip	intr	4154.58	0.13	97.97	0.98	0.99	0.99
ConsSE IBk	c5-fs	5613.11	167.30	96.36	0.98	0.99	0.98
ConsSE J48	c5-fs	6040.74	0.11	96.67	0.98	0.99	0.98
ConsSE Logistic	c5-fs	6062.71	0.38	88.80	0.89	1.00	0.94
ConsSE NBUd	c5-fs	6178.32	0.37	86.21	0.91	0.96	0.93
ConsSE PART	c5-fs	6434.96	0.33	96.39	0.98	0.99	0.98
ConsSE NB	c5-fs	6654.77	0.15	95.47	0.96	0.99	0.98
CfsSE RBF	c5-fs	7499.59	1.41	91.79	0.92	1.00	0.96
ConsSE REPT	c5-fs	7913.13	0.12	96.82	0.98	0.99	0.99
ConsSE Jrip	c5-fs	9056.25	0.25	94.82	0.96	0.99	0.97
ConsSE RBF	c5-fs	18383.74	1.01	90.01	0.91	0.99	0.95

Table 8.6: Classifiers ordered according to the time required to train on data.

reduced prior to the learning phase. For instance, the PART method learns faster when correlation-based feature selection has been applied to a data representation containing information about five tokens to the left, and five tokens to the right, than it does when applied to a zero context. In this experiment, the correlation-based feature selection method outperforms consistency-based feature selection with respect to training time in all cases. However, neither feature selection method contributes to reducing the learning times in manners that would make the resulting representations alternatives to the intrinsic instance representation. Overall, and not surprisingly, the classifiers all learn faster on the data represented by fewer features. The intrinsic representations are preferred by the top six base learners, followed by seven base learner configurations utilizing reduced feature sets.

Refer to section A.2 in appendix A for a complete listing of the parameters used with each base learner listed in table 8.6.

CLASSIFIER	CTX	TRAIN	TEST	PC	PREC	REC	F
REPT	intr	44.96	0.02	96.78	0.98	0.99	0.99
J48	c0	184.87	0.05	97.44	0.99	0.99	0.99
ConsSE J48	c5-fs	6040.74	0.11	96.67	0.98	0.99	0.98
CfsSE J48	c5-fs	367.69	0.11	97.28	0.99	0.99	0.99
Jrip	intr	4763.83	0.12	97.02	0.98	0.99	0.99
CfsSE REPT	c5-fs	449.57	0.12	96.95	0.98	0.99	0.99
ConsSE REPT	c5-fs	7913.13	0.12	96.82	0.98	0.99	0.99
ConsSE NB	c5-fs	6654.77	0.15	95.47	0.96	0.99	0.98
CfsSE Jrip	c5-fs	1879.65	0.17	96.11	0.97	0.99	0.98
CfsSE NB	c5-fs	435.46	0.17	93.13	0.99	0.95	0.97
ConsSE Jrip	c5-fs	9117.73	0.22	94.69	0.96	0.99	0.97
CfsSE PART	c5-fs	504.79	0.23	97.19	0.99	0.99	0.99
ConsSE PART	c5-fs	7475.34	0.27	96.18	0.98	0.99	0.98
PART	c0	1295.23	0.30	97.43	0.99	0.99	0.99
Logistic	c0	282.21	0.37	93.80	0.97	0.98	0.98
ConsSE NBUd	c5-fs	6178.32	0.37	86.21	0.91	0.96	0.93
CfsSE Logistic	c5-fs	549.89	0.38	91.79	0.93	0.99	0.96
ConsSE Logistic	c5-fs	6062.71	0.38	88.80	0.89	1.00	0.94
NB	intr	20.99	0.61	92.15	0.99	0.94	0.96
ConsSE RBF	c5-fs	18383.74	1.01	90.01	0.91	0.99	0.95
CfsSE RBF	c5-fs	7499.59	1.41	91.79	0.92	1.00	0.96
CfsSE NBUd	c5-fs	360.75	1.67	85.34	0.99	0.89	0.94
RBF	intr	1249.56	2.81	91.30	0.96	0.98	0.97
NBUd	intr	9.30	5.28	69.66	0.99	0.73	0.84
CfsSE IBk	c5-fs	292.46	139.57	96.90	0.99	0.99	0.99
ConsSE IBk	c5-fs	5613.11	167.30	96.36	0.98	0.99	0.98
IBk	intr	1.99	263.18	96.41	0.98	0.99	0.99

Table 8.7: Classifiers ordered according to the time required to apply to data.

8.6.3 Time to test

In terms of time required to apply a classifier to the test portion of the data, table 8.7 reveals that the deviation in time between consecutive classifiers is, in the majority of the cases, not that big. The tail of the list, however, contains three base learner configurations that stand out in a bad way; compared to any of the other machine learning methods under scrutiny, the memory-based learner is a poor performer when it comes to time required to process the test data. The NaiveBayesUpdateable performs well, compared to the IBk. On the other hand, the incremental Naïve Bayesian learner is an order of magnitude slower than its non-incremental sibling, the NaiveBayes learner. In effect, this means that unless the NaiveBayesUpdateable outperforms the NaiveBayes learner significantly in the third way of comparison – classifier accuracy – NaiveBayesUpdateable is of no further interest to this experiment.

The decision tree learners are approximately equally fast, with the REPTree learner in the first position, slightly ahead of the J48. As in the comparison of learning times, the use of a smaller context makes for a faster learner. What is interesting is that the J48 learner is faster using the zero context, than the smaller intrinsic ditto.

Contrary to the case of training the classifiers (section 8.6.2), it is not at all clear which of the two feature selection methods performs the best; the correlation-based feature selection method is better in five of nine cases. But, again, none of the feature selection methods contribute to faster learning.

Refer to section A.3 in appendix A for a complete listing of the parameters used with each base learner listed in table 8.7.

8.6.4 Accuracy

Table 8.8 describes the accuracy of the classifiers. What is striking in the table is that, overall, classifiers perform better when trained on data represented using narrow contexts. The best performer is the Jrip rule learner when trained on intrinsic token representations, followed by the other rule learning scheme employed, PART. Base learner configurations involving the two decision tree learners occupy the third and fourth place.

Most of the classifiers manage to classify more than 95 percent of the tokens in the test portion of the data correctly. Four base learner configurations scored below 90 percent; the NaiveBayesUpdateable with and without automatic feature selection, and the Logistic Regression learner with consistency-based feature selection. The NaiveBayes learner performs considerably better, and the use of the incremental Bayesian learning is therefore off the chart. Although the other incremental learning scheme, IBk, does not perform badly compared to the other methods, it is too far behind to be a top candidate for further exploration (as will be evident in the combination of results in section 8.6.5).

The correlation-based feature selection method is better than the consistency-based feature selection method in seven of nine cases. As in the previous two comparisons in sections 8.6.2 and 8.6.3, the automatic means to reduce the widest context do not lend themselves to results in the top positions.

Refer to section A.4 in appendix A for a complete listing of the parameters used with each base learner listed in table 8.8.

CLASSIFIER	CTX	TRAIN	TEST	PC	PREC	REC	F
Jrip	intr	4154.58	0.13	97.97	0.98	0.99	0.99
PART	c1	12290.66	1.06	97.69	0.99	0.99	0.99
REPT	c0	122.72	0.11	97.68	0.99	0.99	0.99
J48	c0	197.35	0.07	97.68	0.99	0.99	0.99
CfsSE REPT	c5-fs	446.37	0.12	97.32	0.99	0.99	0.99
CfsSE J48	c5-fs	367.69	0.11	97.28	0.99	0.99	0.99
CfsSE PART	c5-fs	504.79	0.23	97.19	0.99	0.99	0.99
CfsSE IBk	c5-fs	284.92	340.04	97.17	0.99	0.99	0.99
IBk	c0	4.20	305.02	97.12	0.99	0.99	0.99
ConsSE REPT	c5-fs	7913.13	0.12	96.82	0.98	0.99	0.99
ConsSE J48	c5-fs	6540.04	0.12	96.69	0.98	0.99	0.98
ConsSE PART	c5-fs	6434.96	0.33	96.39	0.98	0.99	0.98
ConsSE IBk	c5-fs	5613.11	167.30	96.36	0.98	0.99	0.98
CfsSE Jrip	c5-fs	3018.82	0.20	96.34	0.98	0.99	0.98
Logistic	c2	1802.13	0.59	95.90	0.98	0.99	0.99
ConsSE NB	c5-fs	6654.77	0.15	95.47	0.96	0.99	0.98
ConsSE Jrip	c5-fs	9939.44	0.22	94.91	0.96	0.99	0.97
CfsSE NB	c5-fs	435.46	0.17	93.13	0.99	0.95	0.97
NB	c0	22.22	0.65	93.05	0.99	0.94	0.97
CfsSE Logistic	c5-fs	408.81	0.38	91.81	0.93	0.99	0.96
CfsSE RBF	c5-fs	7499.59	1.41	91.79	0.92	1.00	0.96
RBF	c0	1448.02	3.05	91.43	0.96	0.98	0.97
ConsSE RBF	c5-fs	18383.74	1.01	90.01	0.91	0.99	0.95
ConsSE Logistic	c5-fs	6062.71	0.38	88.80	0.89	1.00	0.94
ConsSE NBUd	c5-fs	6178.32	0.37	86.21	0.91	0.96	0.93
CfsSE NBUd	c5-fs	360.75	1.67	85.34	0.99	0.89	0.94
NBUd	c0	10.04	5.71	81.28	0.99	0.85	0.92

Table 8.8: Classifiers ordered according to number of precent correctly classified tokens in the input data.

8.6.5 Combining times and accuracy

The results from the three complete[10] listings of the training time, the testing time, and the accuracy were combined into one list using squared rank sum. The combined results are available in table 8.9. The squared rank sum for an item in the ranked result lists is calculated as $R_1^2 + R_2^2 + R_3^2$, where R_i is the rank of the item in list i.

The results in table 8.9 reveal that the task of as quickly and correct as possible decide whether a single token is part of name can successfully be approached using decision trees. The two decision tree learners used, REP-

[10]Each complete listing contains all 216 experiments conducted, ordered according to the respective field of comparison, not only the portions listed in the tables shown in this section.

CLASSIFIER	CTX	TRAIN	TEST	PC	PREC	REC	F
REPT	c0	122,31	0,10	97,68	0,99	0,99	0,99
J48	c0	184,87	0,05	97,44	0,99	0,99	0,99
CfsSE J48	c5-fs	367,69	0,11	97,28	0,99	0,99	0,99
CfsSE REPT	c5-fs	446,37	0,12	97,32	0,99	0,99	0,99
Jrip	intr	4154,58	0,13	97,97	0,98	0,99	0,99
CfsSE PART	c5-fs	504,79	0,23	97,19	0,99	0,99	0,99
PART	c0	1295,23	0,30	97,43	0,99	0,99	0,99
ConsSE J48	c5-fs	6040,74	0,11	96,67	0,98	0,99	0,98
CfsSE Jrip	c5-fs	1858,39	0,19	96,33	0,97	0,99	0,98
CfsSE NB	c5-fs	435,46	0,17	93,13	0,99	0,95	0,97
ConsSE REPT	c5-fs	7913,13	0,12	96,82	0,98	0,99	0,99
Logistic	c0	282,21	0,37	93,80	0,97	0,98	0,98
IBk	c0	4,20	305,02	97,12	0,99	0,99	0,99
CfsSE IBk	c5-fs	284,92	340,04	97,17	0,99	0,99	0,99
NB	c0	22,22	0,65	93,05	0,99	0,94	0,97
CfsSE Logistic	c5-fs	408,81	0,38	91,81	0,93	0,99	0,96
ConsSE PART	c5-fs	6434,96	0,33	96,39	0,98	0,99	0,98
ConsSE NB	c5-fs	6654,77	0,15	95,47	0,96	0,99	0,98
ConsSE Jrip	c5-fs	9117,73	0,22	94,69	0,96	0,99	0,97
NBUd	c0	10,04	5,71	81,28	0,99	0,85	0,92
CfsSE NBUd	c5-fs	360,75	1,67	85,34	0,99	0,89	0,94
RBF	intr	1249,56	2,81	91,30	0,96	0,98	0,97
ConsSE IBk	c5-fs	5613,11	167,3	96,36	0,98	0,99	0,98
ConsSE Logistic	c5-fs	6062,71	0,38	88,80	0,89	1,00	0,94
ConsSE NBUd	c5-fs	6178,32	0,37	86,21	0,91	0,96	0,93
CfsSE RBF	c5-fs	7499,59	1,41	91,79	0,92	1,00	0,96
ConsSE RBF	c5-fs	18383,74	1,01	90,01	0,91	0,99	0,95

Table 8.9: Classifiers ordered according to squared rank sum.

Tree and J48, occupy the four top-most positions in the results list. While the REPTree learner is slightly better than the J48 when it comes to classification accuracy, it is slightly worse in terms of time required to be applied to test data.

The combined characteristics of IBk, which is the fastest to learn (table 8.6), but require significantly more time to apply (table 8.7), is made clear by using the squared rank sum approach. The same goes for the application of the automatic feature selection methods. No learner configuration employing feature selection to reduce the dimensionality of the data representation made it high enough to be considered as an alternative worth further exploration.

Given the representational scheme, learning in small contexts apparently is sufficient to be able to quickly and accurately classify single tokens as belonging to one of the target classes. A complete listing of the parameters used for the base learners in table 8.9 are available in section A.5 in appendix A.

1. Let $t_i, t_{i+1}, \ldots, t_n$ be a sequence of tokens T.
2. Let L be a set of labels corresponding to the 15 target classes listed in Table 8.1.
3. Let C be a classifier such that $C: t \rightarrow l$, and $t \in T$, $l \in L$.
4. $\forall t \in T$, apply C and obtain a sequence S consisting of pairs $< t, l >$.
5. Assign label l to each longest sub-sequence $s \in S$ in which consecutive pairs, $< t_i, l_j > < t_{i+n}, l_k >$, share the same label $l = l_j = l_k$.

Figure 8.1: Using a single token classifier for recognizing and categorizing multi-token names.

8.7 Named entity recognition results

Following the problem formulation in section 8.1, the results reported in section 8.6 reflect how the base learners behave when faced with the task of classifying individual tokens. As explained earlier, names often consists of several tokens, and as interesting as they may be, the results reached in a single-token classification setting are not particularly useful in an active learning scenario. That is, unless they are transferred to a multi-token setting. The application of single-token classification to recognizing multi-token names is described in figure 8.1. Essentially, a classifier capable of predicting the class label of a given token is applied to all tokens in the text in which names are sought for. The labels are then utilized to group together sub-sequences of tokens with the same label, and re-label the sub-sequences accordingly.

8.7.1 Evaluation the MUC way

Since names are often more than a single token in length, the method used for evaluating single-token classification is not applicable to named entity recognition as it stands. In MUC, shades of correctness other than black and white were used; a predicted name can be awarded a score indicating that it is half-right, in some sense. The Message Understanding Conference Scoring Software User's Manual (Voorhees 2001) describes how the performance of a named entity recognizer used in MUC is to be computed. There are six types of possible matches between a predicted named entity and a named entity given as answer key by mark-up in the corpus:

- **COR** (correct). The key and predicted values agree, both in terms of tokens included (spread) and in type of name.
- **INC** (incorrect). The key and predicted values disagree. Not spread, nor type match.
- **PAR** (partially correct). The predicted value and the key are not identical, but partial credit should still be given. For instance, if a predicted value is of the same type as the key value, but only partially overlapping the key.
- **MIS** (missing). There was a key, but no predicted value.
- **SPU** (spurious). There was a predicted value, but no key.
- **NON** (noncommittal). The alignment does not contribute to scoring.

Based on the above listed six different types of matches possible, there are several additional values calculated for the final scoring.

- **POS** (possible). The number of items in the key which contribute to the final score.
$$POS = COR + INC + PAR + MIS \tag{14}$$
- **ACT** (actual). The number of items in the predicted response.
$$ACT = COR + INC + PAR + SPU \tag{15}$$
- **REC** (recall). A measure of how large a degree of key items were predicted, cf. equation 3.
$$REC = \frac{COR + (0.5 \times PAR)}{POS} \tag{16}$$
- **PRE** (precision). A measure of how large a degree of predicted items are actually in the key, cf. equation 2.
$$PRE = \frac{COR + (0.5 \times PAR)}{ACT} \tag{17}$$
- **F-score** cf. equation 4.
$$F\text{-}score = \frac{(\beta^2 + 1.0) \times PRE + REC}{(\beta^2 \times PRE) + REC} \tag{18}$$
where β is set to 1.

A class for evaluating named entity recognition according to the specification in The MUC Scoring Software User's Manual was implemented in Kaba and used for evaluating the performance throughout the remainder of the dissertation.

8.7.2 A baseline for named entity recognition

Equipped with a way of applying single-token classification for recognizing multi-token names, as well as with means to evaluate such an application, the results of using the best base learner configuration from section 8.6.5 for name recognition on the MUC-7 corpus can be presented.

Table 8.10 lists the precision, recall, and F-score for the overall task, as well as for ENAMEX, TIMEX, and NUMEX when using the best learner configuration from section 8.6.5; the REPTree decision tree learner. The results are the average of five runs, in each of which the REPTree learner was trained on 90 randomly drawn documents from the corpus, and evaluated on the remaining 10 documents.

	RECALL	PRECISION	F-SCORE
ENAMEX	0.796	0.808	0.802
TIMEX	0.766	0.784	0.775
NUMEX	0.674	0.916	0.747
ALL	0.789	0.804	0.796

Table 8.10: Baseline named entity recognition results.

A more interesting aspect of learning is how the base learner performs as more data is made available, that is, the learning curve of the learner. Learning curves are commonly used to illustrate the progress of active learning (section 4.9). A baseline learning curve is crucial for judging the success of the experiments on active learning for document selection conducted in chapter 9.

Figures 8.2, 8.3, and 8.4 on pages 116–117 show the baseline learning curve obtained by using the REPTree learner on the MUC corpus; the curves depict recall, precision, and F-score for the ENAMEX, TIMEX, and NUMEX tasks, as well as for their combination. The curve illustrating the learning rate in terms of F-score of the REPTree learner on the combined task in figure 8.4 is the one which is primarily used as a reference in the experiments in chapter 9. As can be seen from the figures, the performance of the base learner does not increase monotonically with the amount of training data. This is most evident for the precision of the classifier on the NUMEX class of names depicted in figure 8.3. The reason for the fluctuations is to be sought for in the distribution of the names. Generally, while the amount of training data available to the

base learner is small, the variance (fluctuation) in its performance is large. On average, there are only 1.4 occurrences of NUMEX per document, as shown in table 7.1. Since the NUMEX class is so sparsely distributed, the effect of each NUMEX training example is more explicit when illustrated by means of the learning curve, than is the effect of each ENAMEX instance. On average, there are 49.58 occurrences of ENAMEX per document (table 7.1), which means that it is necessary to use 35 documents to obtain the same amount of NUMEX training examples, that is available in one document for the ENAMEX class.

A learning curve is obtained by first randomly selecting 10 documents from the MUC corpus to use as a held-out test set. The remaining 90 documents are then iteratively divided into sub-corpora. Initially at iteration $i = 1$ a single document is randomly drawn, without replacement, from the remaining $n = 90$ documents. The document is stored as a separate sub-corpus of size 1 document. In the next round, that is $i = i + 1$, another document is randomly drawn, without replacement, from the remaining $n = n - 1$ documents. The document is stored together with the previously selected documents in a new sub-corpus of size i. This is repeated until there are no documents remaining in the original corpus. At this point, there are 90 sub-corpora of sizes 1 to 90 documents for training, and one for testing. The REPTree base learner is then applied to induce a classifier for each sub-corpus. A curve is then calculated by evaluating each classifier on the test set. This is repeated five times, and the average is shown in figures 8.2, 8.3, and 8.4.

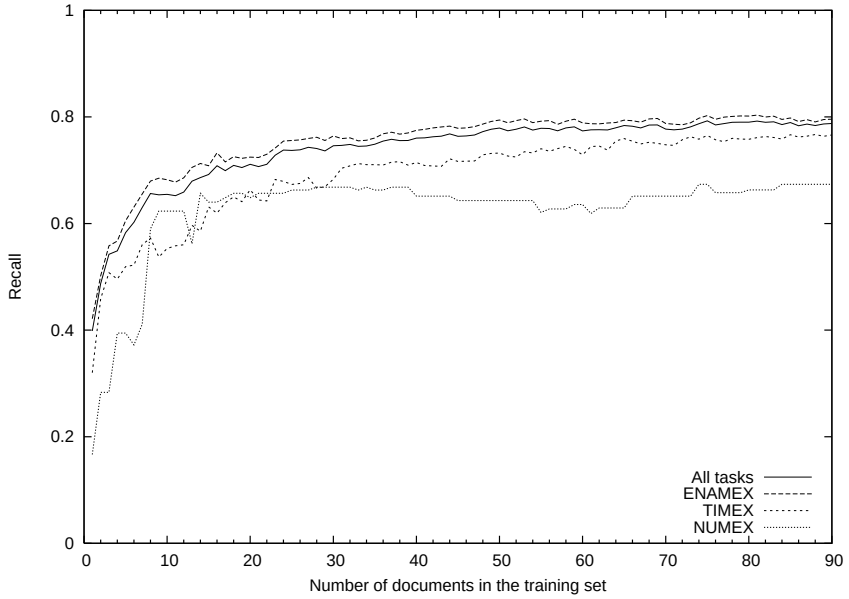

Figure 8.2: A baseline learning curve depicting the recall for the ENAMEX, TIMEX, and NUMEX tasks, as well as their combined recall.

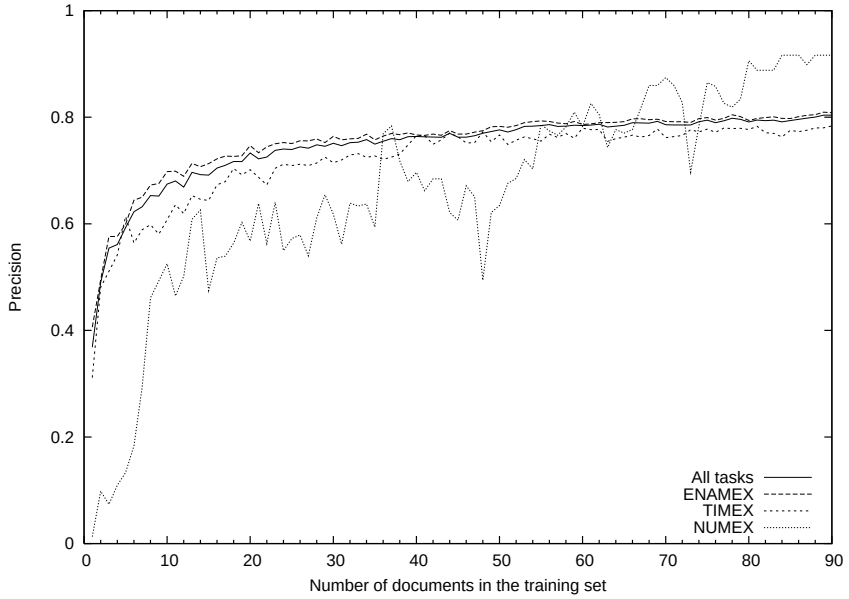

Figure 8.3: A baseline learning curve depicting the precision for the ENAMEX, TIMEX, and NUMEX tasks, as well as their combined precision.

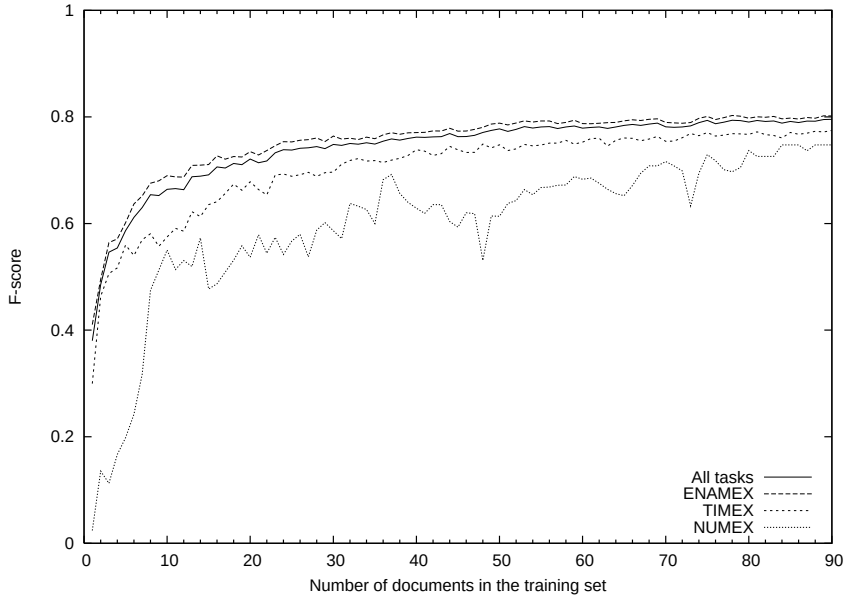

Figure 8.4: A baseline learning curve depicting the F-score for the ENAMEX, TIMEX, and NUMEX tasks, as well as their combined F-score.

9 ACTIVE SELECTION OF DOCUMENTS

This chapter deals with emerging issue E–3 concerning active learning for document selection as outlined in section 6.6. The issue pertains to whether it is possible to actively learn to select documents to mark up with named entities. In particular, the following questions are addressed:

- Which active learning paradigms are applicable to document selection for named entity recognition?
- Which uncertainty, or disagreement metrics are suitable?

As pointed out in the description of the BootMark method in chapter 6, the applicability of the method hinges on the ability to distinguish one document as more informative than another by means of information contained at sub-document levels. Distinguishing between documents based on names is a multi-class *and* multi-instance task. Selecting a document can be considered as selecting a fixed batch of, on average, more than 900 tokens containing approximately 63 names with an average length of around 1.8 tokens (table 7.1). Considering these facts along with the characteristics of the baseline learning curve in figure 8.4, which proves learning from randomly selected documents to yield a rather steep curve which flattens out after only about 20 documents, it is expected to be hard to beat randomly selecting documents from which to learn by means of active learning.

9.1 Active learning experiment walk-through

The active learning experiments conducted are all very similar to the prototypical active learning algorithm outlined in figure 4.1, but the experiments differ in one significant respect from the algorithm; where the algorithm states that the active learner is to ask the human oracle to classify instances, the approach taken in the experiments is to simulate the oracle by substituting the unlabeled instances with labeled instances taken from an annotated reference

corpus. What follows is a walk-through of the algorithm as used in the experiments.

1. The input to an experiment is a labeled corpus D containing k documents, and a base learner configuration B.
2. Select a test set T containing n documents from the labeled corpus D.
3. Select a seed set S containing m documents from D. At this point, D contains $r = k - n - m$ documents.
4. Use B on S to obtain token classifier C.
5. Apply C to each document $d \in D$ in order to mark up the names. The original name annotations in D are not affected by C.
6. Based on the *predicted* names, calculate the informativeness I of each $d \in D$ according to some uncertainty metric if the experiment concerns query by uncertainty, or disagreement metric if it is a query by committee setting.
7. Select the most informative $d \in D$ according to I, obtaining d_I.
8. Simulate an oracle annotating d_I by moving the correctly annotated d_I from D to S. D now contains $r = r - 1$ documents. This is where the experiment set-up differs from the prototypical active learning algorithm in figure 4.1.
9. Apply and evaluate C on T, d_I, and S in order to collect data for further analysis. The evaluation is made with respect to named entity recognition as outlined in section 8.7.1.
10. Repeat steps 4 to 9 until $r = 0$, that is, D is empty.

Essentially, the output of an experiment is the data collected in step 9. Although a classifier is trained, it is not saved for future use.

9.2 Query by uncertainty

The primary purpose of experimenting with query by uncertainty, introduced in section 4.1, is to see whether active learning with a single base learner can beat the passive learning baseline from chapter 8, that is, produce a steeper learning curve, reaching higher F-score with fewer documents in the training

set. Query by uncertainty is attractive since it, if successful, provides an approach to selecting documents which is computationally cheaper than query by committee, which is explored in section 9.3.

9.2.1 Candidate uncertainty quantification metrics

The main problem in this experiment is to identify ways of quantifying the informativeness, or uncertainty, of a document relative to the other documents available. The reason why this is needed is to be able to select the document that, in some sense, confuses the classifier the most. Once correctly marked-up for names and added to the training set, the document is assumed to contribute to the ability of the re-trained classifier to recognize names in previously unseen documents in a way that no other document would have done if selected.

When given an instance to classify, a classifier in Weka can be made to produce as output a probability distribution over the class labels. In the case of token classification as described in chapter 8, such a distribution consists of 15 numbers, each of which indicates the estimated probability that the current instance is of a particular class. One way of understanding the probability distribution produced for a given instance is as a measure of the uncertainty that the classifier has in the classes for the corresponding token. Since the subject matter of the active learning experiment is not individual tokens, but entire documents, the probability distributions for each token in a document have to be combined in some way to reflect the uncertainty that the classifier has in the document as a whole. The remainder of this section introduces 15 metrics for quantifying the uncertainty of documents. The metrics are used in steps 6 and 7 in the active learning experiment set-up outlined in section 9.1.

The candidate metrics used for selecting documents in the query by uncertainty setting described here can be divided into five groups, based on average probability, maximal difference of probabilities, minimal probability, standard deviation of probabilities, and log probabilities (entropy). All uncertainty metrics that make use of the class probabilities obtained from the classifier use only the probability of the most likely class label assigned to each token.

When comparing two documents according to any of the average probability-based uncertainty metrics, the document assigned the smallest value is considered more informative. The metrics are defined as follows:

Avg The average token probability in a document is used as its uncertainty score.

Avgno The uncertainty of a document is represented as the average of the probabilities of the tokens that have not been assigned the class label OUT.

Avgsd A combination of the average token probability Avg for a document, and the standard deviation *sd* of the probabilities assigned to the tokens in a document such that $\text{Avgsd} = \text{Avg} \times (1 - sd)$ is used as uncertainty measure.

Avgsdno As Avgsd, but the average probability of the tokens in a document is calculated the same way as for Avgno.

Contrary to the average probability-based metrics, the ones based on difference in probability should be interpreted such that a document assigned a larger value is more informative than one assigned a smaller value. The probability difference-based metrics are defined as:

Diff The uncertainty of a document is represented as the difference between the largest class label probability assigned to a token in the document, and the smallest value assigned to a token in the same document.

Diffno Analogously to Avgno described above, Diffno is defined as the difference between the largest and smallest probability assigned to tokens in the same document, without considering tokens that have been predicted as belonging to the class representing non-names, that is, with class label OUT.

Sdanddiffno A combination of Diffno and the standard deviation *sd* for the probabilities assigned to the tokens in a document, such that

$$\text{Sdanddiffno} = \text{Diffno} \times sd \tag{19}$$

is used for representing the uncertainty of the document.

Diffbitot Diff weighted according to the ratio of the number of tokens that are assigned class labels indicating that they are at the beginning (B) or in (I) a name, to the total number of tokens in the document such that

$$\text{Diffbitot} = \text{Diff} \times \frac{|IB|}{|IOB|} \tag{20}$$

Diffbitotno Same as Diffbitot but Diffno is used instead of Diff.

Just as with the metrics based on average probabilities, selecting documents using the minimal probability-based metrics is done under the assumption that a lower score indicates a more informative document. The two minimal probability-based metrics used are defined as follows:

Minconf The uncertainty of a document is represented by the smallest probability assigned to a token in it.

Minconfno The uncertainty of a document is represented by the smallest probability assigned to a token that is not predicted as a non-name.

The metrics based on standard deviation of the probabilities assigned to the tokens in a document rely on the assumption that a document with a larger spread in probabilities is a document more informative than one with a smaller spread. The metrics are defined as:

Sd The standard deviation of the probabilities assigned to the tokens is used for representing the uncertainty of the document.

Sdno The standard deviation of the probabilities assigned to all name part tokens is used for representing the uncertainty of the document.

The log probability-based metrics resemble the Shannon entropy as defined in equation 5. A larger log probability-based value indicates a higher uncertainty. The two metrics used are defined as:

Logprob The uncertainty of a document is calculated as

$$\text{Logprob} = -\sum_{i=1}^{k} p_i \log_2 p_i \tag{21}$$

where k is the number of tokens in the document, and p_i is the probability assigned by the classifier to the i-th token.

Tllogprob The token label log probability is the only metric that does not rely explicitly on the probability assigned to each token. Instead, Tllogprob use the relative frequencies of the tokens in a document to compute the uncertainty. In this sense, Tllogprob is more similar the definition of the Shannon entropy in equation 5 than is Logprob defined above.

$$\text{Tllogprob} = -\sum_{i=1}^{k} \frac{1}{|c_i|} \log_2 \frac{1}{|c_i|} \tag{22}$$

where k is the number of tokens in the document, and $|c_i|$ is the number of occurrences in the document of the class label predicted for the i-th token.

9.2.2 Evaluation of the selection metrics

The evaluation of the uncertainty metrics introduced in the previous section is performed by running the active learning experiment as it is described in section 9.1 for all metrics. The experiments are repeated five times for each metric and thus the resulting curves shown in the figures below are the averages of five runs. A successful uncertainty metric is one that causes the learning curve of the base learner to be steeper than, as well as lie above, the baseline curve. The baseline used is obtained by training a single REPTree base learner on randomly selected documents, as described in section 8.7.2. The evaluation of the metrics is run on the full corpus, but constrained to the first 40 iterations; if the resulting learning curve is not above baseline after 40 documents, it is deemed not to be of any help to a human annotator. Further, as the constitution of the seed-set is not the subject matter for these experiments, the seed-set is simply made up by five randomly selected documents from the annotated corpus (step three in the walk-through available in section 9.1). The issue of seed set compilation is explored in chapter 10.

Figure 9.1 shows the results for the average probability-based uncertainty metrics. As the learning curves for all metrics lie under the baseline curve, it is evident that using the various forms of average calculation introduced in section 9.2.1 is not suitable for discriminating between documents to select for annotation.

Figure 9.2 shows the results for the group of probability difference-based metrics. Although using the difference between the largest and smallest token probability results in learning curves that are slightly better than those resulting from the average probability-based metrics, the curves are still mostly below the baseline, and thus not of any use.

Figure 9.3 illustrates the results obtained by using query by uncertainty with the two minimal probability-based uncertainty metrics. The resulting curves are both approximately matching the baseline, which is a general improvement over the results obtained for the two previous groups of metrics. Still, the results are not good enough.

Figure 9.4 shows that using the standard deviation of the predicted probabilities as uncertainty metric is not a good idea. The resulting curves are well inferior to the baseline.

Finally, figure 9.5 illustrates the effect of using the two log probability-based metrics as means to quantify uncertainty over documents. The curves for both metrics actually improve on the baseline, and it is hard to settle for a winner. Although the improvement is obviously small, the results are encouraging since they show that it is possible, even in a single learner setting, to actively select document for named entity recognition.

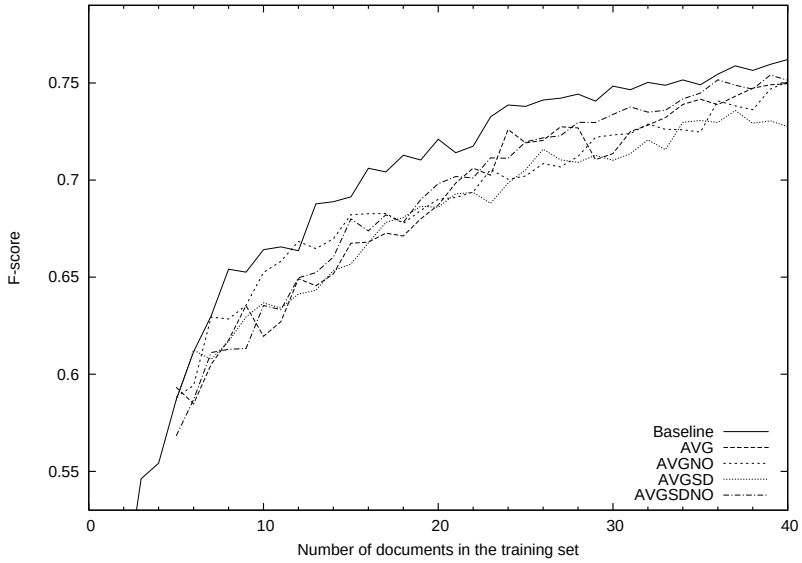

Figure 9.1: The performance of query by uncertainty using the average probability-based document uncertainty metrics.

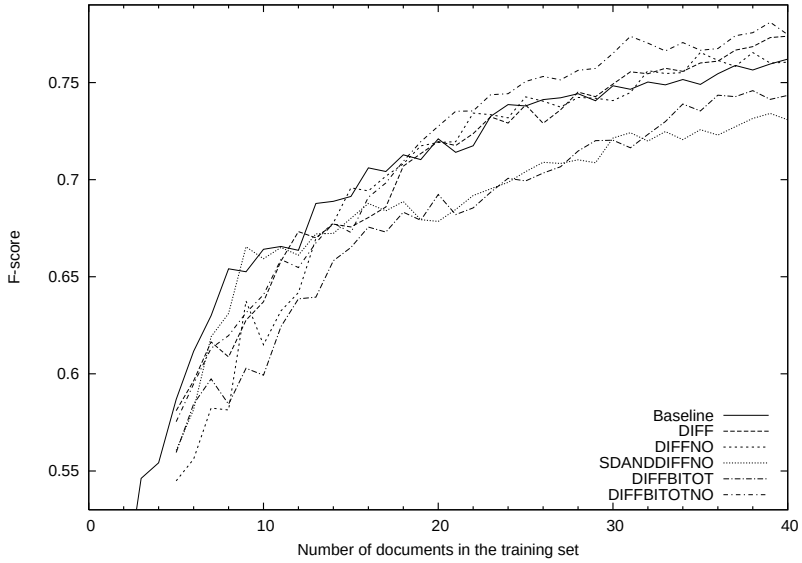

Figure 9.2: The performance of query by uncertainty using the probability difference-based document uncertainty metrics.

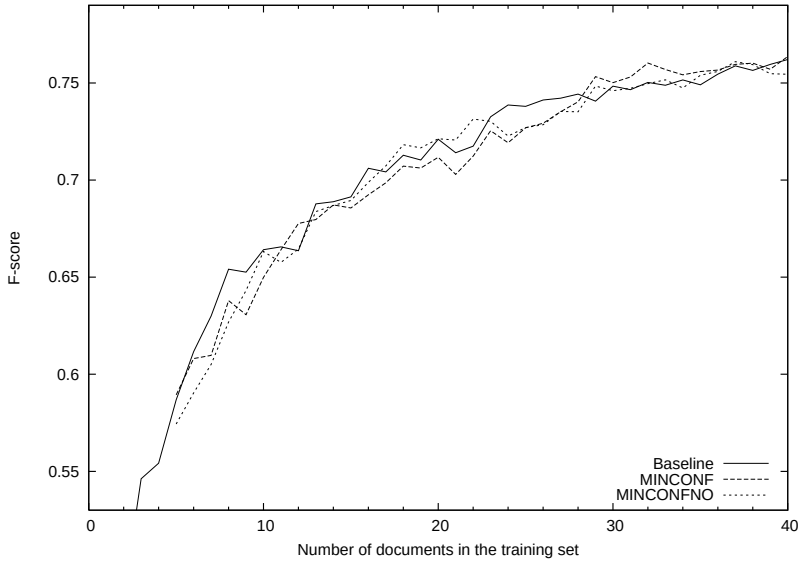

Figure 9.3: The performance of query by uncertainty using the minimal probability-based document uncertainty metrics.

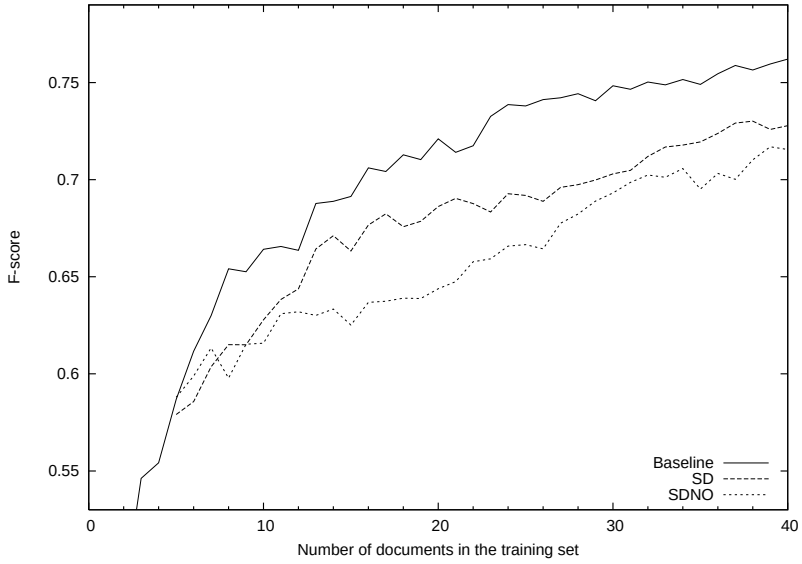

Figure 9.4: The performance of query by uncertainty using the probability standard deviation-based document uncertainty metrics.

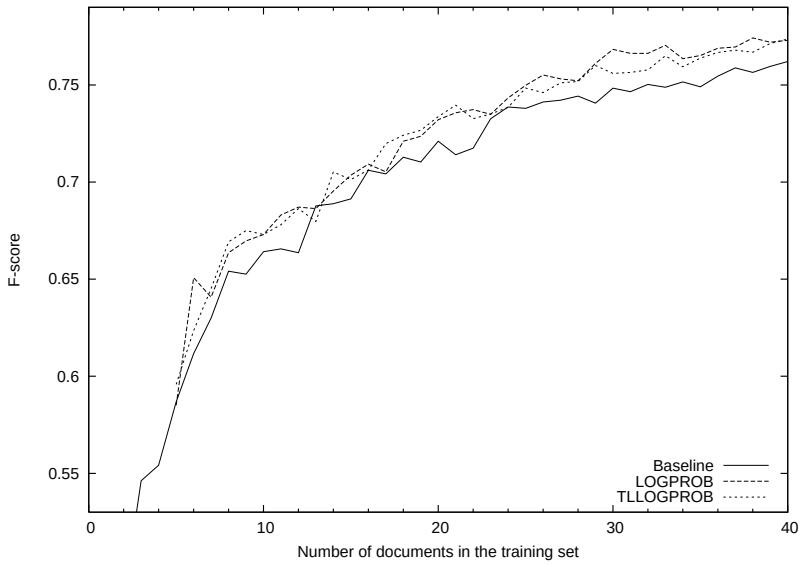

Figure 9.5: The performance of query by uncertainty using the log probability-based uncertainty selection metrics.

The probabilities delivered by a classifier trained on little data are generally of little use as they are calculated based on too few observations. Essentially, in that situation, a classifier can be very certain about the predicted class label for a token, but still be very wrong in its prediction.

9.3 Query by committee

Query by committee is a multi-classifier approach to active learning, in which each committee member contributes to the collective decision as to whether a given instance is more informative than another. Using a decision committee allows for different uncertainty metrics than the ones described in section 9.2.1. The uncertainty of a document is quantified as the disagreement between the members of the decision committee, and the metrics used are therefore referred to as disagreement metrics.

The query by committee approach to active learning is introduced in section 4.2, and the three different ways of assembling a decision committee explored for the purpose of selecting documents to annotate with named entities – query by boosting, ActiveDecorate, and Co-testing – are introduced in sections 4.2.1, 4.2.2, and 4.3, respectively.

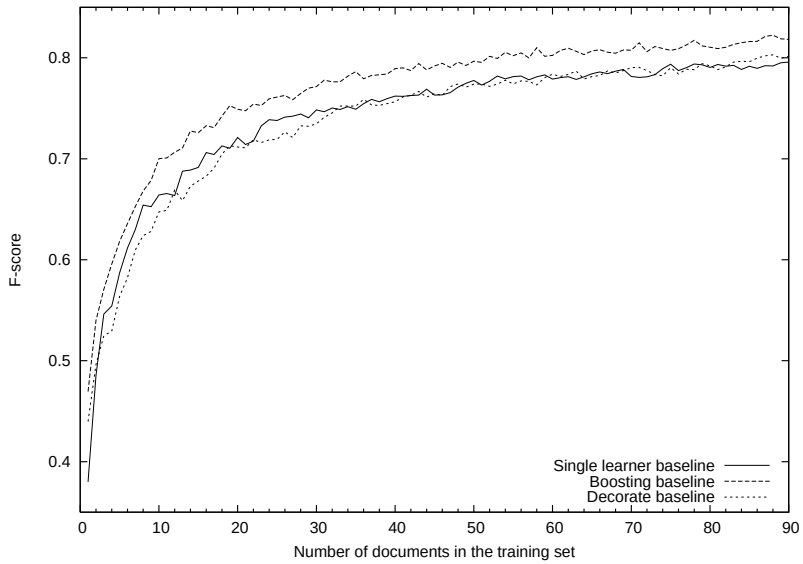

Figure 9.6: A comparison of the Decorate, Boosting and single learner baselines.

Utilizing a different scheme – multi-classifier learning instead of single-classifier ditto – calls for the assessment of additional baselines. Two of the three committee-based approaches have passive learning counterparts; baseline learning curves corresponding to Boosting and Decorate are created and compared to the single learner baseline used in the query by uncertainty setting in figure 9.6. The three ways to create the final classifier in Co-testing, as described in section 4.3, do not lend themselves to be realized in a passive learning setting; all three ways require the presence of an active learning component. Hence, as it stands, there is no passive learning baseline for Co-testing.

Figure 9.6 reveals the differences between the baselines of the single learner, Boosting, and Decorate. Each curve is created as described in section 8.7.2. The Boosting baseline is clearly the best one. For the course of the initial 30 documents, the single learner baseline proves to be better than that for Decorate, while during the remaining 60 documents, the two baselines are intertwined in a way that makes it hard to decide on a winner. The Boosting and Decorate meta learners employ the same base learner configuration as is used for the single learner baseline, that is, the one deemed the best in chapter 8.

9.3.1 Candidate disagreement metrics

The disagreement metrics introduced in section 4.4 apply to the level at which the data is marked-up. In the named entity recognition task, the annotations are made at the token level. Just as is the case with query by uncertainty, the disagreement among committee members as calculated at the token level has to be transferred to apply to the document level.

Most of the disagreement metrics introduced in section 4.4 are defined for two-class tasks, and most of the metrics are reported to be used in single-instance settings. Active learning for the purpose of selecting documents in which to learn to recognize named entities is a multi-class and multi-instance task, something which has to be accounted for when utilizing existing metrics, or developing new ones.

Ten disagreement metrics categorized into five groups are used; based on difference between disagreement values assigned to individual tokens, accumulated log probabilities, minimal margins, vote entropy, and Jensen-Shannon divergence.

9.3.1.1 Minimal margin-based metrics

The notion of minimal margins, introduced in section 4.4.1, is based on the observation that an instance that is assigned two different class labels by a decision committee, where the difference between the probabilities with which each label is assigned is small, constitutes an informative instance. In other words, the disagreement among the committee members is manifested as a small difference in probability between the assignments of competing class labels; the difference is referred to as a margin. The predicted, aggregated, class label for a document is constructed from the predicted class labels for the tokens making up the document. A smaller value on a margin-based disagreement metric indicates a more informative document.

Minmarg-diffno The informativeness of a document is calculated as the difference (margin) between the first and second largest difference between predicted token probabilities assigned to the document such that the resulting class labels assigned to the document are different.

Minmarg-logprob The informativeness of a document is defined as the difference between the largest and second largest assigned log probabilities assigned to the document by the committee members, given that the predicted class labels for the two classifications are different. The log probabilities are calculated as in the case of Logprob defined in section 9.2.1.

Minmarg-tllogprob The same as Minmarg-logprob, but the log probabilities are calculated using relative frequencies instead of the predicted class probability per token. The token label log probabilities are calculated the same way as for Tllogprob defined in section 9.2.1.

9.3.1.2 Vote entropy-based metrics

There are two vote entropy-based metrics used in the current experiment. In vote entropy, the predicted class labels for a token is utilized in order to calculate its informativeness. The vote entropy for an entire document is defined as the average vote entropy of the tokens making up the document. A larger vote entropy value indicates a more informative document.

Vote entropy The vote entropy per token is defined as the original vote entropy metric, available in equation 11. The vote entropy for a document is defined as the average token vote entropy.

Weighted vote entropy Calculated similarly to the original vote entropy metric, but with the weight of the committee members substituted for the votes. The weighted vote entropy, WVE, is defined as:

$$\mathrm{WVE} = -\frac{1}{\log w}\sum_{i=1}^{|c|}\frac{W(c_i,t)}{w}\log\frac{W(c_i,t)}{w} \tag{23}$$

where w is the sum of the weights of all committee members, and $W(c,t)$ is the sum of the weights of the committee members assigning label c to token t. The weights used in these experiments are assigned to the individual classifiers by the meta-learner generating them.

9.3.1.3 Maximal difference-based metrics

Maximal difference-based metrics build on the assumption that a document that is assigned the token probabilities, or token vote entropy, such that the difference between the largest and smallest values is greater than that of another document, is more informative than the latter. Maximal difference is implemented using predicted probabilities per token, or the vote entropy assigned to tokens as defined above.

Maxdiff-no The maximal difference is based on the class probabilities predicted by all committee members for each token in the document, except

for the probabilities assigned to tokens whose predicted class is OUT. The maximal difference is defined as the difference between the largest predicted class probability for a token, and the smallest predicted class probability for a different token in the document.

Vote entropy maxdiff The disagreement between committee members concerning the classification of names in a document is defined as the difference between the highest and lowest vote entropy scores assigned to the document. Vote entropy is calculated as defined above.

Weighted vote entropy maxdiff The same as vote entropy maxdiff, but the weighted vote entropy is used instead of the original vote entropy.

9.3.1.4 Accumulated log probabilities

The accumulated log probabilities metric essentially is an adaptation of the Logprob metric used in query by uncertainty (section 9.2.1). It is also similar to what Körner and Wrobel (2006) refer to as ordinary entropy.

The accumulated log probability for a document is calculated as the sum of the log probabilities assigned to a document by each committee member. This metric does not take advantage of the fact that there are different committee members all classifying the same document in the same manner as the other disagreement metrics do. If the members of the committee are considered to work in parallel when calculating, for instance, the Jensen-Shannon divergence or vote entropy, the members work more in a serialized way when computing the accumulated log probability for a document. A larger value on accumulated log probabilities indicates a more informative document.

9.3.1.5 Jensen-Shannon divergence

The Jensen-Shannon divergence used here is the same as defined in equation 10. The divergence is first calculated for each token in a document. The disagreement among the committee members is then realized as the average Jensen-Shannon divergence per token. A larger value indicates a more informative document.

9.3.2 Query by boosting

Query by boosting is introduced in section 4.2.1. The boosting method utilized in the present experiment is called MultiBoost (Webb 2000), which is an ex-

tension of the AdaBoost algorithm (Freund and Schapire 1996) originally used for query by boosting by Abe and Mamitsuka (1998).

MultiBoost is an extension of AdaBoost in that it aims at combining the variance and bias reduction obtained by AdaBoost with the variance reduction of Wagging (Bauer and Kohavi 1999). Wagging is a form of bagging, introduced in section 4.2.1, which assigns random weights to the instances in each training set, instead of randomly sampling from the original data set to form sub-sets from which to learn. Wagging thus retains the training examples in the original training set, while bagging creates sub-samples that may contain duplicates or is missing out on examples. MultiBoost combines AdaBoost and Wagging by first creating training sets by means of wagging, and then apply AdaBoost to the sub-sets in order to create decision committees. Webb (2000) reports results on using MultiBoost with C4.5 as base learner indicating that MultiBoost produces decision committees with lower error than AdaBoost and Wagging for a large number of UCI data sets (Asuncion and Newman 2007).

All candidate disagreement metrics described in section 9.3.1 are evaluated with query by boosting. The evaluation takes place in a manner similar to that of query by uncertainty in section 9.2.2, with the difference that the number of iterations is reduced from 40 to 30. The seed-set is made up by five randomly selected documents. Passive Boosting, as depicted in figure 9.6, is used as baseline throughout the evaluation of query by boosting.

Compared to the query by uncertainty setting, there is one additional parameter that is essential to most query by committee settings; the size of the committee. The impact of the number of committee members is investigated in section 9.3.5. For the purpose of the evaluation of the candidate metrics with query by boosting, the number of committee members is set to ten.

Figure 9.7 illustrates the effects of using difference-based disagreement metrics as means to select documents to annotate. The Maxdiff-no and vote entropy maxdiff metrics result in approximately the same performance. The curve resulting from utilizing Weighted vote entropy maxdiff reveals learning performance which is even better. It is thus discernible that it is possible to utilize active learning for the purpose of selecting documents in a named entity recognition task. The question now arising is whether the results can be even more improved by one of the other disagreement metrics.

In figure 9.8 the margin-based disagreement metrics are compared to the Weighted vote entropy maxdiff metric which egressed as the best of the difference-based metrics. Körner and Wrobel (2006), who investigated the effects of different disagreement metrics in multi-class settings as described in section 4.4, advocate the use of margin-based metrics. However, any overly strong expectations are put to shame here since, as illustrated in figure 9.8, the margin-based metrics utilized for the present multi-class and multi-instance task do not

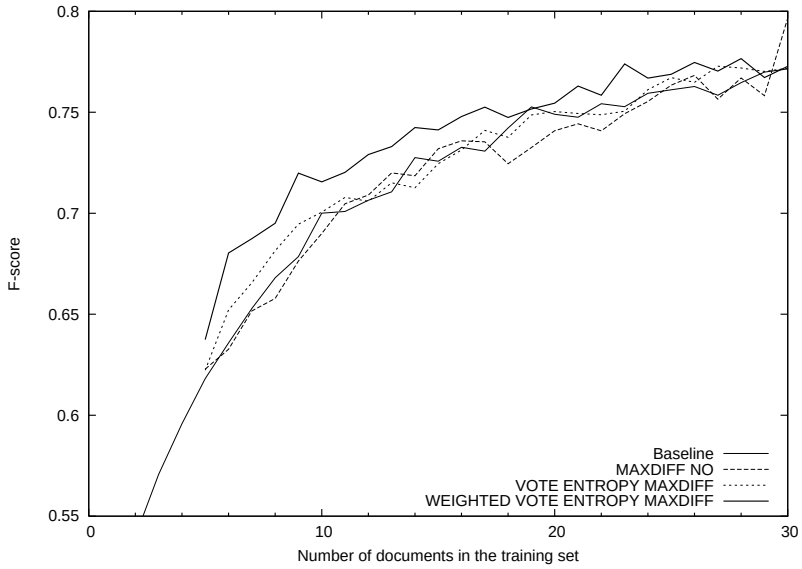

Figure 9.7: Query by boosting with difference-based document selection metrics.

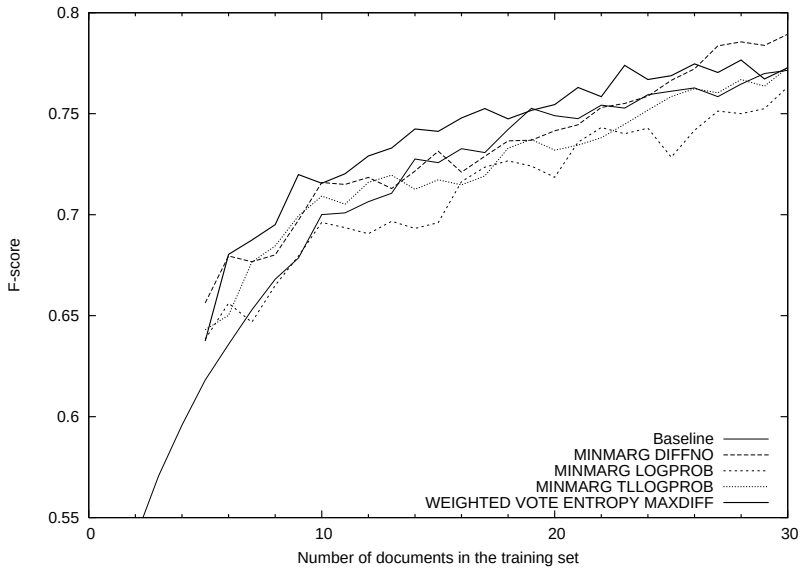

Figure 9.8: Query by boosting with margin-based document selection metrics.

result in a learning curve improving on the baseline. The reason is likely to be found in the fact that the disagreement measures used in the margin-based metrics used here apply to entire documents, and not only individual instances as stipulated by Körner and Wrobel (2006). Any disagreement metric for a piece of data that is defined as the aggregate of the disagreement on its sub-parts runs the risk of becoming a rather blunt instrument, a risk that is not at all specific to the case of margin-based disagreement.

Figure 9.9 shows the application of the vote entropy-based metrics along with the accumulated log probabilities metric. The Weighted vote entropy maxdiff curve is included for comparison. Although it is still better during the first six documents, the Weighted vote entropy maxdiff metric is henceforth replaced by the Vote entropy metric as being the best metric. The reason is that the vote entropy learning curve is more parallel to the baseline curve than is the difference-based curve. In fact, the curve generated by using vote entropy is very similar to a transposition by the baseline curve of approximately 2% F-score.

None of the metrics discussed up until now makes use of the predicted class probability distribution for each token presented by the classifier. The Jensen-Shannon divergence is a metric which does and figure 9.10 shows the resulting learning curve. Utilizing the Jensen-Shannon divergence as disagreement metric results in a performance which is on par with the passive Boosting baseline.

9.3.3 ActiveDecorate

As introduced in section 4.2.2, ActiveDecorate is the active learning counterpart of the Decorate method (Melville and Mooney 2003). The core idea in Decorate is to generate artificial data in line with the characteristics of the available training data in order to further extend the amount of data available for training a committee of classifiers. The disagreement among the classifiers can be measured in much the same way as for query by boosting, with one important exception; while the members of a boosted committee receive weights according to how good they are, in some sense, the committee members in Decorate all are assigned the same weight. This rules out the use of the disagreement metrics based on weighted vote entropy.

The evaluations of ActiveDecorate with the various disagreement metrics are conducted the same way as for query by boosting in section 9.3.2; the seed-set consists of five randomly selected documents, and the committee is comprised by ten members. Figure 9.11 shows the results of using the difference-based metrics to select documents to annotate. Both metrics yield classification performance approximately in line with the Decorate baseline.

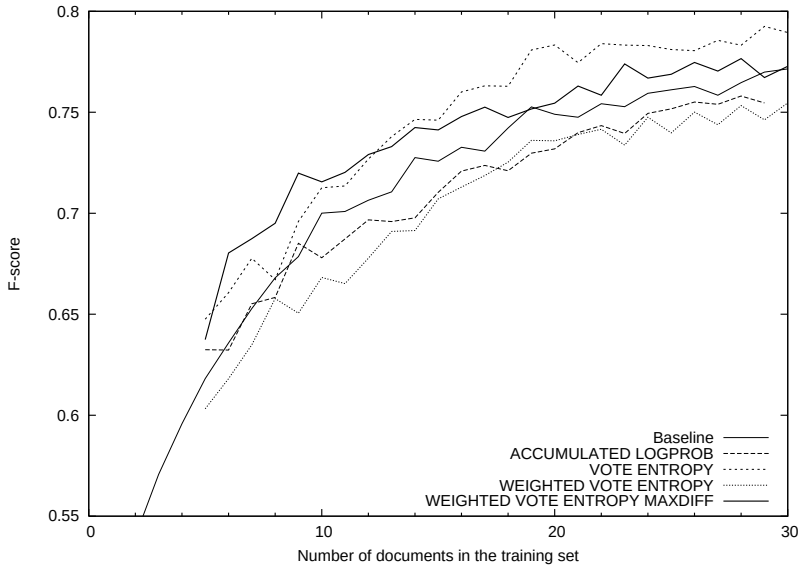

Figure 9.9: Query by boosting with entropy-based document selection metrics.

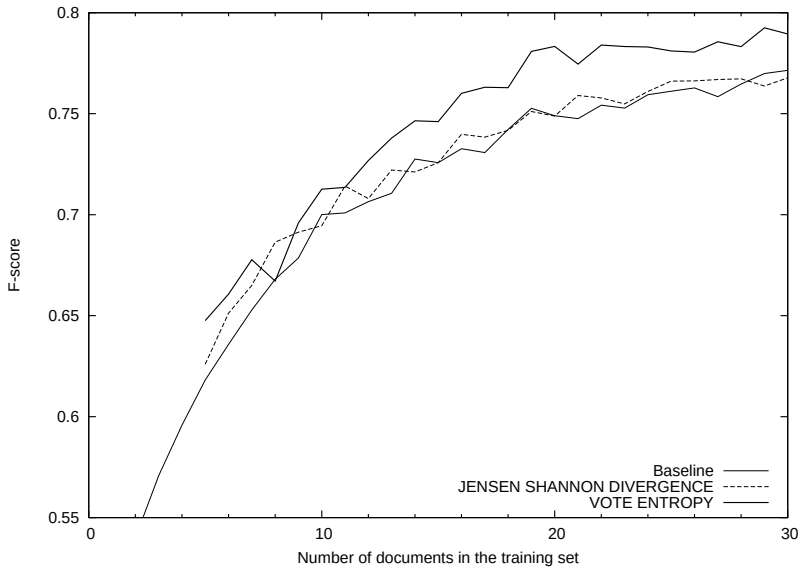

Figure 9.10: Query by boosting with Jensen-Shannon Divergence as document selection metric.

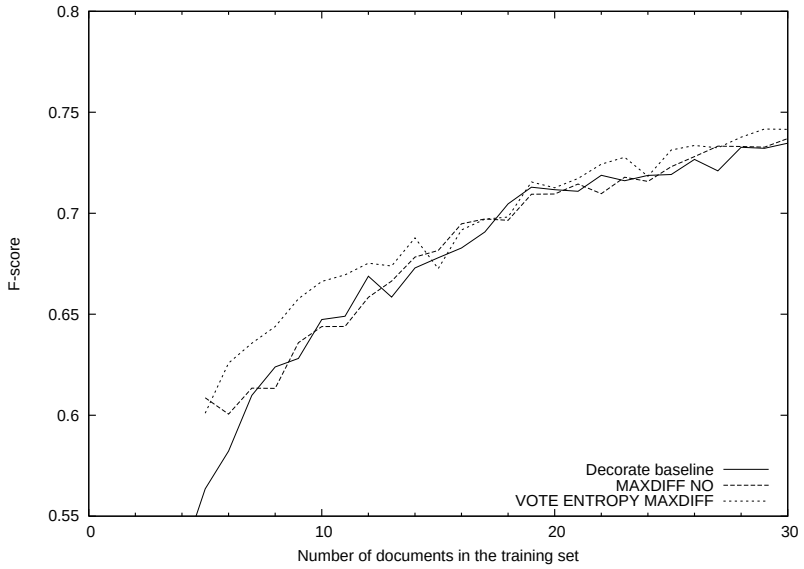

Figure 9.11: ActiveDecorate with difference-based document selection metrics.

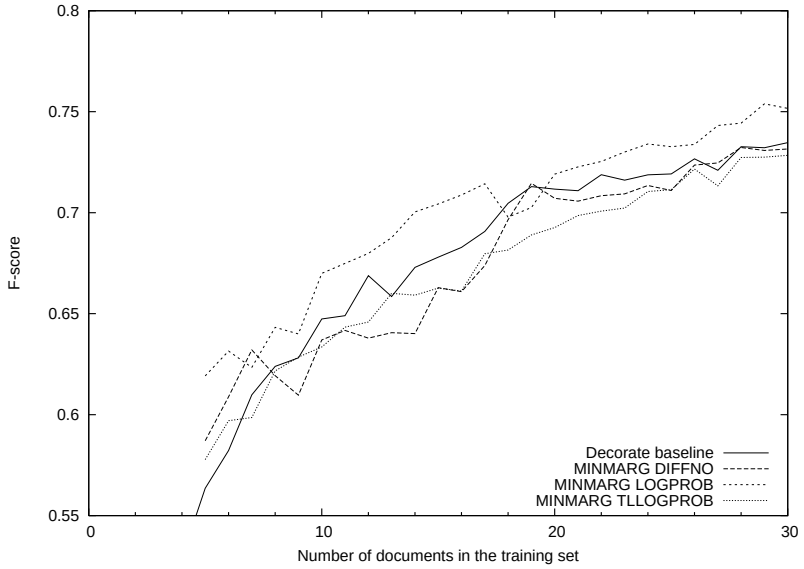

Figure 9.12: ActiveDecorate with margin-based document selection metrics.

The results provided by the three margin-based metrics are depicted in figure 9.12. Overall, document selection based on the Minmarg-logprob metric yields classification performance better than the baseline, while the results produced by the two other margin-based metrics do not.

The best selection metric to use in conjunction with ActiveDecorate is the Accumulated-logprob metric illustrated in figure 9.13. Quite contrary to query by boosting, selecting documents based on vote entropy results in performance of ActiveDecorate inferior to the baseline.

Finally, figure 9.14 shows the classification performance resulting from using the Jensen-Shannon divergence as selection metric. For the greater course of the 30 documents processed, the classification results are worse than those of the baseline.

Although the best selection metric used with ActiveDecorate improves on the corresponding baseline more than the best metric used in conjunction with query by boosting does, the best query by boosting setting is still to be preferred over the ActiveDecorate ditto. When comparing the learning curve for the vote entropy metric used with query by boosting in figure 9.9, and the Accumulated-logprob metric used with ActiveDecorate in figure 9.13, it is clear that the classification performance achieved by the former is superior to that of the latter. Hence, when considering classification performance, query by boosting using vote entropy is to be preferred over any of the ActiveDecorate configurations.

ActiveDecorate has been reported successful on tasks available in the UCI data sets (Asuncion and Newman 2007), such as the classification of diseases in soybean plants (Melville and Mooney 2004). However, the characteristics of the data sets and the set-up used by Melville and Mooney (2004), and the corpus used here differ in one significant respect; the UCI data sets used by Melville and Mooney are relatively small, containing up to 900 instances. Accordingly, Melville and Mooney select small batches of examples in each iteration, reportedly up to three instances in size. This should be compared to the on average 900 tokens (instances) which are selected in each iteration in the current experiment.

Some of the UCI data sets used by Melville and Mooney (2004) contain more classes than does the MUC-7 named entity recognition task pursued. Thus, the number of classes is not likely to be the reason why ActiveDecorate is unsuccessful, relative to query by boosting, for selecting documents for named entity mark-up.

Melville and Mooney (2004) use two disagreement metrics: margins and Jensen-Shannon divergence (introduced in sections 4.4.1 and 4.4.6, respectively). Due to the multi-instance nature of the task pursued in this thesis, the margins metric used by Melville and Mooney does not have a direct corre-

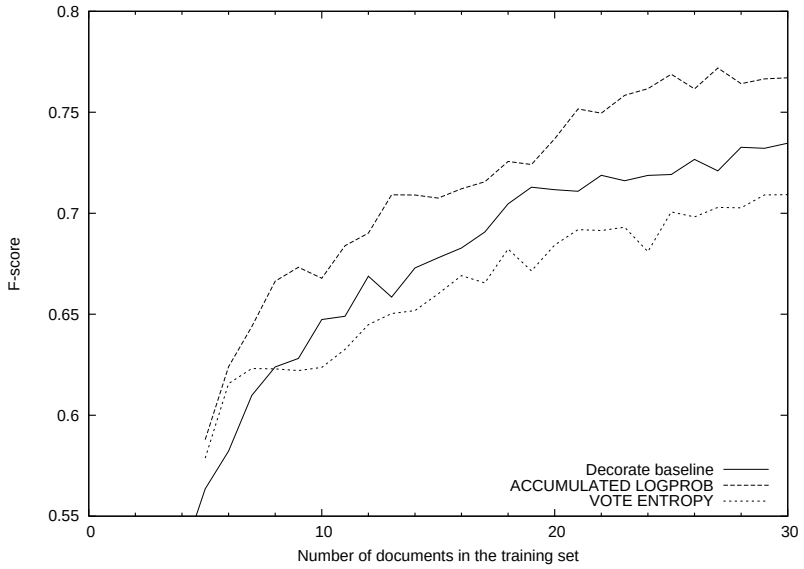

Figure 9.13: ActiveDecorate with entropy-based document selection metrics.

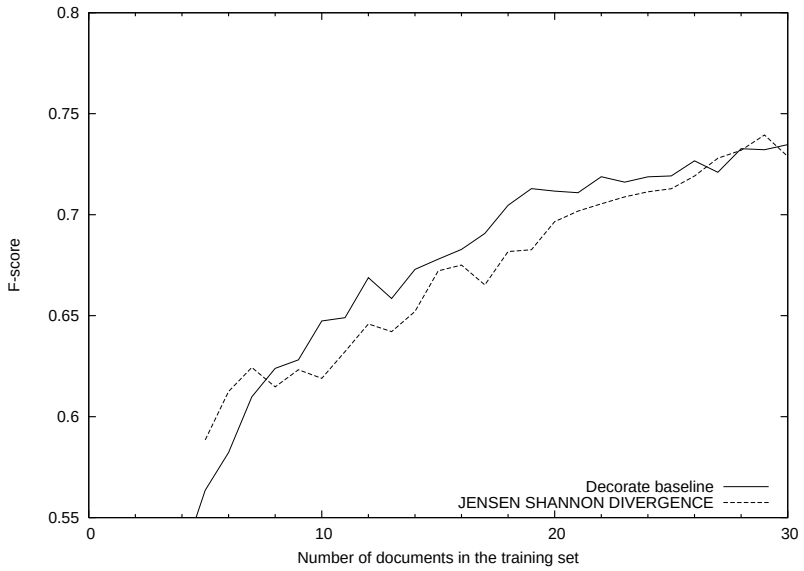

Figure 9.14: ActiveDecorate with Jensen-Shannon divergence as document selection metric.

spondence to any of the disagreement measures introduced in section 9.3.1. The disagreement metrics closest to the margins metric are Minmarg-diffno, Minmarg-logprob, and Minmarg-tllogprob for which the active learning results are reported in figure 9.12; none of the metrics yield results better than the Decorate baseline. The results reported by Melville and Mooney (2004) show that actively selecting instances by using the margins metrics results in a better classification performance than is achieved with Jensen-Shannon divergence.

Melville and Mooney (2004) further report that ActiveDecorate outperform query by boosting in their experiments, in which they used J48 as base learner and 15 members in the decision committees. The impact of altering the number of committee members in ActiveDecorate is examined in section 9.3.5.

Körner and Wrobel (2006) report on successfully using ActiveDecorate for multi-class problems. Their experiment set-up is similar to that of Melville and Mooney's (2004) in that they too select few instances in each round (one, to be precise), use data sets from the UCI repository, and employ J48 as base learner. Thus, it seems reasonable to assume that the failure of the ActiveDecorate set-up used here is due to the size and constitution of the batch of instances selected in each active learning iteration. The size of each batch by far exceeds those used by Melville and Mooney (2004) and Körner and Wrobel (2006). In addition, the order of the instances, as well as the composition of each batch is fixed. This, in turn, implies that the distribution of classes within each batch is severely skewed in favor of the least interesting class, that is OUT, as described in section 7.1. Considering that the Decorate base learner more often than not fails in generating as many committee members as requested, as described in section 9.3.5, the training data at hand seems quite hard to use in order to produce synthetic data; this observation can be taken as a support of the claim that the problem with ActiveDecorate is due to the characteristics of the training data.

9.3.4 Co-testing

The idea in Co-testing is to learn a task by using different feature sets describing the same data. This way, the problem of having access to too small an amount of training data is addressed, without the data itself having to be tampered with as is the case of other committee-based methods such as Boosting and Decorate. Active learning using multiple views of the data is introduced in section 4.3.

9.3.4.1 Defining the views

An original assumption of multiple-view learning is that each view has to be independent of the others and that each view is compatible with the task at hand. Nevertheless, as described in section 4.3.1 the assumption can be relaxed to some extent, as long as the base learners involved are able to correctly learn the task using positive training examples only, which is arguably the case with named entity recognition.

It is debatable whether there exists a natural split of the feature sets used in this experiment to recognize names. Finding two sets of features that are sufficiently independent and sufficiently compatible with the task is a non-trivial issue. Since the best base learner configuration obtained in chapter 8 utilized a zero-context (c0) for learning to classify tokens as being part of names, the current endeavour explores the same context size. The c0 feature set split used is described in table 9.1. Essentially, the c0 feature set is split so that the features in one view pertain to the surface appearance of the target token, while the other view consists of the remaining features. The features and their categorizations are available in table 8.2.

In addition to the c0 feature split, a different way of splitting the feature set is examined by means of using a context size in which the target token is represented by the token itself in conjunction with the token immediately on its left, and the token immediately on its right (c1). The c1 feature set split is made so that the features corresponding to the c0 make up one view, and the remaining features, representing the context of the target token, make up the second view.

There is one earlier approach to Co-testing and named entity recognition. Becker et al. (2005) report on successfully using Co-testing for recognizing astronomical named entities. The feature set split used by Becker and colleagues is handcrafted in such a way that each view yields similar results. However, their feature set split is not applicable to the current experiment since the overlap between the features used here (outlined in section 8.2), and those employed by Becker et al. (2005) are too dissimilar.

Instead of empirically validating that the views produce similar results, the way the two view classifiers are combined into a final classifier makes use of a novel weighting scheme as described in the following section. The weighting scheme seeks to adjust the influence of one view classifier over the other by weighting them according to their performance on the training data. If the weights assigned to each view are similar, it means that the definitions of the views are equally expressive relative to the task and training data at hand.

VIEW 1	VIEW 2
containsDigitsAndDollarSign	caordTag
containsDollarSign	caseTag
containsPercent	classOfPreviousPreviousToken
containsPunctuation	classOfPreviousToken
containsWhiteSpace	compTag
isAllCaps	dependencyFunction
isAllDigits	grammaticalFunction
isAllLowerCase	isFirstInSentence
isAlphaNumeric	isFirstName
isAnyOrAllDigits	isLocation
isCompanyDescriptor	isNamePart
isDigitsAndAlpha	isSalutation
isDigitsAndComma	morphTag
isDigitsAndDash	numTag
isDigitsAndPeriod	partOfSpeech
isDigitsAndSlash	surfaceFormAsLemma
isFourDigitNum	syntacticTag
isHyphen	tenseTag
isInitCaps	
isRomanNumber	
isSentenceDelimiter	
isSingleCapPeriod	
isSingleChar	
isSingleCharAndPeriod	
isSingleLowPeriod	
isTwoDigitNum	
length	
prefix	
suffix	
surfaceForm	

Table 9.1: The split of the c0 features into two views used for Co-testing. View 1 consists of features originating from the surface appearance of the token, while view 2 contains the features relying on pre-compiled lists, linguistic processing, and the prediction made concerning the class label of the token preceding the current one.

9.3.4.2 Combining the view classifiers into a final classifier

The question of how the view classifiers are to be combined so as to form a single, final classifier needs to be addressed. A combined classifier is required in order to be able to mark up names in unseen text. Muslea, Minton and Knoblock (2006) suggest three ways of combining view classifiers; by having the classifiers vote and combine the votes using each classifier's confidence estimation in its prediction as weights, by combining the votes by majority, or

by assigning the classifier that performed the best as the final classifier (see section 4.3). The approach used in this experiment is a variant of the weighted voting scheme.

The probability distributions p and q produced for each token in the input by the two view classifiers are combined into a new probability distribution k such that each element k_i in k is

$$k_i = w_1 p_i + w_2 q_i \tag{24}$$

where p_i, q_i denote the i-th element in distributions p and q, respectively. The weights of the view classifiers $w_1 + w_2 = 1$. The resulting k then represents the combined classifier's classification for the token.

Two variants of weighting are used. The first uses $w_1 = w_2 = 0.5$. In the second variant, the weight for a view classifier is set according to the accuracy obtained when the classifier is evaluated, using 10-fold cross-validation, on the training set. The weights are re-calculated in each iteration of the active learning loop, as outlined in section 9.1, and they are normalized to sum to 1. The motivation for dynamically re-weighting the influence of the classifiers is to favor the best classifier, and suppress the weaker one, thus avoiding using a view if it is less compatible with the task. If the views are equally compatible with the task, it is anticipated that the use of dynamic re-weighting of the view classifiers would have less impact, that is, the two weights would end up converging towards 0.5.

9.3.4.3 Co-testing results

Unfortunately, none of the Co-testing configurations explored managed to produce results better than the single learner baseline. Co-testing is thus not of any practical use to the present experiment. Figures 9.15, 9.16, 9.17, and 9.18 illustrate the performance of Co-testing using the c0 feature set split presented in table 9.1, and the disagreement metrics introduced in section 9.3.1. Figures 9.19, 9.20, 9.21, and 9.22 show the performance of Co-testing using the c1 feature set split introduced earlier in this chapter along with the same disagreement metrics as are tested with the c0 feature split. In the figures, the weighting scheme that produced the best results for each metric is shown. For instance, using the c0 feature set split and the Accumulated log probabilities disagreement metric, the dynamically weighted combination of the view classifiers into a single classifier yields results better than the equally weighted way of combining classifiers; only the learning curve for the former configuration is on display.

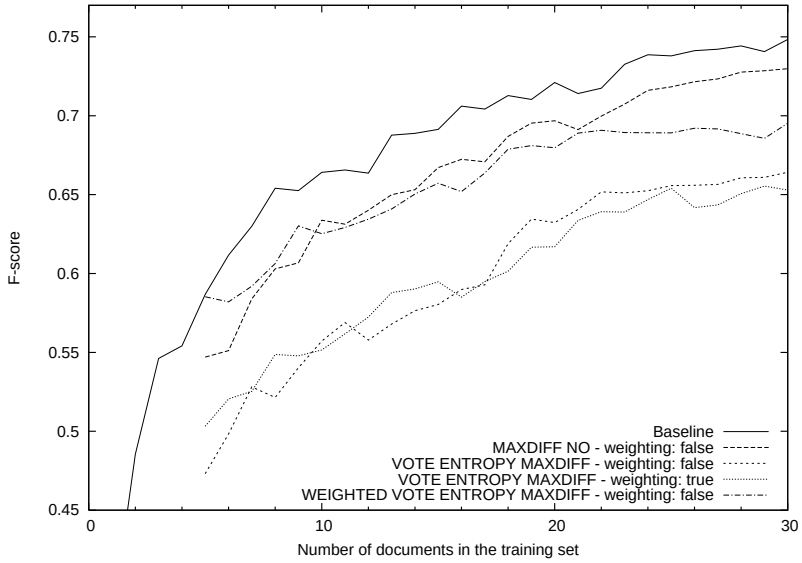

Figure 9.15: Co-testing using the c0 feature set split with difference-based document selection metrics. The two weighting schemes produce inseparable results for vote entropy maxdiff, hence the inclusion of both vote entropy maxdiff curves.

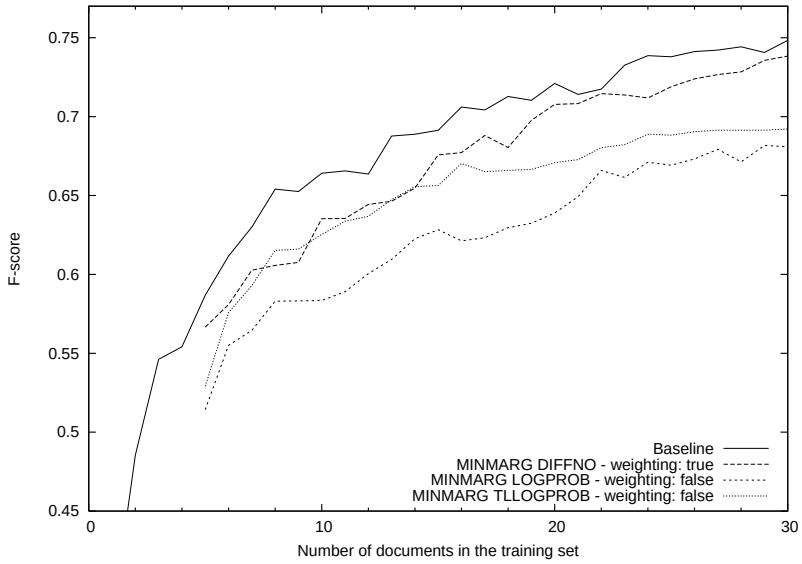

Figure 9.16: Co-testing using the c0 feature set split with margin-based document selection metrics.

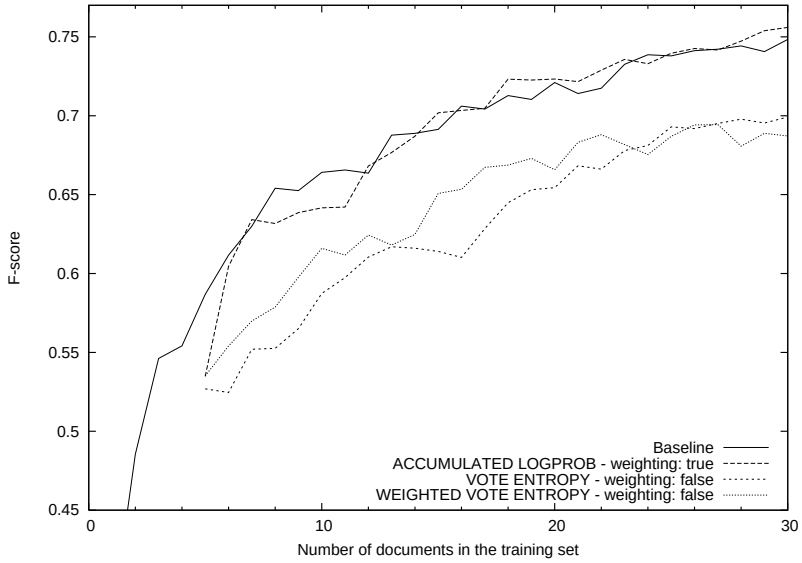

Figure 9.17: Co-testing using the c0 feature set split with entropy-based document selection metrics.

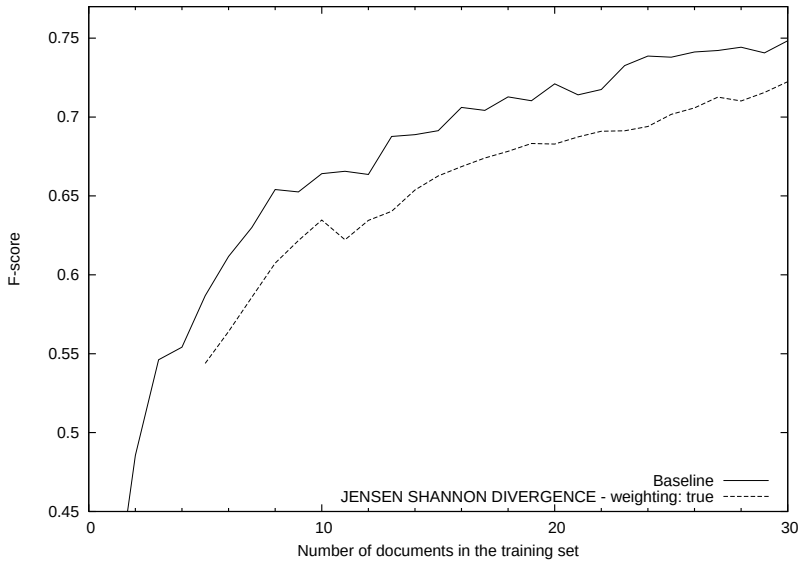

Figure 9.18: Co-testing using the c0 feature set split with Jensen-Shannon Divergence as document selection metric.

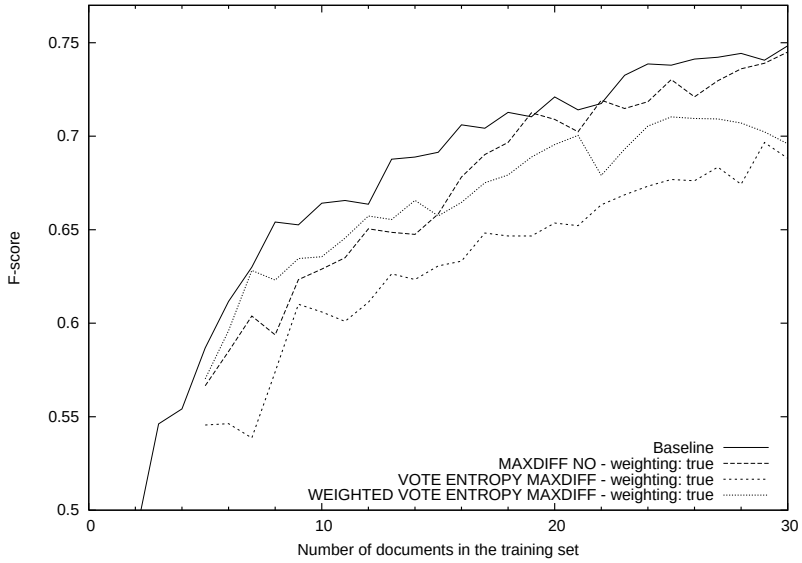

Figure 9.19: Co-testing using the c1 feature set split with difference-based document selection metrics.

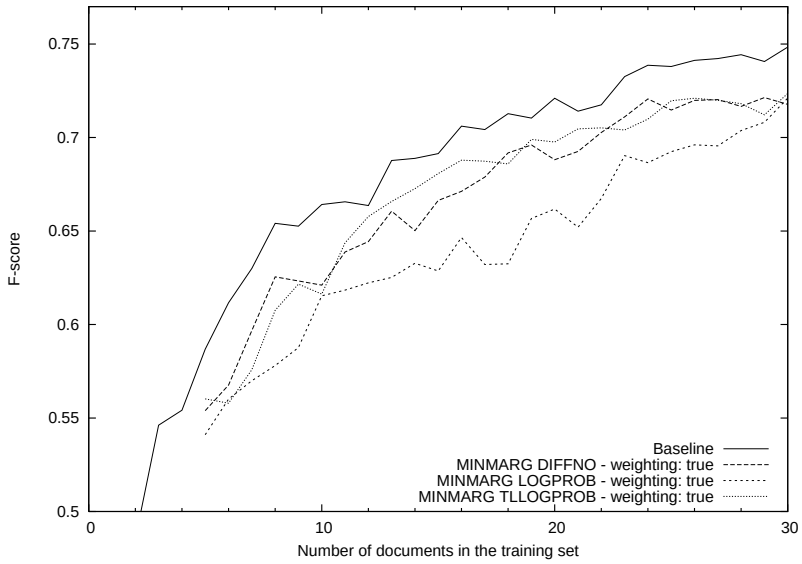

Figure 9.20: Co-testing using the c1 feature set split with margin-based document selection metrics.

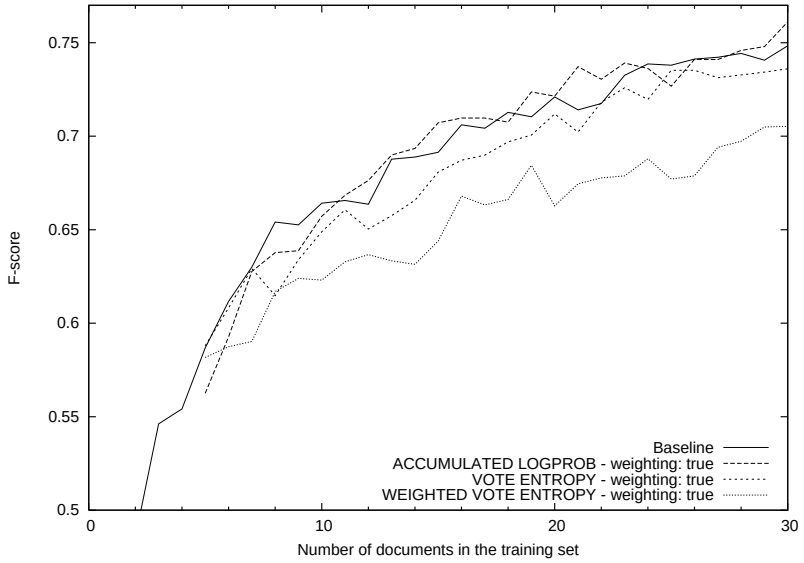

Figure 9.21: Co-testing using the c1 feature set split with entropy-based document selection metrics.

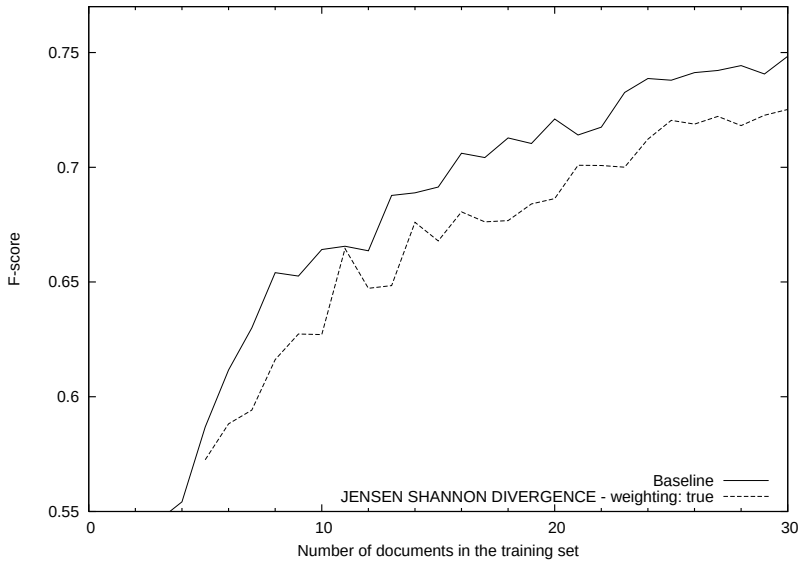

Figure 9.22: Co-testing using the c1 feature set split with Jensen-Shannon Divergence as document selection metric.

For the c1 feature set split, the dynamic re-weighting of the view classifiers produces better results than the equally weighted classifiers for all disagreement metrics. This fact implies that the c1 feature views are not equally compatible to the task, and hence that the split is not an optimal one. For the c0 feature set split, however, the equally weighted view classifiers produce results better for 6 of the 10 disagreement metrics, while using dynamically weighted view classifiers win in 2 cases, indicating the c0 views to be better off than the c1 ones. This, in turn, suggests that splitting the feature sets according to type of features, as is the case with the c0 views, might be better than to split the feature sets according to the surface context, which is done in the c1 split.

Due to the lack of positive results, Co-testing will not be further investigated within the scope of this thesis. However, Co-testing as such should not be dismissed as a plausible candidate for active learning for multi-class problems such as named entity recognition. What is needed is a more thorough investigation into the issue of (semi) automatically splitting the feature set into separate views, and means to validate the views with respect to their independence and compatibility to the task at hand.

9.3.5 Effects of the committee size

The number of members in the decision committee is likely to have an impact on the results obtained by a query by committee setting. Out of the three query by committee approaches examined in this chapter, two allow for the alteration of the number of committee members; query by boosting, and ActiveDecorate. Depending on the way that the features used for representing the data lend themselves to be split into different views, Co-testing is inherently tied to using a small committee, usually with two members. Given the results reported in section 9.3.4, it is not likely that further forking the features, and thus accommodating for additional committee members, would be beneficial to the overall performance. Hence, the effect of varying the number of ensemble members for Co-testing is not within the scope of the current investigation.

Figure 9.23 shows that varying the number of committee members has an impact on the performance of query by boosting. The figure illustrates the use of 2, 5, 10, and 20 committee members; using 2 and 5 members produce results worse than the passive Boosting baseline. Note that there is a small difference between using 10 and 20 members. In fact, figure 9.23 suggests that the results of using 20 committee members are inferior to those of using 10 members. The relation between the number of committee members and the performance is anticipated; in Boosting, earlier committee members contribute more to the performance of the committee than do subsequent members (Webb 2000).

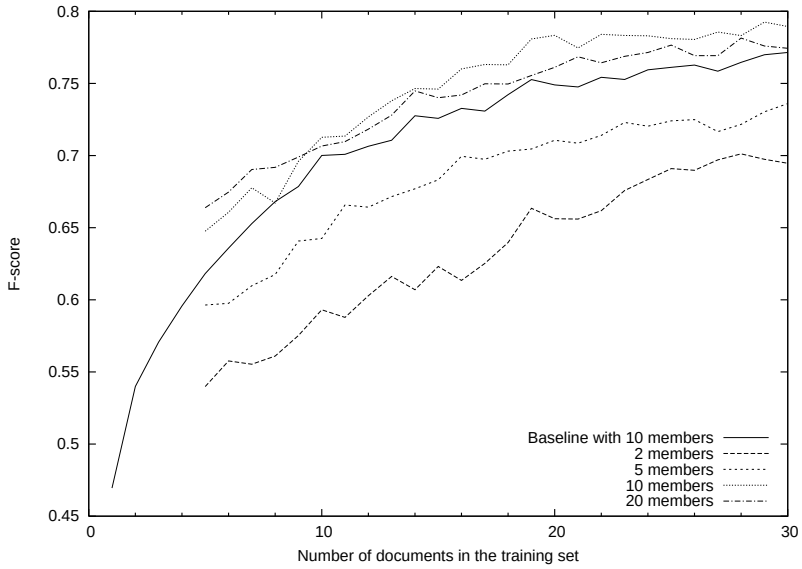

Figure 9.23: Effects of altering the number of committee members in query by boosting. Vote Entropy is used as document selection metric.

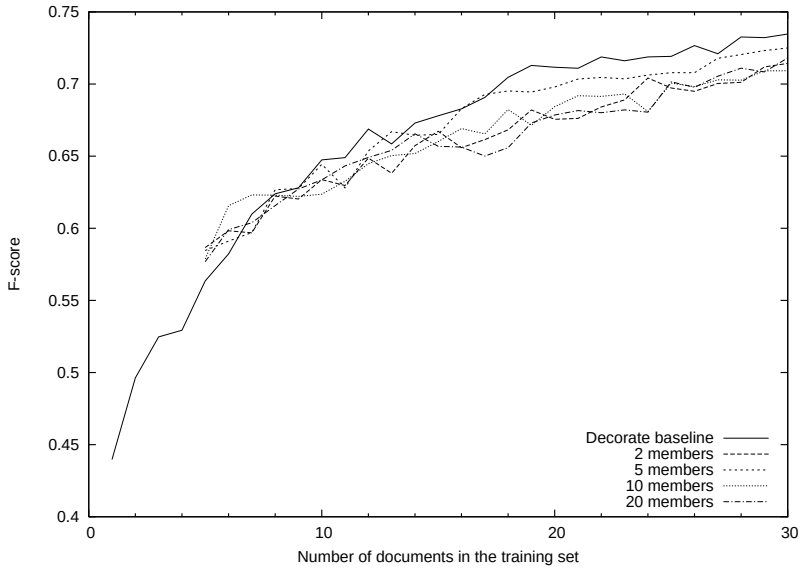

Figure 9.24: Effects of altering the number of committee members in ActiveDecorate. Vote Entropy is used as document selection metric.

For ActiveDecorate, requesting the meta learner to generate different numbers of committee members has virtually no effect at all, as illustrated by figure 9.24. The results obtained are all approximately the same. However, there is more to this than meets the eye. Decorate – the passive counterpart to ActiveDecorate – operates in a different manner than MultiBoosting for creating sub-samples of the training data. Where MultiBoosting uses re-weighting of the training instances to form sub-sets with different characteristics, Decorate generates additional, synthetic instances based on the characteristics of the training data at hand and the contribution to classifier performance made by the synthetic training examples. The bottom-line is that Decorate is more involved with the characteristics of the training data when generating sub-sets on which to train committee members, than is MultiBoosting. This becomes evident when looking at the relation between the requested number of committee members, and the actual number of members generated by Decorate as illustrated in figure 9.25. The information in the figure is based on the average of five runs. Requesting Decorate to generate two committee members always succeeds. On the other hand, requesting Decorate to supply more than two members always results in a number less than the desired. Nonetheless, the lack of effect of altering the committee size in ActiveDecorate depicted in figure 9.24 cannot be explained by the inability of Decorate to appropriately respond to the request for a specific number of committee members alone; despite the large variation on actual committee members, the performance of ActiveDecorate does not exhibit the corresponding fluctuations. ActiveDecorate appears to be fairly insensitive to the number of committee members involved, albeit in a rather discouraging way.

The actual number of committee members obtained in query by boosting corresponds to the number of members requested; MultiBoosting obviously is able to accommodate the learning process in such a way that the partitioning of the training data is less sensitive to the data characteristics than is Decorate.

Another expected effect of increasing the number of committee members is the increase of the time required to create the committee. Figure 9.26 illustrates the training and testing time as an effect of the number of committee members for query by boosting. The time required to train is measured for an increasing amount of data, while the time required to test the learned models is measured for the 10 documents in the test set. The ratio of the training time to the testing time is used as time measurement in order to abstract away the particulars of the computer used (see section 8.6.1). Increasing the number of committee members has an articulate effect on the amount of time needed to train the committee; a committee consisting of 20 members require approximately 3 times the amount of time to learn from 5 documents, than is required to classify the same amount of data. When facing 30 documents in the training data, the

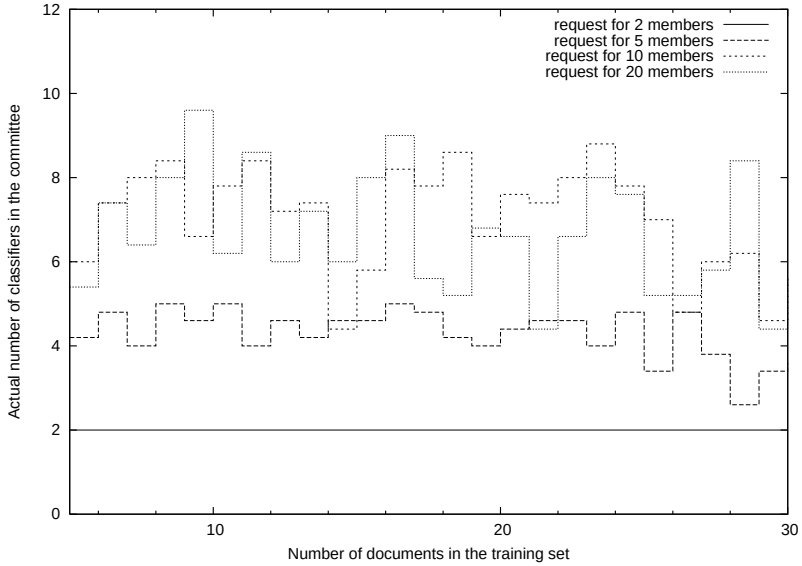

Figure 9.25: The desired number of committee members versus the actual number of members generated by Decorate.

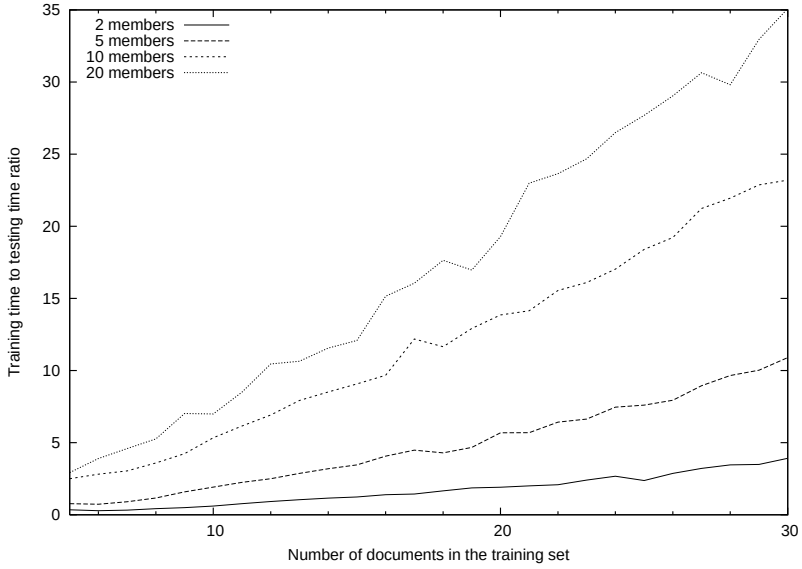

Figure 9.26: The effect on increase in time required to train and test the committee used in query by boosting as an effect of the committee size.

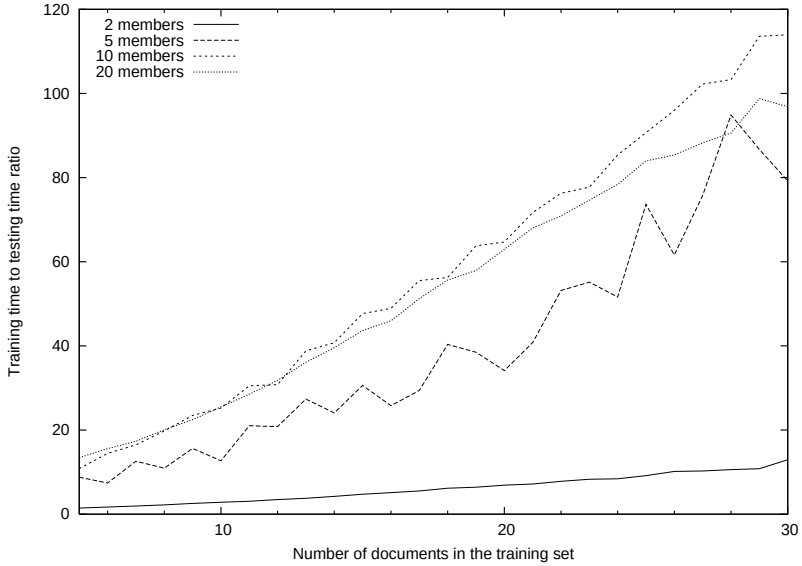

Figure 9.27: The effect on increase in time required to train and test the committee used in ActiveDecorate as an effect of the committee size.

20 committee members require 35 times more time to train on data, than is required to classify the same amount.

Looking at the corresponding time ratios for ActiveDecorate available in figure 9.27, it is clear that the phenomenon is not isolated to query by boosting; rather, it is an effect of the use of committee-based learning methods proper. ActiveDecorate exhibits even worse training to testing time ratios than query by boosting. Even though the desired number of committee members appears to be uncorrelated to the obtained number of members, as illustrated in figure 9.25, the effect of increasing the number of committee members on the training and testing time is evident. When Decorate tries to learn a desired 20 committee members on 5 documents, the training time required is approximately 15 times the time required to classify the same amount of data. The same ratio when faced with 30 documents is approximately 115. Both figures are significantly larger than the corresponding figures for query by boosting.

Although the results in figures 9.26 and 9.27 are independent of each other, the facts that the test sets used both for query by uncertainty and ActiveDecorate are of the same size, and that the two meta learners use the same base learner configuration provide a hint that ActiveDecorate is much slower than query by boosting.

9.4 An active world order

The results reported in this chapter point out query by boosting using MultiBoost with 10 committee members generated by a REPTree base learner, and vote entropy as disagreement metric as the best combination for realizing active document selection in a named entity recognition setting. The seed set used is made up by 5 randomly selected documents, and the test set on which the active learning is evaluated consists of 10 randomly drawn documents. In this section, the results are further analyzed.

Figure 9.28 shows the performance, in F-score, of query by boosting relative to the Boosting baseline. The query by boosting curve is the average of 10 active learning runs, while the baseline is the average of 5 passive runs. Breaking down the F-score into precision and recall, figure 9.29 reveals that the performance gain of active learning compared to the passive baseline is made primarily in terms of increased recall, but that there also is an increase in precision.

The results reported in figure 9.28 can be interpreted in two ways. The first way is that a given classification performance can be reached using less training data in the active learning case, than would be necessary when learning from a randomly selected set of documents form the same corpus. For instance, reaching 78% F-score by means of query by boosting requires the human annotator to mark up named entities in approximately 20 documents, while reaching the same performance in a passive learning setting would require him to annotate approximately 30 documents selected at random from the same corpus. If instead the desired performance is 80% F-score, the human annotator would have to mark up approximately 42 actively selected documents or 50 randomly selected documents. Thus, the difference in classification performance between active and passive learning decreases as the learning process progresses.

The second way to interpret figure 9.28 is that given a fixed amount of data, active learning (almost) always results in classification performance superior to that of learning from randomly selected documents.

9.4.1 Sub-task performance

By analyzing the performance on the named entity recognition sub-tasks, as described in section 8.7.2 it is possible to pin-point where the performance gain of active learning, compared to passive learning, is made. Figure 9.30 shows the F-score for the named entity recognition sub-tasks ENAMEX, TIMEX, and NUMEX as obtained by query by boosting compared to the baseline.

The active learning results for the two largest classes of names, ENAMEX and TIMEX, both improve on the baseline (the characteristics of the data is

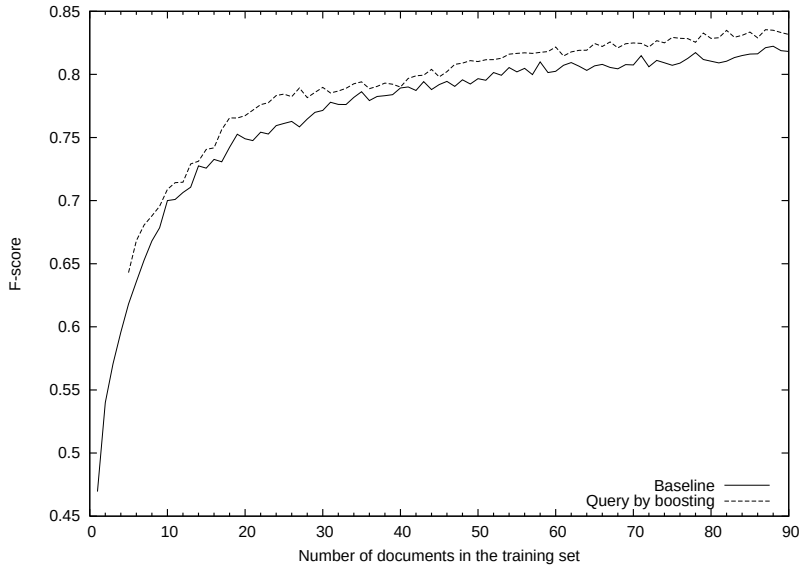

Figure 9.28: The performance of the vote entropy selection metric used with query by boosting on the full corpus.

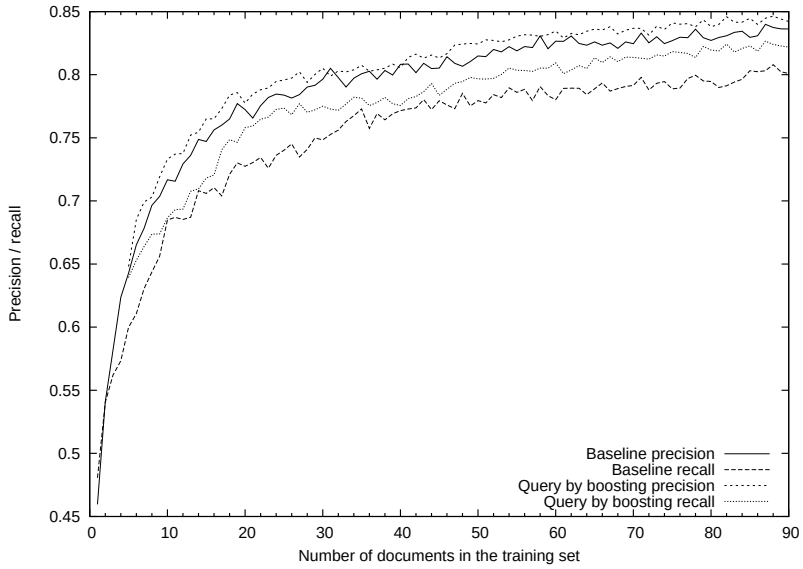

Figure 9.29: A comparison between the precision and recall for query by boosting and the baseline.

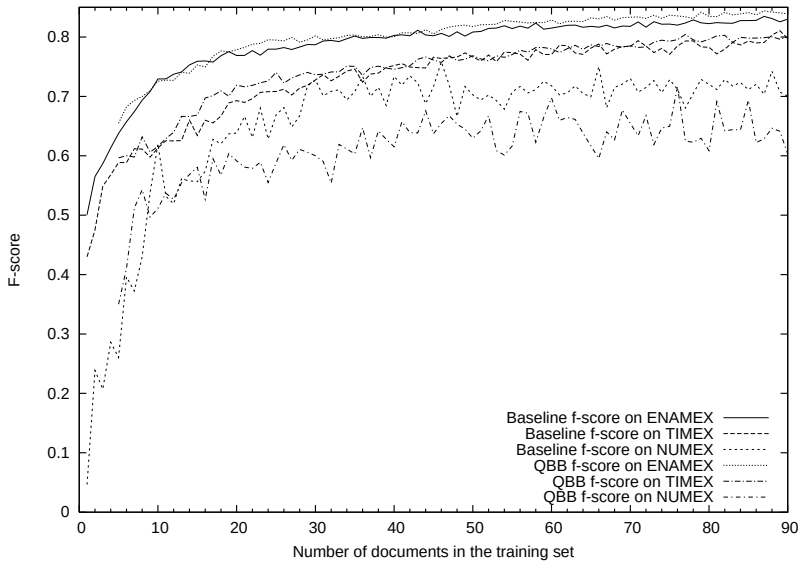

Figure 9.30: Query by boosting versus the baseline performance. Named entity recognition broken down to the three sub-tasks ENAMEX, TIMEX, and NUMEX.

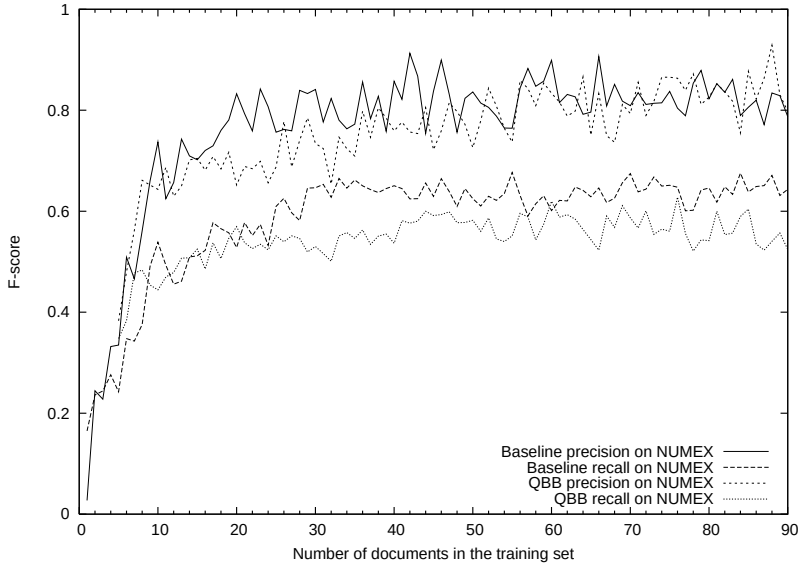

Figure 9.31: The precision and recall obtained by query by boosting for the NUMEX named entity recognition sub-task compared to the results obtained by passively learning the same task.

outlined in section 7.1). In fact, the smallest class of names, NUMEX, proves to be a substantially harder task to solve by means of active learning. Query by boosting fails severely in learning to recognize NUMEX entities, and the results are well inferior to that of the passive Boosting baseline. Figure 9.31 further breaks down the performance on the NUMEX sub-task into precision and recall as obtained by active learning precision and the baseline. Initially, the recall of query by boosting is almost comparable to that of the baseline, but as the learning progresses, the active learning recall diverges from the baseline's recall. The query by boosting precision exhibits the opposite trend; while starting badly, the active learner produces precision better than the baseline's precision after having seen the whole corpus. Thus, the classification performance for a small class suffers from the kind of active learning set-up employed.

9.4.2 Performance variations

The learning curves used for illustration so far are all the average of several runs. As such, they do not tell the whole story. For example, they do not explain the difference between the starting points of the active learning curve with respect to the passive one in figure 9.28. In theory, there should be no difference since the starting points of both curves are the result of randomly selecting documents (see further chapter 10). The explanation for discrepancies like this can be the variation in results that occur as an effect of randomly selecting the test set and the seed set. Figure 9.32 on page 158 shows the standard deviation of the classification performance, in F-score, for the baseline as well as query by boosting. It turns out that the standard deviation of both learning methods overlap more or less throughout the processing of the corpus. Ideally, the active learning curve should exhibit a clear decrease in standard deviation as an effect of the committee of classifiers being trained on increasing amounts of data. While active learning achieve performance superior to that of the passive Boosting baseline, the variance of the active learning results are worse than that of the baseline. The average standard deviation of the F-score achieved by active learning is 2.5% F-score, while the corresponding number for the baseline is 2.35% F-score. Despite the fact that the two curves in figure 9.32 are based on a different number of runs – query by boosting on 10 active learning runs, and the passive baseline on 5 runs – it is possible to observe a tendency of the former to be smoother both in terms of less variation of standard deviation between consecutive documents, and in terms of a more even distribution of the standard deviation. Also, the baseline performance variance shows a tendency to increase from 20 documents and onward, the tendency for query by boosting is rather the opposite.

In the span between the start of the active learning, which in this case is at 5 documents, and the point at which the active learning curve has leveled out, approximately at 40 documents, the difference between the query by boosting curve and the baseline is the greatest, both in terms of performance, and in terms of variation of the performance (see figures 9.28 and 9.32). It is in this span that such a difference is expected to be important to a human annotator since it is here that the contribution of the active learning to the annotation process is most substantial.

In the previous section, the sub-task of recognizing percentages and monetary expressions, NUMEX, is identified as the task which active learning has the most trouble learning. Figure 9.33 on page 158 illustrates the standard deviation of query by boosting versus the passive baseline for NUMEX. Apparently, the active learner not only has troubles learning to solve the task, but it also does it in a way that results in a very large variance of the results.

Figures 9.34 and 9.35 on page 159 show the corresponding graphs for query by boosting and the baseline on the ENAMEX and the TIMEX class, respectively. The average standard deviation of the F-score achieved by query by boosting on the ENAMEX and TIMEX classes is lower than that obtained by passively learning. Query by boosting obtains an average standard deviation of 2.84% F-score for ENAMEX, and 3.22% F-score for TIMEX. The corresponding numbers for the baseline are 2.88% and 4.31%, respectively. In these cases, the standard deviation of the performance of active learning actually does decrease as an effect of more data being available. However, the big spoiler is the average standard deviation of the F-score obtained by query by boosting on the NUMEX class; 19.25% F-score. The baseline manages to cut that in half, and ends up at 10.01%.

9.5 Implications for the BootMark method

What do the results reported in this chapter entail in terms of practical implications for the BootMark method outlined in chapter 6? As a response to the questions posed initially in this chapter in relation to emerging issue E–3, the following holds for the particular task pursued:

- Query by boosting using vote entropy as disagreement metric yields the best classification performance. A second candidate, albeit not at all as good, is ActiveDecorate using Accumulated-logprob for quantifying the informativeness of documents.
- It is enough to use 10 members in the query by boosting decision committee. Employing fewer members yields worse performance, while us-

ing more members increases the training time without increasing the performance correspondingly.

ActiveDecorate appears to be insensitive to the number of committee members used, for good or worse. Further, Decorate more often than not has troubles generating the number of committee members requested.

- As a function of the availability of growing amounts of training data, the time required to train the committee used in query by boosting increases slower in relation to the time required for the committee to classify a fixed size test set, than the corresponding ratio does for ActiveDecorate.

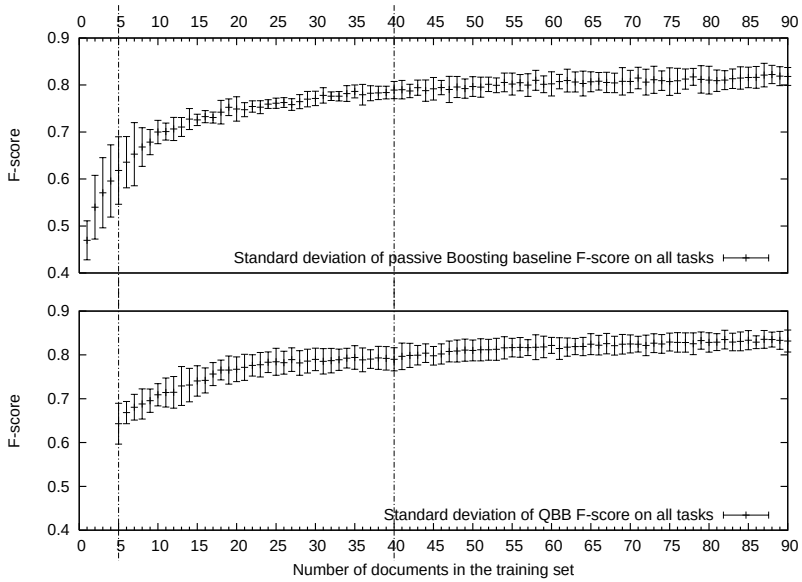

Figure 9.32: The standard deviation of the performance in F-score achieved by the baseline (top graph) and query by boosting (bottom graph). The vertical lines mark the area between the start of the active learning process until the active learning curve levels out. The average standard deviation for the baseline is 2.35% F-score, the average for query by boosting is 2.50% F-score.

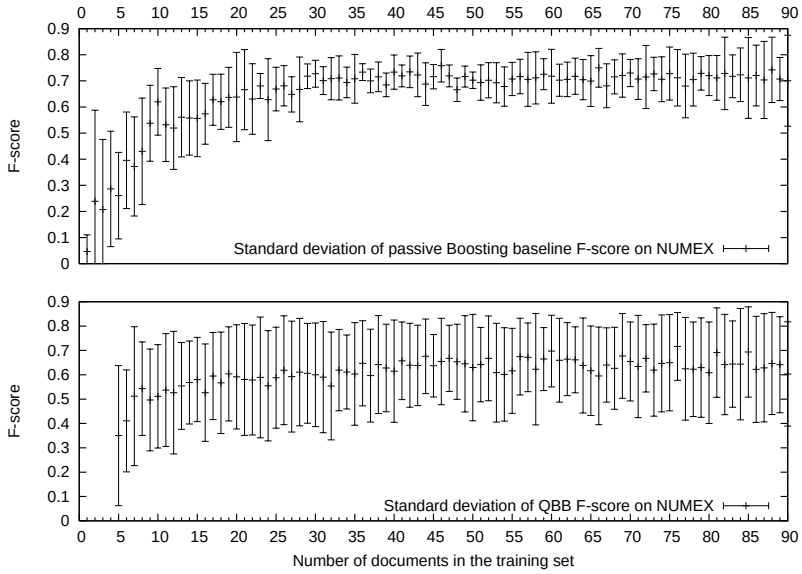

Figure 9.33: The standard deviation of the performance in F-score achieved by the baseline and query by boosting for the NUMEX class of names. The average standard deviation for the baseline is 10.01% F-score, and 19.25% F-score for query by boosting.

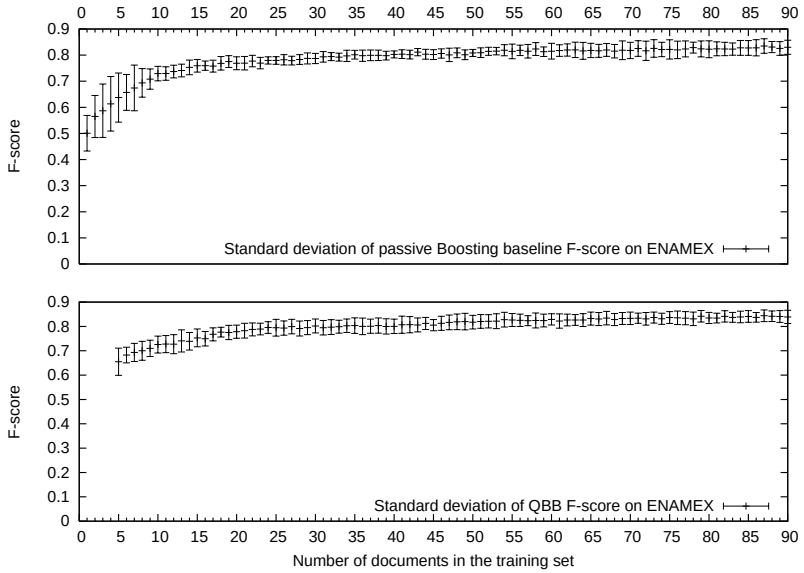

Figure 9.34: The standard deviation of the performance in F-score achieved by the baseline and query by boosting for the ENAMEX class of names. The average standard deviation for the baseline is 2.88% F-score, and 2.84% for query by boosting.

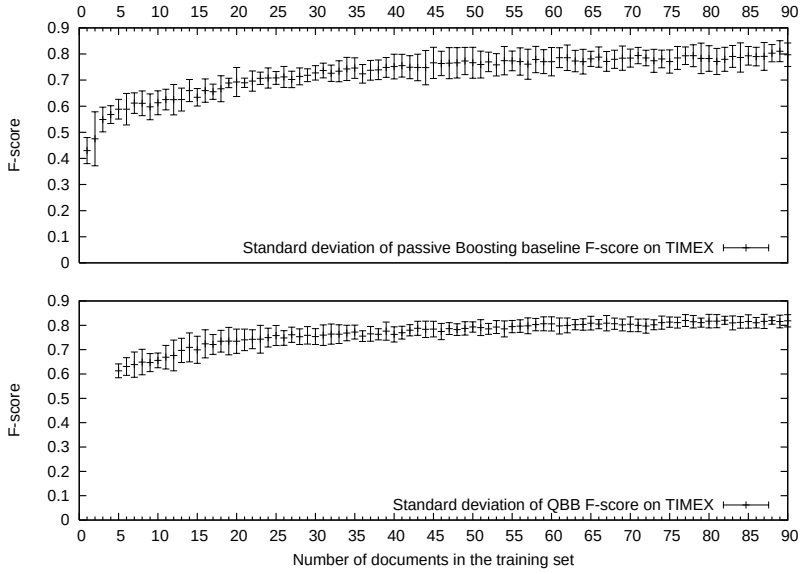

Figure 9.35: The standard deviation of the performance in F-score achieved by the baseline and query by boosting for the ENAMEX class of names. The average standard deviation for the baseline is 4.31% F-score, and 3.22% for query by boosting.

10 SEED SET CONSTITUTION

This chapter investigates emerging issue E–2 in section 6.6 pertaining to the constitution of the set of documents initially used for starting the active learning process; the seed set. The following two questions are addressed:

- How does the number of documents in the seed set affect the learning results?
- Is selecting documents based on clustering better than random selection?

10.1 Seed set size

Assume that the seed set is constructed by randomly selecting documents. Intuitively, an active learning curve should start in the vicinity of the baseline curve, since the starting point of the active curve, and the point at the baseline curve corresponding to the same amount of data, are both the effect of randomness. Following from this, the effect that the size of the seed set might have on the active learning curve should be sought not in terms of differentiated starting points and overall performance with respect to the baseline curve, but rather in terms of the fluctuation of performance. A small seed set is expected to yield a greater initial variance in performance than a larger seed set. Does the size of the seed set have an impact on the standard deviation of active learning results when the learning process proceeds?

Figures 10.1 and 10.2 indeed show that the size of a randomly selected seed set has little effect on the resulting active learning curve. The figures depict query by uncertainty, using the best configuration as reported in section 9.2, while varying the size of the seed set from 1 to 10 randomly selected documents. All active learning curves start near the baseline curve, and they all follow approximately the same trajectory, slightly improving on the baseline.

The effect of the number of randomly selected documents making up the seed set on the standard deviation of learning is illustrated in table 10.1. There

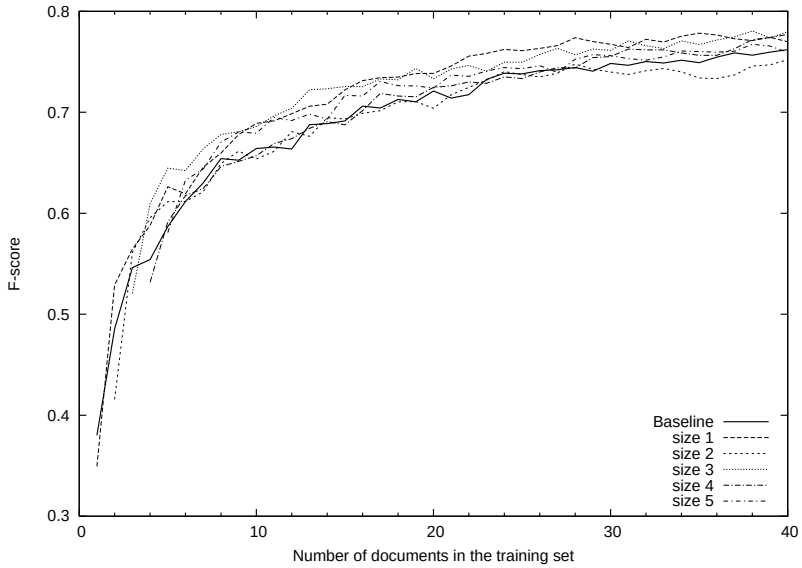

Figure 10.1: Effects of randomly selecting seed sets of sizes 1 to 5 documents for query by uncertainty. Using REPTree as base learner and Log probability as document uncertainty metric.

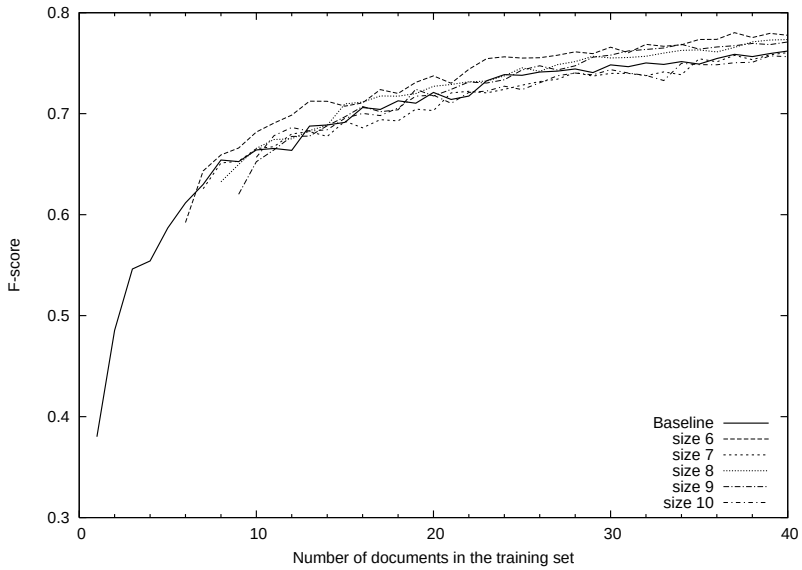

Figure 10.2: Effects of randomly selecting seed sets of sizes 6 to 10 documents for query by uncertainty. Using REPTree as base learner and Log probability as uncertainty metric.

SIZE	SD-INIT	SD-40	SD-AVG
1	0.0426	0.0466	0.0421
2	0.0454	0.0163	0.0207
3	0.0428	0.0099	0.0249
4	0.0564	0.0344	0.0297
5	0.0667	0.0453	0.0413
6	0.0645	0.0371	0.0387
7	0.0113	0.0354	0.0289
8	0.0472	0.0266	0.0364
9	0.0314	0.0373	0.0246
10	0.0465	0.0107	0.0165

Table 10.1: The effect of the seed set size on the standard deviation of performance of query by uncertainty. SIZE is the number of documents randomly selected as the seed set, SD-INIT is the standard deviation of the F-score when using SIZE number of documents, and SD-40 is the standard deviation of the F-score for the curve starting with SIZE documents when 40 documents are available in the training set. SD-AVG is the average standard deviation of the curve starting with SIZE documents.

is no correlation worth mentioning[11] between the size of the seed set, and the standard deviation of classifier performance after training on the seed set, or on 40 documents. The average of the standard deviation is not correlated with the seed set size either.

In figure 10.3, the effects of randomly selected seed sets of sizes one, five, and ten documents are illustrated for the best query by boosting configuration obtained in section 9.3.2. As with query by uncertainty, the learning curves generated by learning on the different seed sets follow the same trajectory. Table 10.2 shows the initial standard deviations of the performance, the standard deviation after training on 30 documents, as well as the average standard deviation for each of the three curves. The average standard deviation of classification performance is clearly affected by the number of documents in the seed set.

SIZE	SD-INIT	SD-30	SD-AVG
1	0.0776	0.0264	0.0290
5	0.0428	0.0101	0.0196
10	0.0230	0.0158	0.0162

Table 10.2: The effect of the seed set size on the standard deviation of performance of query by boosting.

[11] The Spearman ρ correlation is computed, and if a correlation coefficient value less than 0.5 for two variables is obtained, the variables are considered to be uncorrelated.

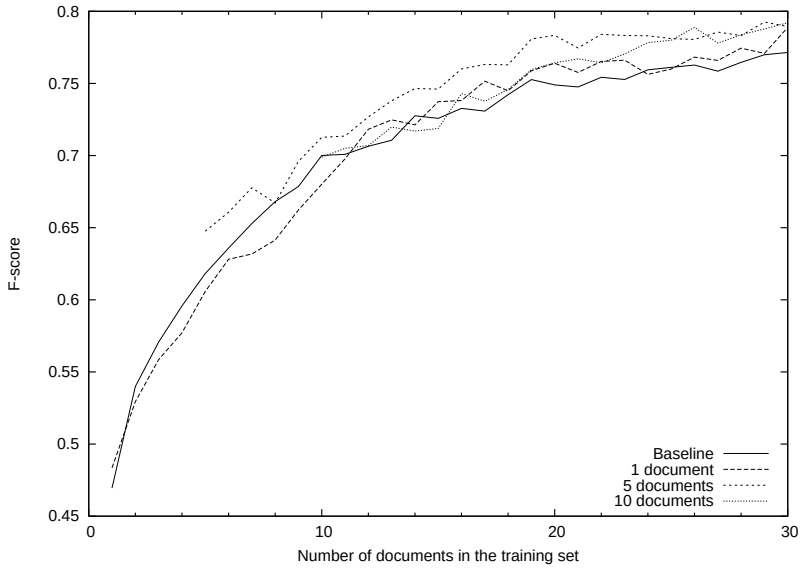

Figure 10.3: Effects of randomly selecting seed sets of sizes 1, 5 and 10 documents on query by boosting. Using REPTree as base learner and Vote entropy as disagreement metric.

Due to its ability to harness small amounts of training data, the performance by query by committee, after having trained on the seed set, exhibits less variance than does query by uncertainty. Overall, both active learning methods exhibit reduced standard deviation of classification after 30 documents, as an effect of larger seed sets.

10.2 Clustering-based versus random selection

Instead of drawing seed set documents at random from the entire document set, the idea pursued here is to cluster the documents into n clusters, and select the documents to include in the seed set of size n from the centers of the clusters. The hope is for the different clusters to contain different distributions of name types. Sampling one document from each cluster would then mean that names with different characteristics are represented in the seed set. The problem is that the names as such are not directly addressable since their whereabouts and appearance are not known. A better seed set can be manifested by an initial classification performance better than that resulting from randomly selecting the seed set, or by a reduced variation of classification results as the learning process proceeds. It should be noted that the approach taken here is not an

exhaustive investigation into the possibilities of clustering-based methods, but merely a first attempt to make more informed, yet automated decisions as to what goes into the seed set.

An algorithm called Simple K-means is employed for clustering the documents in the corpus. The algorithm is described by, for instance, Witten and Frank (2005). Simple K-means is attractive since it allows for deciding on the number of resulting clusters beforehand. It is also fast, which is an advantage since it is used in a way that requires it to be applied repeatedly, as will be explained later. The Weka implementation of Simple K-means requires as input a set of objects to cluster, documents in this particular case, information about how many clusters to create, as well as a random seed. The algorithm is as follows:

1. Based on the random seed, randomly select *K* documents to act as the initial cluster centroids. A centroid can be defined as an (imaginary) document that sits at the center of a cluster.

2. Assign each of the documents in the document set to the cluster that has the closest centroid.

3. When all documents have been assigned to a cluster, re-calculate the positions of the centroid in each cluster.

4. Repeat steps 2 and 3 until the centroids do not move between re-calculations.

Simple K-means is proven to always terminate; however, the constitution of the clusters is sensitive to the initial, random, selection of cluster centers.

Each document used as input to Simple K-means is represented as a feature vector in which the features are lemmatized words and the feature values are the relative weights of the words. The lemmatized version of the corpus is obtained by using EN-FDG (introduced in chapter 7). The weights are calculated based on term frequency and inverse document frequency, as well as filtered by using a Weka built-in English stop word list. The number of features used is constrained to the 1000 most highly ranked lemmas in the corpus.

The output from Simple K-means is information about the sizes of the clusters, the cluster centroids, as well as about the sum of squared errors within each cluster. The sizes of the clusters are given as the percentages of the documents in the original document set belonging to each cluster.

Simple K-means is put to use in two different ways in order to create clusters of the corpus from which to select the seed set. For both ways, the documents closest to the cluster centroids are selected. In the first way, as evenly

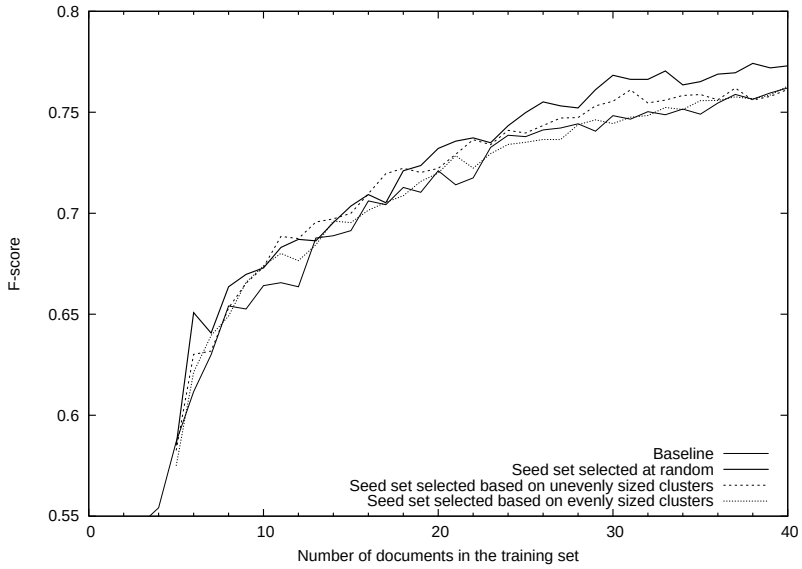

Figure 10.4: Effects of selecting seed set documents based on clustering for query by uncertainty.

sized clusters as possible are created. In the second way, clusters of as uneven size as possible are exploited. The reason to use such orthogonal approaches is to facilitate the verification of any impact that Simple K-means clustering may have on selecting the seed set. Cluster size similarity is quantified as the least standard deviation of the cluster sizes given by the clusterer as percentages. The sensitivity of Simple K-means towards the initial selection of cluster centers is exploited for controlling the sizes of the clusters. In order to obtain evenly sized clusters, Simple K-means is supplied with different seeds for the random selection of the centroids, and the seed yielding the set of clusters that exhibits the smallest standard deviation in cluster size is selected. The same approach is used to obtain unevenly sized clusters, except of course for the fact that the configuration selected is the one exhibiting the largest standard deviation. The seed supplied to Simple K-means is varied between 0 and 60 in increments of one. In the case there is a tie between cluster configurations, the setting which results in the clusters with the least within-cluster sum of squared errors is selected.

Figure 10.4 shows the effect of applying the two clustering approaches to query by uncertainty. As illustrated in the figure, there is no clear difference between the results produced by the two clustering-based seed set selection methods. The figure also suggests that randomly selecting the seed set re-

METHOD	SD-INIT	SD-40	SD-AVG
Random	0.0312	0.0267	0.0268
Even	0.0252	0.0246	0.0235
Uneven	0.0327	0.0335	0.0286

Table 10.3: The effect of the seed set selection method on the standard deviation of performance of query by uncertainty. The METHODs used are randomly selecting 5 documents, selecting them based on evenly sized clusters, or on unevenly sized clusters. SD-INIT is the standard deviation of the performance after having trained on the seed set, SD-40 is the performance after having trained on 40 documents. SD-AVG is the average standard deviation.

sults in performance slightly better than that of either of the two clustering approaches. It appears as if the way of clustering the corpus described above, prior to selecting the seed set, has no effect on the initial performance. The performance after 40 documents indicates that a seed set selected at random is better than the clustering-based approaches.

Table 10.3 shows the standard deviations of the classification performance for the three different seed set selection methods. From the table, it is clear that the clustering method in which the clusters are made as evenly sized as possible results in the least deviation of the performance in terms of the standard deviation when trained on the seed set, when trained on 40 documents, and as the lowest average standard deviation of the three methods. Randomly selecting the seed set is better than using the clustering approach that uses unevenly sized clusters.

Figure 10.5 illustrates the effects of clustering-based seed set selection on query by boosting. Neither of the clustering methods contribute to improved results compared to random selection of the seed set. On the contrary, clustering seems to inhibit classification performance and place it on par with the Boosting baseline. Turning to the performance standard deviation, table 10.4 shows that unlike query by uncertainty, query by boosting does not benefit to the same extent from a seed set selected by clustering. Selecting the seed set

METHOD	SD-INIT	SD-30	SD-AVG
Random	0.0428	0.0101	0.0196
Even	0.0399	0.0201	0.0267
Uneven	0.0324	0.0347	0.0335

Table 10.4: The effect of the seed set selection method on the standard deviation of performance of query by boosting.

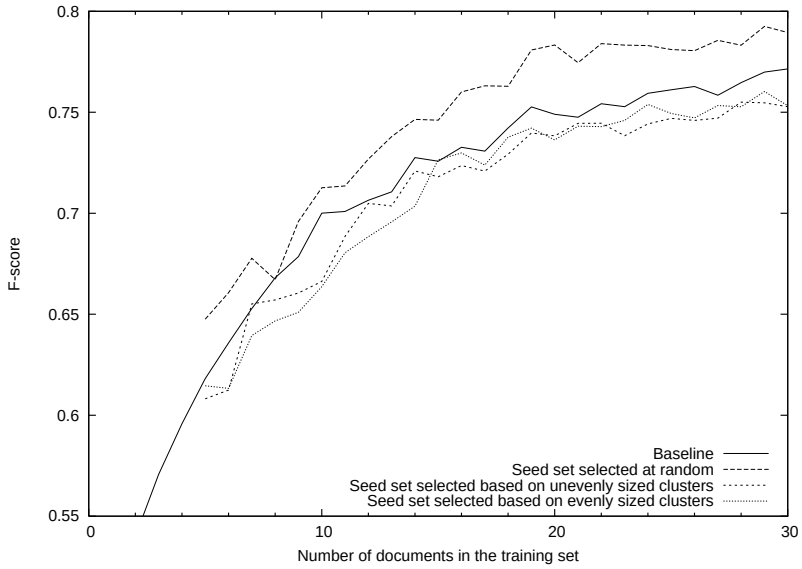

Figure 10.5: Effects of selecting seed set documents based on clustering for query by boosting.

from a document space clustered into evenly sized clusters results in a smaller average standard deviation of performance than does selecting the set from uneven clusters, albeit the initial standard deviation is slightly larger with the former method. What is particularly worth noticing is that while selecting the seed set at random, or from evenly sized clusters, results in a decrease in classification standard deviation as the learning process proceeds, the same effect is not accomplished when the seed set is selected from uneven clusters.

10.3 Implications for the BootMark method

The results reported in this chapter pertain to emerging issue E–2 in section 6.6, and the two questions posed initially are answered in the following.

- Increasing the number of documents in a randomly selected seed set has the obvious effect of increasing initial classification performance. Other than that, neither query by uncertainty, nor query by boosting produces results that vary as an effect of the seed set size.

Increasing the size of the seed set affects the variance of the classification results. Both learning schemes exhibit reductions in the standard deviation of the performance as the sizes of the seed sets increase.

- Generally, selecting documents based on clustering, as it is used here, is not better than randomly selecting the seed set.

None of the learning schemes managed to produce better classification results when seed sets selected based on clustering were used. On the contrary, the results achieved prove to be inferior to those using random seed set, approximately on par with the baselines.

As for the effect of clustering-based seed set selection on the variance of classification performance, query by uncertainty and query by boosting are affected somewhat differently. The smallest average standard deviation achieved for query by uncertainty follows from selecting the seed set from the centroids of evenly sized clusters. For query by boosting, the same effect is achieved using a randomly selected set. Selecting the seed set based on unevenly sized clusters is sub-optimal for both learning schemes.

11 MONITORING AND TERMINATING THE LEARNING PROCESS

This chapter addresses the questions concerning the monitoring and terminating of the active learning process, listed as emerging issue E–4 in section 6.6. The purpose of keeping track of the learning process is to provide the human annotator with means to form a picture of the learning status. Once the status is known, the annotator has the opportunity to act accordingly, for instance, to manually stop the active learning process, or to form an assessment of the quality of the annotations made so far.

The purpose of defining the stopping criterion is slightly different than that of monitoring the learning process. A stopping criterion is used to automatically shift between phases two and three in the BootMark annotation method outlined in chapter 6, and as such it should not hinder nor disturb the human annotator. The questions addressed in this chapter are the following:

- Can the progress of the active selection of documents be monitored and visualized without the use of a designated, held-out, marked-up test set?

- Can the point at which it is appropriate to halt the active selection of documents, and transit between phases two and three in BootMark, be identified?

It should be remembered that there is a readily available way of monitoring the process, and thus also to be able to manually decide when the active learning should be stopped; to use a marked-up, held-out test set on which the learner is evaluated in each iteration. This is the way the learning curves presented so far have been calculated. The drawback of this method is that the user has to manually annotate more data before the annotation process takes off. As such, it clearly counteracts the goal of the BootMark method and should only be considered a last resort.

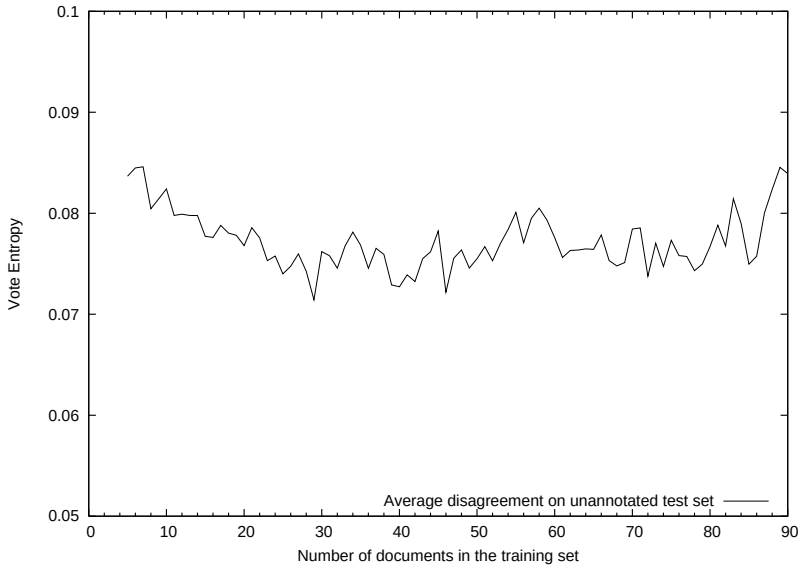

Figure 11.1: An illustration of the change in validation set agreement on a held-out unannotated document set as an effect of annotated, actively selected documents available for training.

11.1 Monitoring as decision support for terminating learning

Tomanek and Hahn (2008) propose to use a held-out *un*annotated set of examples on which the disagreement among the committee members used in a query by committee setting is calculated in each iteration. This is referred to as the validation set agreement. The key to calculating the validation set agreement is that the distribution of instances in the held-out data set (the validation set) is the same as, or very similar to the distribution of instances in the data used for active learning. The purpose of calculating this score is to form a picture of the disagreement between the committee members concerning the set as an effect of the committee being trained on increasingly more, actively selected data. The change in disagreement, Tomanek and Hahn claim, can be used as an indirect approximation of the learning curve that would have been the result of evaluating the committee's performance on a designated and marked-up test set. The underlying assumption is that the agreement within the committee is reflected in its performance. In other words, that a higher agreement concerning the predicted labels also corresponds to higher accuracy of the committee when evaluated on a test set. Tomanek and Hahn (2008) show that their approach works for active learning in a named entity recognition setting in which sentences are selected for annotation.

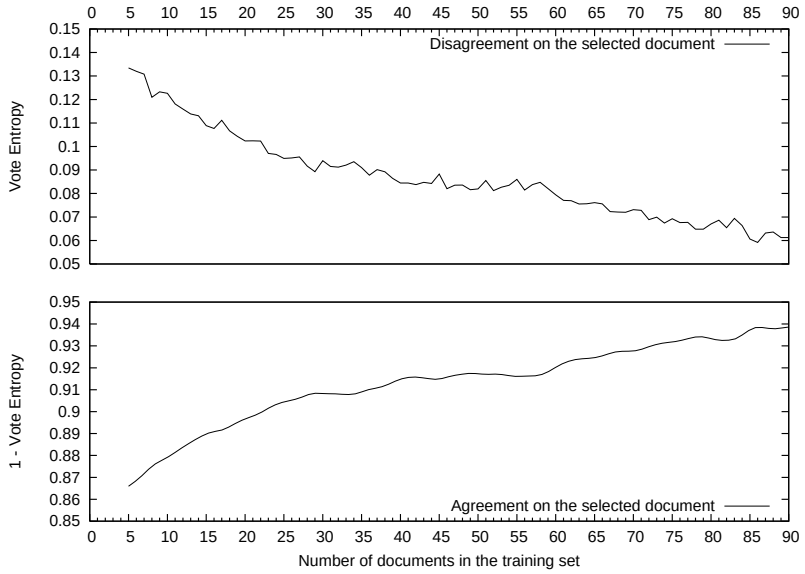

Figure 11.2: The disagreement (Vote entropy) among the members of the committee concerning the most informative document in each active learning iteration (top graph). When the graph is inverted and smoothed using cubic splines, its shape resembles that of a learning curve (bottom graph).

For the purpose of learning how to recognize named entities based on active selection of documents, the validation set agreement curve does not at all resemble the ones presented by Tomanek and Hahn (2008). Figure 11.1 plots the disagreement of the committee members to the number of documents used for training, and it is clear that the resulting graph is not immediately useful for forming a picture of the progression of learning. The reason that the graph in figure 11.1 is so different from those presented by Tomanek and Hahn is because of the granularity of the data for which the disagreement is computed; Tomanek and Hahn calculate the disagreement as the average token vote entropy per sentence, while the current experiment defines disagreement as the average token vote entropy in a document. Admittedly, using the latter definition appears to be too blunt an instrument to properly measure the disagreement for the purpose of approximating the classification performance of the classifier learned in the active learning process.

Another candidate measure is the disagreement among the members of the decision committee concerning the classification of the document selected as the most informative one, that is, the document for which the committee disagrees the most. This measure is introduced by Tomanek, Wermter and Hahn (2007a) as the selection agreement. Figure 11.2 consists of two parts. The top

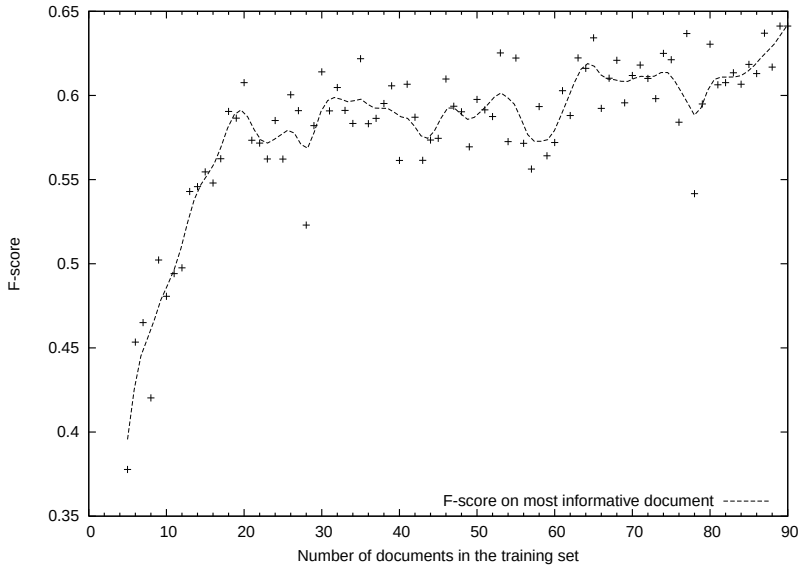

Figure 11.3: The classification performance in F-score on the document deemed the most informative in each active learning iteration. The classification results are illustrated as dots, and the curve is created using cubic splines to get a smoother path.

part illustrates the disagreement, vote entropy, among the members of the decision committee regarding the classification of the most informative document selected in each active learning iteration. In the bottom part of the figure, the disagreement curve has been inverted to depict the agreement, and smoothed using cubic splines.[12] In shape, the curve resembles a learning curve such as the one in figure 9.28, but in figures it does not. The bottom graph in figure 11.2 is as close to the prototypical learning curve shape that can be produced without actually using a designated test set.

A third candidate way to form an image of the progression of learning is that of evaluating the learned named entity recognizer on the document deemed the most informative in each iteration of the active learning process. The performance of the committee of classifiers is calculated *after* the human annotator (which is simulated, see section 9.1) has annotated the document, but *before* it is added to the training set. The result is available in figure 11.3. However, the resulting curve, although smoothed, is too rough to be expected to be of use

[12] A spline is a mathematical way of describing parametric curves used for data smoothing and interpolation. GnuPlot, the program used for constructing the graphs in the thesis, provides a function implementing cubic splines for smoothly connecting data points in a graph.

to a person annotating data; with such an oscillating curve, it is hard to know what is going on until an additional number of active learning iterations have passed. Consider for instance the peak occurring after 18 documents. How is the annotator using this curve as a way of monitoring the learning process for the documents he has annotated to know that it is a peak, and a local one at that, that he is experiencing when he sees the performance after 19, 20 or even 21 active learning iterations? Indeed, the "knee" depicted by the curve approximately around 20–25 documents corresponds to a point on the learning curve obtained by evaluating the committee of classifiers on a held-out, annotated test set where the performance has leveled out (see figure 9.28). This kind of interpretation of the graph is only possible when sufficiently many additional documents have been processed.

11.2 Using committee consensus for terminating learning

The purpose of defining a stopping criterion is to facilitate the automatic transition between the phase in which documents are actively selected for annotation, and the phase in which the learned classifier acts as a pre-tagger (chapter 6). A number of ways to know when to stop active learning have been proposed in the literature, as outlined in section 4.8. There are basically three ways to define a stopping criterion: as a function of the amount of data processed, as an effect of having reached a given classification performance, or based on indications that the active learning process no longer contributes to the overall performance. The two former ways involve deciding, beforehand, on some number of documents, or level of performance at which the learning should be stopped. To make such decisions, a great deal has to be known about the learning process, the domain, and the expected number of documents processed or the performance possible to reach. In defining the BootMark method, it is desirable to refrain from having to introduce arbitrary thresholds to be set by the human annotator, since such thresholds would merely confuse the user. Instead, a dynamic stopping criterion, based on the characteristics of the data at hand, is what is aimed for.

The selection agreement measure proposed by Tomanek, Wermter and Hahn (2007a) is an intuitive criterion for the stopping of active learning; the learning process should be aborted when the active learning no longer contributes to increasing the classification performance. That point in the process is reached when the committee members no longer disagree (or completely agree) on the classification of most informative (disagreed on) item in the remaining unlabeled pool of data.[13] The selection agreement is mainly influenced by two fac-

[13]Depending on the context, the terms *selection agreement* and *selection disagreement* are used to describe the same thing. The disagreement is quantified using the vote entropy metric, while the agreement is 1 – vote entropy, as exemplified in figure 11.2.

tors: the ability of the decision committee to harness the information provided in the informative examples that are labeled by the human annotator and later used for training the classifier; as well as the data available in the diminishing pool of unlabeled data.

In the current experiment set-up, active learning does not contribute to increasing the classification performance when all committee members used in query by boosting agree on the classification of a document from the set of unlabeled documents, that is, when the vote entropy for the most informative document is zero. The top graph in figure 11.2 depicts the vote entropy for the most disagreed on document versus the number of documents in the training set. As is clear from the figure, the vote entropy does not reach zero during the course of processing the 90 unannotated documents in the corpus. This can mean one of two things: either the corpus is too small and active learning actually does contribute to the classification performance after having processed 90 documents, or it might mean that the disagreement will never reach zero. While the former surely is a part of the reason why the selection disagreement does not reach zero, the latter might be a more plausible explanation. For the selection agreement to be complete (when vote entropy is zero), all ten members of the decision committee must assign the same class label to each individual token in the most informative document. Either way, the implication of the graphs in figure 11.2 is that the stopping criterion proposed by Tomanek, Wermter and Hahn (2007a) can neither be confirmed to work in the current experiment set-up, nor put to practical use on its own.

11.3 An intrinsic stopping criterion

When the committee used in active learning for selecting the next document to annotate completely agrees on the proposed classification, it is clear that active learning does not contribute anything more to the learning process than would randomly selecting documents from the remainder of the unannotated corpus. Thus, at that point, the active learning process should be terminated to avoid computational overhead. So, if a completely agreeing committee is a sign that the active learning process *should* be terminated, when is the first point at which there is a balance between the annotation effort required by the annotator and the performance gained, that is, where the learning *may* be terminated without running the risk of losing too much in performance?

Figure 11.4 illustrates the combination of the validation set agreement curve, and the selection agreement curve. That is, the figure shows the committee disagreement concerning a held-out unannotated set of ten documents randomly selected from the original corpus, and the same committee's disagreement re-

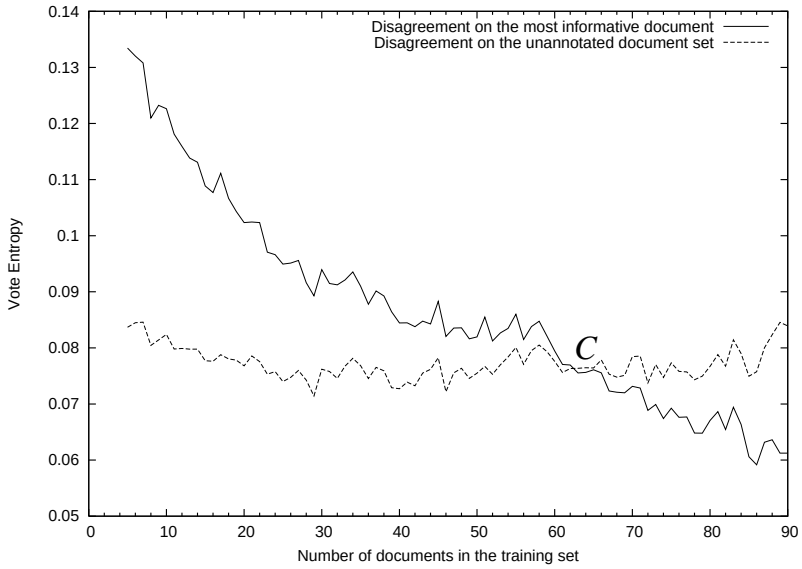

Figure 11.4: The disagreement (vote entropy) among the committee of classifiers on the most informative document versus the disagreement on a held-out unannotated set of documents. The intersection *C* is a candidate point for early stopping of the active learning process.

garding the classification of the most informative document from the remaining unannotated corpus. The results in the figure are the averages of ten runs. A candidate for a point at which the active learning *may* be stopped is the intersection, *C*, between the two disagreement curves in figure 11.4. As the data available for training increases beyond the intersection point of the two curves, the decision committee becomes more in agreement concerning the classification of the most informative document out of the 28 documents that remain in the set of unannotated documents (set *A*), than it is regarding the classification of the ten documents in the held-out unannotated test set (set *B*). This situation implies that the committee of classifiers would learn more from a sufficiently large random sample from a document set with the same distribution as *B*, than it would from actively selecting documents from *A*.

Two cases regarding the distribution of named entities emerge. Assume that the distribution of the classes in set *B* and in the original document set is not the same. Then the intersection point *C* in figure 11.4 is irrelevant for monitoring the learning process, and is hence not a candidate for early stopping. Thus, to be able to deem *C* as a plausible place for early halting of the active learning, more knowledge about the distribution of target classes in set *B* in relation to the original document set is required.

Assume, on the other hand, that the distribution of instances in set *B* is the same as, or very similar to, the original distribution of instances in the unlabeled document set *A*. In this case, the validation set agreement, when calculated on *B*, can be used as an approximation of the validation set agreement of a version of *A* from which no instances has been removed. In effect, *B* is used as a version of *A* which is not affected by the active learning process.

When the intersection between the validation set agreement curve and the selection agreement curve, denoted *C* in figure 11.4, is reached, the performance gain of active learning from the remaining documents in *A* is less than that resulting from learning from random samples from the original distribution *B*. At this point, the question is: Is the overhead involved in pursuing active learning beyond *C* worth the effort, or should the active learning be halted? Although the scope of the thesis does not include issues pertaining to cost models of specific annotation tasks, one thing can be said in favor of halting the learning at *C*: If the agreement concerning the hardest (most informative) example (document) remaining in the unlabeled pool is greater than the average agreement on the classification on all documents in the validation set (set *B*), then it is safe to say that the active learning process has managed to "squeeze" the most out of the original unlabeled pool, and that the steepest part of (a presumed) learning curve has been passed. Thus, at this point, the learning process may be aborted.

Compared to using the selection agreement as sole indicator of an appropriate point at which to stop the active learning, the intersection between the selection agreement curve and the validation set agreement curve represents an *early* stopping criterion for pool-based query by committee. Furthermore, the information required to decide whether the intersection has been reached is *intrinsic* to the data and the learner configuration used; as such, it does not make use of pre-defined thresholds.

Figure 11.5 combines the results of the best active learning setting for selecting documents to annotate (available in figure 9.28 on Page 153) with the validation set agreement curve and the selection agreement curve (shown in figure 11.4). Figure 11.5 shows that the intersection *C* of the latter two curves corresponds to a point at the learning curve at which it has clearly passed the "knee", that is, the steepest part.[14] Thus, under the previously mentioned assumption concerning the distribution of entities in the data sets used, the intersection *C* appears to correspond to a point at which it the active learning process *may* be terminated without running the risk of ending up with a final

[14]It should be noted here that the document set used for calculating the validation set agreement is the same document set that is used for calculating the learning curve. The set consists of ten randomly selected documents, that is, one tenth of the documents available.

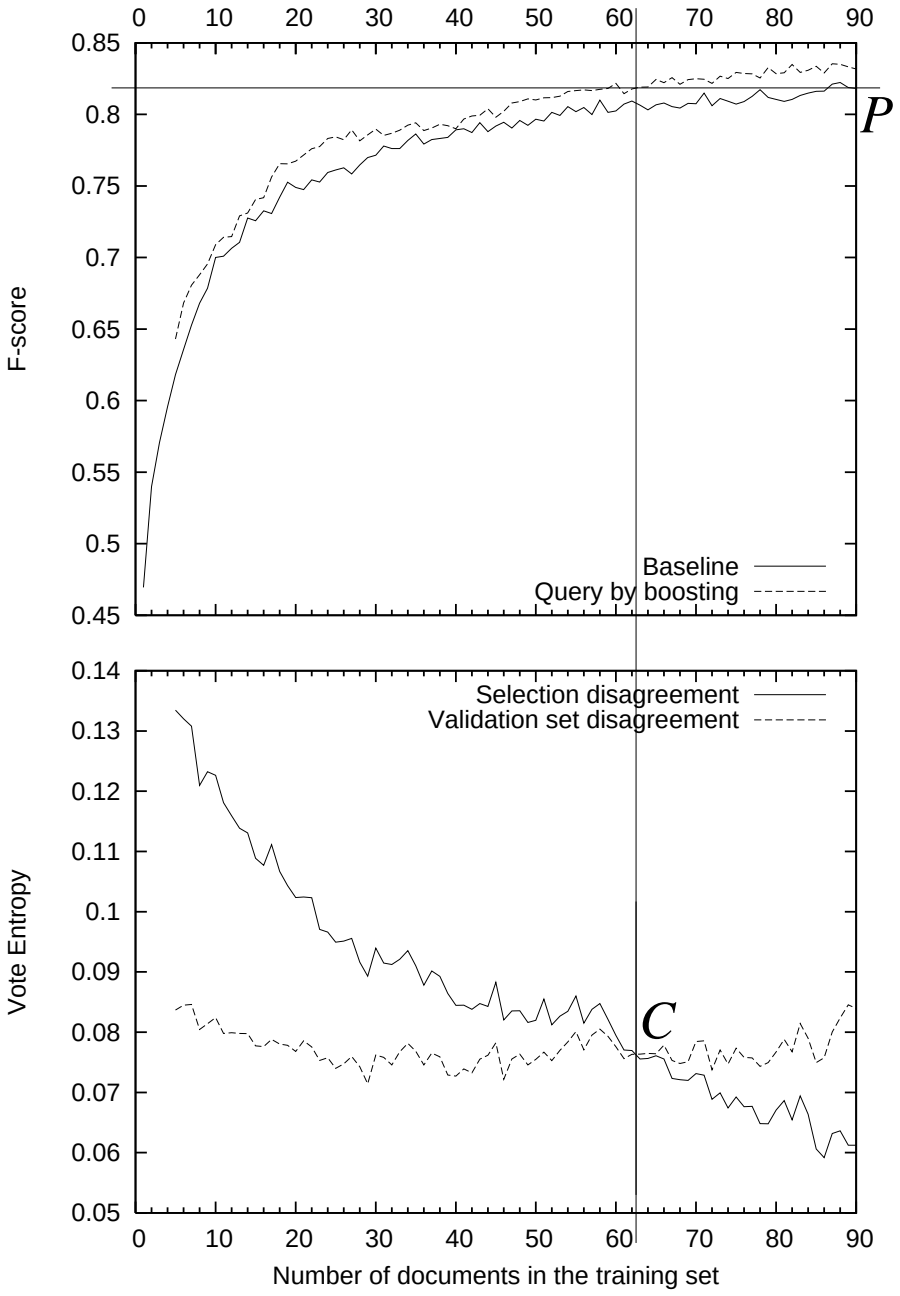

Figure 11.5: In this particular experiment, the intersection between the validation set agreement curve and the selection agreement curve, *C*, corresponds to a point on the active learning curve which indicates that the benefits of active learning has (mostly) been collected.

classifier exhibiting a performance significantly worse than one trained on all data.

In this particular experiment, halting the active selection of documents at *C* results in a classifier achieving roughly the same F-score as a classifier passively trained on all annotated data (as illustrated by *P* in figure 11.5). Given this, the benefits of actively selecting documents to annotate outlined in section 9.4 can now be further clarified: given the experimental set-up used in the thesis, halting the active learning process once the intersection *C* has been reached results in a reduction of the number of documents that require annotation from 90 to 62.

11.4 Implications for the BootMark method

The results reported in this chapter pertain to emerging issue E–4 outlined in section 6.6. The answers to questions posed initially in this chapter follow.

- The progress of the active learning process can be monitored and visualized without using a held-out designated annotated test set. However, the two ways of visualization suggested do not reveal the estimated performance of the classifiers learned.

The first proposed way is to use the (selection) agreement among the classifiers in the committee regarding the classification of the most informative (least agreed on) document in each active learning iteration (figure 11.2).

The second way to visualize the learning process is to plot the selection disagreement (the inversion of the above), as well as the disagreement within the committee regarding the classification of a held-out *un*annotated document set (figure 11.4). Although the graph is not similar to a learning curve, it conveys more information about the current status of learning than figure 11.2, under the assumption that the held-out document set is of the same distribution as the original unlabeled corpus used for active selection of documents.

- The point at which to terminate active learning and transit between Phases Two and Three in BootMark is proposed to be at the intersection between the selection agreement curve and the validation set agreement curve, that is, when the selection agreement is larger than the validation set agreement (see figure 11.5). This approach constitutes a new, intrinsic stopping criterion for committee-based active learning.

Following Tomanek, Wermter and Hahn (2007a), the active learning process *should* be stopped once the selection disagreement among the members of the

decision committee as to the classification of the most informative document has reached zero.

By using a randomly selected, held-out unannotated document set for calculating a reference disagreement curve – validation set agreement – a candidate early point at which the active learning process *may* be stopped is defined as the intersection between the reference disagreement curve and the curve obtained by calculating the disagreement concerning the most informative document remaining in the unlabeled document set, provided that the distribution of classes in the set underlying the reference curve is the same as in the unlabeled document set. When the intersection is reached, the user knows that active learning contributes less to the increase in performance than would a sufficiently large random sample from a document set of the same distribution as the original corpus.

12 On the applicability of pre-tagging with revision

This chapter addresses emerging issue E–5 outlined in section 6.6 pertaining to the plausibility of using an actively learned named entity recognizer as a pre-tagger, as proposed for the third phase of the BootMark method described in section 6.5. The idea of using a pre-tagger is to be able to turn a manual annotation process into one in which the human annotator reviews annotations suggested by the tagger instead of creating annotations from scratch. The aim is to speed up the annotation process, while at the same time maintaining consistent annotation results. In particular, the following questions are discussed:

- What are the requirements on the classifier used as pre-tagger in terms of accuracy?
- Is pre-tagging with revision applicable during the bootstrapping process, that is, is it applicable already in phase two of the BootMark method while the tagger is being trained?

12.1 Pre-tagging requirements

As previously mentioned, there are two strands to using pre-tagging, both of which appear to be valid depending on the outset of the annotation endeavour ahead. The first camp advocates the use of pre-tagging with the argument that using a pre-tagger speeds up the annotation process, since the human annotator is assumed to only have to focus on the system-suggested annotations. The other camp objects to pre-tagging due to the potential bias it introduces. The core of the two arguments is essentially the same; the human annotator is most likely to focus only on the annotations produced by the pre-tagger, which, depending on the quality of the pre-tagging, can be used as an argument for or against pre-tagging as such.

Whether the use of pre-tagging is a problem or not, seems to some extent to depend on the nature of the tagging process proper. If the results of the

pre-tagger include most of the annotations desired (high recall), then consequently the user is able to assess most of the annotations by only taking into consideration those produced by the tagger. If, on the other hand, the results of the pre-tagger are exact (high precision), the user might miss out on plausibly correct annotations if he is only considering the system-suggested ones.

The results of the experiments on actively selecting documents to annotate reported on in section 9.4 demonstrate that while both recall and precision increase, the principal achievement is made in terms of recall. Further, if the second phase of the BootMark method is terminated as is suggested in section 11.2, the resulting actively learned named entity recognizer performs on par with a passively learned recognizer trained on the full corpus. Thus, the active learning process results in a named entity recognizer that is as suitable for pre-tagging as a recognizer obtained the traditional way. In the light of the arguments of pre-tagging pros and cons, this suggests that the use of an actively learned tagger at least is not inimical to the annotation process. However, due to the fact that the human annotator is simulated in the experiments carried out in the present thesis, it is not at this point possible to determine exactly how pre-tagging affects the outcome of the BootMark method. To be able to assess the effects of pre-tagging, a set-up involving real users annotating a corpus for a real named entity recognition task is best used. Thus, the issue of requirements posed on a pre-tagging step used in the BootMark method has to be deferred to future investigations.

12.2 Pre-tagging during bootstrapping

The experiments reported in chapter 9 provide a view on how the errors made by the actively learned classifiers are distributed in a case where the annotator is assumed to be consistent and not to make any mistakes. Figure 12.1 contains information about the matches made by the best query by boosting set-up on the test set, as an effect of the decision committee being trained on an increasing number of documents. There are different kinds of matches possible, as explained in section 8.7.1. Figure 12.1 shows that of the erroneous matches, the number of spurious ones decreases the most, followed by a decrease in number of missing, and partial matches. After 20 documents have been processed, the number of incorrect matches made are approximately constant throughout the remainder of the process. The re-distribution and largest decrease of erroneous matches occur in the first third of the active learning process, in terms of number of documents processed.

Given the re-distribution of errors, and the fact that the benefits of using active learning take place around the point at which the learning curve levels out

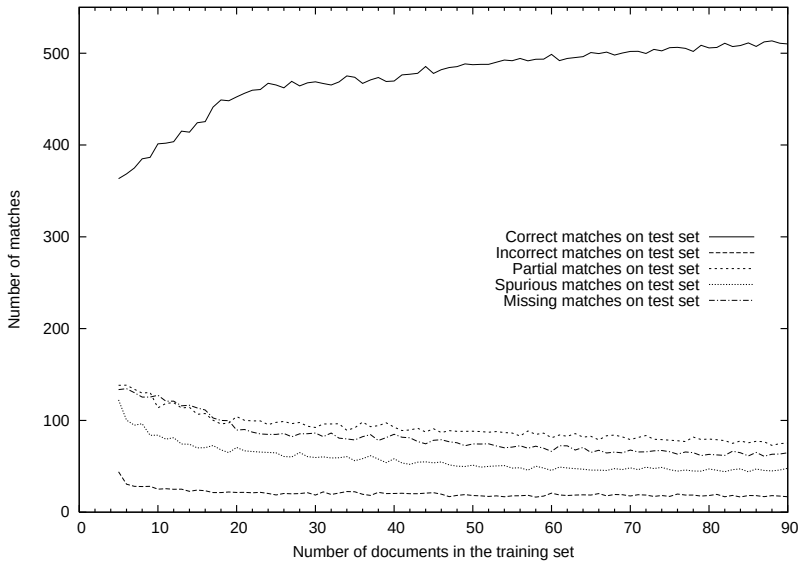

Figure 12.1: The actual number and types of matches made on the test set as an effect of the training data available.

(illustrated in, for instance, figure 9.30), it appears harmful to use the classifier under training as a pre-tagger. There is a possibility that the changes made by the human annotator to the annotations suggested by the system does not change over time. As the annotator finds what he believes to be patterns in the suggestions with respect to the corrections that he should introduce, there is a risk that the user sticks to those patterns; for instance, a given word marked as being a particular type of name might in fact be a spurious match, and should thus not be marked as a name at all. As described above, the learning process involves a re-distribution of the errors made by the classifier. There is a risk that the human pattern matching of errors to appropriate annotations does not keep up with this re-distribution. Thus, unconditional use of the classifier as a pre-tagging may, at this stage, prove volatile to the annotation quality.

Hence, using the classifier as pre-tagger for indiscriminately suggesting labels for *all* tokens in a document during the second phase of the BootMark method should be avoided as long as the “knee” in the learning curve has not been cleared.

If, on the other hand, the classifier can be made to suggest labels for only the tokens of which it is relatively certain, and leave the introduction of labels for uncertain tokens to the human annotator, then pre-tagging may be feasible during the second phase of BootMark too. The problem, then, is to decide

whether the classifier is certain of a predicted label for a token. As in the case of the stopping criterion described in section 11.2, it is desirable to avoid the use of pre-defined thresholds for separating certain tokens from uncertain ones; the user should not be forced to decide on threshold values. Instead, the same sources of information that form the base for the stopping criterion can be utilized in order to judge the label for a token as certain. The following is a sketch of how it can be done: For each token in the document selected by the system for marking up by the human annotator, use the selection metric value (for instance, the vote entropy) assigned to it by the committee of classifiers constructed in the active learning process. Compare the selection metric value with the validation set agreement value obtained for the held-out unannotated test set used for deciding when the learning process should be terminated.[15]

Analogously to the selection agreement used for the stopping criterion, the selection metric value of each token may be used to relate the agreement of the decision committee concerning the predicted label of the token to the average agreement among the committee members regarding the classification of the validation set. Given that the distribution of relevant instances in the validation set (in this case sequences of tokens making up names) is the same as the distribution of instances in the original unlabeled pool of data used for bootstrapping the classifier, we can say something about the informativeness of the current token in relation to all other tokens. If the committee agrees more on the labeling of the current token, than it does on the average labeling of the validation set, then the label of the token can be considered certain, and may thus constitute a suggestion to the human annotator. Conversely, if the decision committee is less in agreement concerning the label of the current token, than it is about the validation set, then the predicted label should not be displayed to the annotator. Thus, making use of the same kind of information as underlies the stopping criterion described in chapter 11, it is possible to decide whether a predicted label for a particular token in a given document should be suggested to the user.

Given the cautious way described above to selectively suggest labels to the user, utilizing the classifier under construction as a pre-tagger during the bootstrapping phase of BootMark appears feasible.

In addition, the selective suggestion of labels addresses the issue of utility of the BootMark method. Serving the user entire unannotated documents instead of smaller chunks of data (sentences, for instance) could be thought of as a drawback in terms of the workload put on the user; the user would have to

[15]Note that the selection metric used for computing the informativeness of the token under scrutiny must be the same selection metric as is used to compute the validation set agreement value; otherwise, the values are incomparable.

mark up the whole document to get to the instances useful to the learning process, while in a smaller context, the user would only have to annotate the immediate surroundings of the informative bits. Even though the description of the BootMark method makes no assumptions about the interaction between the system implementing the method and the human annotator operating it, it is thus possible to supply the technical components necessary for designing the interaction in such a way that there is little difference, from the user's point of view, between using a BootMark-based system that serves the user entire documents to mark up, and approaches utilizing smaller contexts. In either case, it will be possible to direct the user's focus to those tokens marked by the system as uncertain.

12.3 Implications for the BootMark method

The short answers to the questions posed initially in this chapter are:

- It is not possible to decide on the requirements posed on a pre-tagger for the BootMark method without conducting empirical investigations involving real annotators who are working on real annotation tasks.
- Although it is not possible to assess the requirements posed on pre-tagging for the BootMark method, it is safe to say that it is not suitable to *indiscriminately* use pre-tagging with revision during the bootstrapping process unless the benefits of active learning have been collected, that is, the learning process has been terminated. On the other hand, pre-tagging with revision *may* be used during the learning phase if only the certain labels are suggested to the human annotator, and the uncertain ones are not. This chapter outlines a way to do the latter.

Part IV

Finale

13 SUMMARY AND CONCLUSIONS

13.1 Summary

This thesis introduces a method – BootMark – for bootstrapping the marking up of named entities in textual documents and provides in-depth empirical investigations of a number of issues related to its realization.

The work described is rooted in the observation that it is hard to develop, maintain, and adapt information extraction systems, as well as the fact that the methods and tools for addressing these issues more often than not involve machine learning. The quality of the data used for learning puts an upper limit on the performance achievable. Consequently, high quality data is a prerequisite for successfully developing information extraction systems and the corresponding sub-systems.

13.1.1 Part I – Background

The dissertation consists of four parts. Part I provides the background necessary to understand the rest of the thesis. First, named entity recognition is presented, followed by an introduction to machine learning in general, and active machine learning in particular. In active machine learning, the learner is in control of the data from which it learns in such a way that it may ask an oracle, typically a human expert, to correctly classify the examples for which the model learned so far makes the most unreliable predictions. Part I is concluded by an overview of annotation methods that utilize machine learning, including active learning, for assisting the human annotator in marking up text.

13.1.2 Part II – Introducing the BootMark method

In part II, the BootMark method is introduced. BootMark consists of three phases and entails the main contribution of the thesis. The first phase is straight-

forward, aiming to have the human annotator produce a set of correctly annotated documents pertaining to the task of named entity recognition. The second phase constitutes the bootstrapping part. It involves active machine learning for the purpose of selecting which document to annotate next, given the ones already annotated. In phase three, the classifier learned during phases one and two is used in a pre-tagging-with-revision fashion by means of which the remaining unannotated documents of the original corpus are marked up. Along with the description of the BootMark method, a number of emerging issues are identified and described. Their common denominator is that they all depend on the realization of the named entity recognition task to which BootMark is to be applied, and as such, they require the context of a practical setting to be properly addressed.

13.1.3 Part III – Empirically testing the BootMark method

Part III introduces the specifics of the named entity recognition task in which the emerging issues resulting from part II are addressed. Data and task definitions stemming from a leading venue for information extraction – the Message Understanding Conference series – is used throughout the empirical tests of the emerging issues.

13.1.3.1 Issue 1 – Base learner and task characteristics

The first issue subject to investigation in part III is that of deciding on an appropriate learning scheme for named entity recognition. The decisions made include the definition of the learning task and address questions such as data representation and algorithm parameter settings, as well as means to evaluate the learning process.

Nine base learners, in a total of 216 different configurations, are used for this part of the empirical investigation. When weighing together the classification accuracy, training time, and time required to test the classifiers on a token classification task, a REPTree decision tree configuration proves to be the best choice.

13.1.3.2 Issue 2 – Actively selecting documents to annotate

The second issue addressed in part III is that pertaining to the most crucial requirement of the BootMark method, stating that in order for BootMark to be applicable to a task, it must be possible to distinguish between the documents

to annotate by means of active learning. Query by uncertainty, query by boosting, ActiveDecorate, and Co-testing along with a large number of metrics for quantifying uncertainty and committee disagreement are investigated.

The best active learning results are obtained by query by boosting using the above-mentioned REPTree configuration for named entity recognition. The ten members of the decision committee are generated by MultiBoost. Vote entropy turns out to be superior to all other disagreement metrics investigated, including Jensen-Shannon divergence, and various margin-based metrics.

13.1.3.3 Issue 3 – The constitution of the seed set

The third issue subject to investigation in part III is the constitution of the set of documents used to start the BootMark annotation process.

Three ways of assembling the seed set for query by uncertainty, as well as for query by boosting, are examined: random sampling, and two ways of K-means clustering. In all three approaches, the number of documents included in the seed set is varied between one and ten. In the first clustering approach, as evenly sized clusters as possible are produced. The documents closest to the centroids are then selected for inclusion in the seed set. The second approach to clustering exploits as unevenly sized clusters as possible. Here too, the documents in the corpus that are the closest to the cluster centroids are used in the seed set.

The size of the seed set has no impact on the active learning results. Regardless of whether one or ten documents are used as a seed, the active learning curves follow a similar trajectory. Further, random selection of the documents to include in the set is better than to use any of the two K-means clustering setups. Selecting the seed set from unevenly sized clusters has a negative effect on the classification variance; the standard deviation of the results obtained in both query by uncertainty and query by boosting increases.

13.1.3.4 Issue 4 – Monitoring and terminating the learning process

The fourth issue addressed in part III is how to monitor and terminate the active learning process without having to supply an annotated held-out document test set.

It is possible to visualize the progress of the learning process without the use of a designated, annotated, test set. Although the visualization fails to indicate the performance of the classifiers learned, it can be used as an indication of the learning progress.

A novel stopping criterion is proposed. It is based on the intrinsic characteristics of the data and the based learner configuration in use, and is thus not dependent on the user to define thresholds prior to initiating the learning process. The stopping criterion is defined so that the learning should terminate when the selection agreement for the most informative instance from the diminishing unlabeled pool used for active learning is larger than the average agreement concerning the classification of the held out unannotated validation set. This intrinsic stopping criterion is motivated by the fact that after this point in the process, the learner would learn more from a sufficiently large random sample from the validation set, than it would if the examples used for training were drawn from the remainder of the unlabeled pool of data.

13.1.3.5 Issue 5 – Revision of system-suggested annotations

The fifth and final issue addressed in part III concerns the applicability of the actively learned classifier as a pre-tagger.

Overall, it is concluded that this matter is best settled by conducting a user-study involving real annotators working on a real named entity recognition task. However, it is also argued that the classifier learned by means of active learning is as good a pre-tagger as one learned by random sampling if it is used for pre-tagging once the benefits of active learning are collected. That is, the classifier may be used as a pre-tagger once the active learning process has been terminated.

Additionally, a sketch of how the tagger can actually be utilized for pre-tagging *during* the active learning process is presented. It involves suggesting labels for those tokens for which the classifier is certain, and leave the labeling of the uncertain ones to the human annotator. The certain versus uncertain issue can be settled by using the same information that is used for terminating the active learning process.

13.1.4 Part IV – Wrapping up

Finally, part IV comprises the summary you are currently reading, as well as concluding remarks and future directions.

13.2 Conclusions

The thesis (section 1.1) states that the BootMark method requires a human annotator to manually mark-up fewer documents in order to produce a named

entity recognizer with a given performance, than would be needed if the documents forming the base for the recognizer were randomly drawn from the same corpus.

The results reported in chapters 8, 9, 10, and 11 support the thesis: Phase one and phase two of the BootMark method indeed reduce the number of documents a human user has to annotate to facilitate the training of a named entity recognizer competitive or superior to a recognizer trained on randomly selected documents. However, the BootMark method is not at all guaranteed to produce better results than those achieved by random selection followed by manual annotation. The applicability of the proposed bootstrapping method hinges on the ability to distinguish between documents by means of actively learning to classify the named entities contained in them.

13.3 Future directions

While the thesis (section 1.1) is covered by this dissertation, there are a number of interesting questions remaining regarding matters that are not immediately within the scope of the thesis. Hence, this final section serves the purpose of sketching lines along which the current work may be extended.

13.3.1 Further investigating pre-tagging with revision

The questions raised in conjunction with the use of the named entity recognizer as a pre-tagger, discussed in chapter 12, require further investigations to be satisfactorily answered. Thus, it will be necessary to specify and conduct a proper named entity annotation task such as the first one outlined in section 13.3.2, involving real annotators. In the context of such an experiment, the matter of interaction between the annotators and a system implementing the BootMark method will also require attention. In particular, future work should seek to verify the utility of displaying entire documents to the user instead of displaying sentences, which is the usual way in active learning for named entity recognition.

Analogous to the document centered approach outlined by Mikheev (2000), the working hypothesis in an experiment on the matter will be that displaying entire documents, as opposed to sentences, may well facilitate easier disambiguation of otherwise hard instances. Under the assumption that a given name is used to address a particular entity within the scope of a document, the occurrence of several hard examples within a document can be collapsed to one. Also, if the display of entire documents is combined with the way of suggest-

ing labels only for those tokens for which the classifier is certain (proposed in chapter 12), the overhead of using a document-based approach may, from the user's point of view, be reduced to that of displaying sentences.

13.3.2 Other languages, domains and tasks

Another natural extension of the work is to investigate the applicability of BootMark to other languages, domains and tasks.

As previously stated, the applicability of the BootMark method hinges on the possibility to judge one document as more informative than another based on the named entities therein. It is imperative that this constraint permeates any attempts to apply BootMark to languages, domains and tasks other than the ones used in this thesis. Ultimately, the successful realization of BootMark depends on the characteristics of the task and data at hand. The task, its representation, the learning scheme configuration, and the way that active learning is set-up all influence the final result, most likely even more so than the language or domain from which the task and data stem. Thus, while the question of whether BootMark is applicable to a certain language is a perfectly reasonable one, its answer depends on the specifics of the given data and task, and the answer thus cannot, with certainty, be generalized to cover all tasks and all data in that particular language. Of course, the same is true for questions regarding the issues of the applicability of BootMark to other domains and tasks as well. In other words, the realization of the BootMark method is so dependent on the data and task at hand that a simple "yes" or "no" as answer to the question "Is BootMark applicable to Swedish?" does not mean anything unless the answer is also qualified by the details of the choices made in the implementation of the issues touched on in this thesis. Hence, it does not make much sense to investigate the applicability of BootMark to languages, domains or tasks in isolation. That said, combinations of new languages, domains and tasks on which I would like to try BootMark include:

- Anonymization of Swedish medical records. The task in this setting is to recognize named entities, mostly person names, in Swedish texts. The outcome would be a corpus of medical records in which the names of persons have been removed. Thus, the task is similar to the one pursued in the thesis, while the domain and language are not.

- Co-reference annotation in English news wire texts. The purpose is to investigate whether BootMark can be applied to a task which is different from that of named entity recognition, but still important to the realization of information extraction systems. The outcome would be a corpus

annotated with co-reference information. In this case, the task is different to the one in the thesis, while the domain and language are not.

If the application of BootMark to the above combinations prove successful, the realization of BootMark using cost sensitive active learning, as described in section 13.3.3, is next on the agenda.

13.3.3 From exploitation to exploration

Active learning, as it is used here, exploits the documents processed so far to facilitate informed decisions regarding which document to process next. What if the bootstrapping phase of BootMark would incorporate means to make the annotation process sensitive to the curiosity of an external source, such as the oracle? The annotation process should not be guided solely by the search for the most informative document to annotate, but should allow for a change of focus in mid-process. The realization of this kind of cost-sensitive active learning could facilitate the transformation from exploitation to active learning as exploration. An immediate practical use for cost sensitive – exploratory – active learning is to "unskew" the distribution of named entities in an otherwise severely skewed corpus by sampling from the corpus in such a way that the name distributions appear similar to the learner. The purpose would be to learn a named entity recognizer that is equally well equipped to handle names of different types. Such a recognizer would be more suitable to use as a pre-tagger in the final phase of BootMark due to its supposed ability to annotate documents more consistently.

Cost sensitive active learning operating at the document level would open up for a range of interesting applications. For instance, the support of annotation for several tasks simultaneously such as named entity recognition, the annotation of relations between entities, and co-reference resolution. By allowing the different tasks to take precedence at different times, that is, to allow for the switching of focus during the annotation process, the resulting annotated corpus can be made to contain controlled distributions of the various target concepts.

13.3.4 The intrinsic stopping criterion

Although the novel stopping criterion for pool-based query by committee presented in chapter 11 appears to work for the task of actively selecting documents to annotate with named entities, its applicability to other active learning settings remains an issue for future investigations. For instance, the selection

agreement and validation set agreement curves obtained in "ordinary" active learning for the purpose of creating named entity recognizers will be examined to see if they exhibit the desired characteristics.

An initial investigation of the applicability of the intrinsic stopping criterion to additional active learning scenarios is presented by Olsson and Tomanek (2008).

REFERENCES

Abe, Naoki and Hiroshi Mamitsuka 1998. Query learning strategies using boosting and bagging. *Proceedings of the Fifteenth International Conference on Machine Learning*, 1–9. Madison, Wisconsin, USA: Morgan Kaufmann Publishers Inc.

Aha, David W., Dennis Kibler and Marc K. Albert 1991. Instance-based learning algorithms. *Machine Learning* 6 (1): 37–66 (January).

Alias-I 2008. LingPipe. URL: <http://alias-i.com/lingpipe/>.

Angluin, Dana 1988. Queries and concept learning. *Machine Learning* 2 (4): 319–342.

Appelt, Douglas E. and David J. Israel 1999. Introduction to Information Extraction Technology. A tutorial prepared for IJCAI-99.

Argamon-Engelson, Shlomo and Ido Dagan 1999. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research* 11: 335–360.

Asuncion, Arthur and David Newman 2007. UCI Machine Learning Repository. URL: <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Balcan, Maria-Florina, Avrim Blum and Ke Yang 2005. Co-training and expansion: Towards bridging theory and practice. *Advances in neural information processing systems 17*, 89–96. Cambridge, Massachusetts, USA: MIT Press.

Baldridge, Jason and Miles Osborne 2004. Active learning and the total cost of annotation. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 9–16. ACL, Barcelona, Spain.

Baram, Yoram, Ran El-Yaniv and Kobi Luz 2004. Online choice of active learning algorithms. *Journal of Machine Learning Research* 5 (December): 255–291.

Bauer, Eric and Ron Kohavi 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning* 36 (1-2): 105–139 (July).

Becker, Markus, Ben Hachey, Beatrice Alex and Claire Grover 2005. Optimising selective sampling for bootstrapping named entity recognition.

Stefan Rüping and Tobias Scheffer (eds), *Proceedings of the ICML 2005 Workshop on Learning with Multiple Views*, 5–11. Bonn, Germany.

Becker, Markus and Miles Osborne 2005. A two-stage method for active learning of statistical grammars. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 991–996. Edinburgh, Scotland, UK: Professional Book Center.

Bikel, Daniel M., Richard Schwartz and Ralph M. Weischedel 1999. An algorithm that learns what's in a name. *Machine Learning* 34 (1-3): 211–231 (February).

Blum, Avrim and Tom Mitchell 1998. Combining labeled and unlabeled data with co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory*, 92–100. ACM, Madison, Wisconsin, USA.

Borin, Lars, Dimitrios Kokkinakis and Leif-Jöran Olsson 2007. Naming the past: Named entity recognition and animacy recognition in 19th century Swedish literature. *Proceedings of the workshop: Language Technology for Cultural Heritage Data (LaTeCh), held in conjunction with the 45th Annual Meeting of the Association for Computational Linguistics*. Prague, Czech Republic.

Borthwick, Andrew, John Sterling, Eugene Agichtein and Ralph Grishman 1998. NYU: Description of the MENE Named Entity System as used in MUC-7. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia, USA.

Brants, Thorsten and Oliver Plaehn 2000. Interactive corpus annotation. *Proceedings of the 2nd International Conference on Language Resources and Engineering*, 453–459. ELRA, Athens, Greece.

Breiman, Leo 1996. Bagging predictors. *Machine Learning* 24 (2): 123–140 (August).

Brinker, Klaus 2003. Incorporating diversity in active learning with support vector machines. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, 59–66. Washington DC, USA: AAAI Press.

Carreras, Xavier, Lluís Màrquez and Lluís Padró 2003. Learning a perceptron-based named entity chunker via online recognition feedback. *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003)*, 156–159. Edmonton, Alberta, Canada.

le Cessie, Saskia and Hans C. van Houwelingen 1992. Ridge estimators in logistic regression. *Applied Statistics* 41 (1): 191–201.

Chan, Yee Seng and Hwee Tou Ng 2007. Domain adaptation with active learning for word sense disambiguation. *Proceedings of the 45th Annual Meet-*

ing of the Association for Computational Linguistics (ACL-07), 49–56. ACL, Prague, Czech Republic.

Chapelle, Oliver, Bernhard Schölkopf and Alexander Zien (eds) 2006. *Semi-supervised learning*. MIT Press.

Chawla, Nitesh V. and Grigoris Karakoulas 2005. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research* 23 (March): 331–366.

Chen, Jinying, Andrew Schein, Lyle Ungar and Martha Palmer 2006. An empirical study of the behavior of active learning for word sense disambiguation. *Proceedings of the Human Language Technology Conference - North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL 2006)*, 120–127. ACL, New York, New York, USA.

Chinchor, Nancy 1998. Overview of MUC-7/MET-2. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia, USA.

Chinchor, Nancy, Patty Robinson and Erica Brown 1998. HUB-4 named entity task definition. Technical report, SAIC.

Chklovski, Timothy and Rada Mihalcea 2002. Building a sense tagged corpus with open mind word expert. *Proceedings of the SIGLEX/SENSEVAL Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, 116–122. ACL, Philadelphia, Pennsylvania, USA.

Chou, Wen-Chi, Richard Tzong-Han Tsai, Ying-Shan Su, Wei Ku, Ting-Yi Sung and Wen-Lian Hsu 2006. A semi-automatic method for annotating a biomedical proposition bank. *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora*, 5–12. ACL, Sydney, Australia.

Ciravegna, Fabio, Alexei Dingli, Daniela Petrelli and Yorick Wilks 2002. Timely and non-intrusive active document annotation via adaptive information extraction. *Proceedings of the ECAI Workshop on Semantic Authoring, Annotation & Knowledge Markup (SAAKM-02)*. Lyon, France.

Ciravegna, Fabio, Daniela Petrelli and Yorick Wilks 2002. User-system cooperation in document annotation based on information extraction. *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002)*. Siguenza, Spain: Springer Verlag.

Cohen, William W. 1995. Fast effective rule induction. *Proceedings of the 12th International Conference on Machine Learning*, 115–123. Tahoe City, California, USA: Morgan Kaufmann.

Cohn, David, Les Atlas and Richard Ladner 1994. Improving generalization with active learning. *Machine Learning* 15 (2): 201–221 (May).

Collier, Nigel, Hyun Seok Park, Norihiro Ogata, Yuka Tateishi, Chikashi Nobata, Tomoko Ohta, Tateshi Sekimizu, Hisao Imai, Katsutoshi Ibushi and Jun-ichi Tsujii 1999. The GENIA project: Corpus-based knowledge acquisition and information extraction from genome research papers. *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 271–272.

Collins, Michael and Yoram Singer 1999. Unsupervised models for named entity classification. *Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 100–110. ACL, University of Maryland, College Park, Maryland, USA.

Cowie, Jim and Wendy Lehnert 1996. Information extraction. *Communications of the ACM* 39 (1): 80–91 (January).

Culotta, Aron, Trausti Kristjansson, Andrew McCallum and Paul Viola 2006. Corrective feedback and persistent learning for information extraction. *Journal of Artificial Intelligence* 170 (14): 1101–1122 (October).

Daelemans, Walter and Véronique Hoste 2002. Evaluation of machine learning methods for natural language processing tasks. *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, 755–760. ELRA, Las Palmas, Gran Canaria, Spain.

Dagan, Ido and Sean P. Engelson 1995. Committee-based sampling for training probabilistic classifiers. *Proceedings of the Twelfth International Conference on Machine Learning*, 150–157. Tahoe City, California, USA: Morgan Kaufmann.

Day, David, John Aberdeen, Sasha Caskey, Lynette Hirschman, Patricia Robinson and Marc Vilain 1998. Alembic workbench corpus development tool. *Proceedings of the 1st international conference on language resource and evaluation*, 1021–1028. ELRA, Granada, Spain.

Day, David, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson and Marc Vilain 1997. Mixed-initiative development of language processing systems. *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 438–355. ACL, Washington DC, USA.

Dempster, Arthur, Nan Laird and Donald Rubin 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society* 39 (1): 1–38.

Doddington, George, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel and Ralph Weischedel 2004. The automatic content

extraction (ACE) program – tasks, data, & evaluation. *Proceedings of the 4th International Conference on Language Resources and Evaluation*, 837–840. ELRA, Lisbon, Portugal.

Domingos, Pedro 2000. A unified bias-variance decomposition and its applications. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, 231–238. Stanford University, California, USA.

Douglas, Shona 2003. Active learning for classifying phone sequences from unsupervised phonotactic models. *Proceedings of Human Language Technology Conference – North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL 2003)*, 19–21. ACL, Edmonton, Alberta, Canada.

Engelson, Sean P. and Ido Dagan 1996. Minimizing manual annotation cost in supervised training from corpora. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 319–326. ACL, Santa Cruz, California, USA.

Finn, Aidan and Nicolas Kushmerick 2003. Active learning selection strategies for information extraction. *Proceedings of the International Workshop on Adaptive Text Extraction and Mining (ATEM-03)*, 18–25. Catvat, Dubrovnik, Croatia.

Frank, Eibe and Ian H. Witten 1998. Generating accurate rule sets without global optimization. *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, 144–151. Madison, Wisconsin, USA: Morgan Kaufman Publishers.

Freund, Yoav and Robert E. Schapire 1996. Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)*, 148–156. Bari, Italy: Morgan Kaufmann.

Freund, Yoav and Robert E. Schapire 1997. A decision-theoretic generalization of on-line learning and application to boosting. *Journal of Computer and Systems Science* 55 (1): 119–139 (August).

Freund, Yoav, Sebastian H. Seung, Eli Shamir and Naftali Tishby 1997. Selective sampling using the query by committee algorithm. *Machine Learning* 28 (2-3): 133–168 (August/September).

Ganchev, Kuzman, Fernando Pereira and Mark Mandel 2007. Semi-automated named entity annotation. *Proceedings of the Linguistic Annotation Workshop*, 53–56. ACL, Prague, Czech Republic.

Goldman, Sally A. and Yan Zhou 2000. Enhancing supervised learning with unlabeled data. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, 327–334. Stanford, California, USA.

Grishman, Ralph 1997. Information extraction: Techniques and challenges. Maria Teresa Pazienza (ed.), *Information extraction: A multidisciplinary approach to an emerging information technology*, Volume 1299 of *Lecture Notes in Artificial Intelligence*, 10–27. Springer.

Grishman, Ralph, Ted Dunning, Jamie Callan, Bill Caid, Jim Cowie, Louise Guthrie, Jerry Hobbs, Paul Jacobs, Matt Mettler, Bill Ogden, Bev Schwartz, Ira Sider and Ralph Weischedel 1997. *Tipster text phase II architecture design. version 2.3*. New York, New York, USA.

Grishman, Ralph and Beth Sundheim 1996. Message understanding conference-6: A brief history. *Proceedings of the 16th conference on Computational Linguistics*, 466–471. ACL, Copenhagen, Denmark.

Hachey, Ben, Beatrice Alex and Markus Becker 2005. Investigating the effects of selective sampling on the annotation task. *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, 144–151. ACL, Ann Arbor, Michigan, USA.

Haertel, Robbie, Eric Ringger, Kevin Seppi, James Carroll and Peter McClanahan 2008. Assessing the costs of sampling methods in active learning for annotation. *Proceedings of the 46th annual meeting of the association for computational linguistics: Human language technologies, short papers (companion volume)*, 65–68. ACL, Columbus, Ohio, USA.

Hall, Mark A. 1999. Correlation-based feature subset selection for machine learning. Ph.D. diss., Department of Computer Science, University of Waikato, Hamilton, New Zealand.

Hall, Mark A. and Geoffrey Holmes 2003. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering* 15 (6): 1437–1447 (November).

Hamming, Richard W. 1950. Error detecting and error correcting codes. *Bell System Technical Journal* 26 (2): 147–160 (April).

Hoi, Steven C. H., Rong Jin and Michael R. Lyu 2006. Large-scale text categorization by batch mode active learning. *Proceedings of the 15th International World Wide Web Conference (WWW 2006)*, 633–642. Edinburgh, Scotland.

Hwa, Rebecca 2000. Sample selection for statistical grammar induction. *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 45–52. ACL, Hong-Kong.

Hwa, Rebecca, Miles Osborne, Anoop Sarkar and Mark Steedman 2003. Corrected co-training for statistical parsers. *Proceedings of the Workshop on*

the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining. Washington DC, USA.

John, George H. and Pat Langley 1995. Estimating continuous distributions in bayesian classifiers. *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, 338–345. Montreal, Quebec, Canada: Morgan Kaufman.

Jones, Rosie, Rayid Ghani, Tom Mitchell and Ellen Riloff 2003. Active learning for information extraction with multiple view feature sets. *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*. Washington DC, USA.

Kaiser, Katharina and Silvia Miksch 2005. Information extraction – a survey. Technical Report Asgaard-TR-2005-6, Vienna University of Technology, Institute of Software Technology & Interactive Systems, Vienna, Austria.

Kim, Seokhwan, Yu Song, Kyungduk Kim, Jeong-Won Cha and Gary Geunbae Lee 2006. MMR-based active machine learning for bio named entity recognition. *Proceedings of the Human Language Technology Conference – North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL 2006)*, 69–72. ACL, New York, New York, USA.

Kokkinakis, Dimitrios 2004. Reducing the effect of name explosion. *Proceedings of the workshop: Beyond Named Entity Recognition, Semantic Labelling for NLP Tasks, held in conjunction with the 4th International Conference on Language Resources and Evaluation*, 1–6. Lisbon, Portugal.

Körner, Christine and Stefan Wrobel 2006. Multi-class ensemble-based active learning. *Proceedings of The 17th European Conference on Machine Learning and the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 687–694. Berlin, Germany: Springer-Verlag.

Kuo, Jin-Shea, Haizhou Li and Ying-Kuei Yang 2006. Learning transliteration lexicons from the web. *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association of Computational Linguistics*, 1129–1136. ACL, Sydney, Australia.

Lafferty, John, Andrew McCallum and Fernando Pereira 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, 282–289. Williamstown, Massachusetts, USA.

Laws, Florian and Hinrich Schütze 2008. Stopping criteria for active learning of named entity recognition. *Proceedings of the 22nd International Con-*

ference on Computational Linguistics (COLING 2008), 465–472. ACL, Manchester, England.

Lewis, David D. 1995. A sequential algorithm for training text classifiers: Corrigendum and additional data. *ACM SIGIR Forum* 29 (2): 13–19.

Lewis, David D. and William A. Gale 1994. A Sequential Algorithm for Training Text Classifiers. *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 3–12. Dublin, Ireland: ACM/Springer.

Liere, Ray and Prasad Tadepalli 1997. Active learning with committees for text categorization. *Proceedings of the fourteenth national conference on artificial intelligence*, 591–597. AAAI, Providence, Rhode Island, USA.

Lin, Jianhua 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37 (1): 145–151 (January).

Linguistic Data Consortium 2001. Message Understanding Conference (MUC) 7. LDC2001T02. FTP FILE. Philadelphia: Linguistic Data Consortium.

Linguistic Data Consortium 2008. Automatic Content Extraction (ACE). URL <http://projects.ldc.upenn.edu/ace/>.

Liu, H. and R. Setiono 1996. A probabilistic approach to feature selection – a filter solution. *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)*, 319–327. Bari, Italy: Morgan Kaufmann.

Marcus, M., B. Santorini and M. A. Marcinkiewicz 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics* 19 (2): 313–330 (June).

McCallum, Andrew, Dayne Freitag and Fernando Pereira 2000. Maximum entropy Markov models for information extraction and segmentation. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, 591–598. Stanford University, California, USA.

McCallum, Andrew and Wei Li 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003)*, 188–191. ACL, Edmonton, Alberta, Canada.

McCallum, Andrew and Kamal Nigam 1998. Employing em and pool-based active learning for text classification. *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, 350–358. Madison, Wisconsin, USA: Morgan Kaufmann.

Melville, Prem and Raymond J. Mooney 2003. Constructing diverse classifier ensembles using artificial training examples. *Proceedings of the Eigh-*

teenth International Joint Conference on Artificial Intelligence (IJCAI-03), 505–510. Acapulco, Mexico.

Melville, Prem and Raymond J. Mooney 2004. Diverse ensembles for active learning. *Proceedings of the 21st International Conference on Machine Learning (ICML-2004)*, 584–591. Banff, Canada.

Mihalcea, Rada and Timothy Chklovski 2003. Open mind word expert: Creating large annotated data collections with web user's help. *Proceedings of the EACL 2003 Workshop on Linguistically Annotated Corpora (LINC 2003)*. EACL, Budapest, Hungary.

Mikheev, Andrei 2000. Document centered approach to text normalization. *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 136–143. ACM, Athens, Greece.

Mitchell, Tom 1997. *Machine learning*. McGraw-Hill.

Morton, Thomas and Jeremy LaCivita 2003. Wordfreak: An open tool for linguistic annotation. *Proceedings of the Human Language Technology Conference – North American Chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL 2003)*, 17–18. ACL, Edmonton, Alberta, Canada.

Muslea, Ion, Steven Minton and Craig A. Knoblock 2000. Selective sampling with redundant views. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-2000)*, 621–626. Austin, Texas, USA.

Muslea, Ion, Steven Minton and Craig A. Knoblock 2002a. Adaptive view validation: A first step towards automatic view detection. *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, 443–450. Sydney, Australia.

Muslea, Ion, Steven Minton and Craig A. Knoblock 2002b. Active + semi-supervised learning = robust multi-view learning. *Proceedings of the 19th International Conference on Machine Learning (ICML-02)*, 435–442. Sydney, Australia.

Muslea, Ion, Steven Minton and Craig A. Knoblock 2006. Active learning with multiple views. *Journal of Artificial Intelligence Research* 27 (October): 203–233.

Nadeau, David and Satoshi Sekine 2007. A survey of named entity recognition and classification. *Journal of Linguisticae Investigationes* 30 (1): 3–26 (September).

Ngai, Grace and David Yarowsky 2000. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. *Proceedings of*

the 38th Annual Meeting on Association for Computational Linguistics, 117–125. ACL, Hong-Kong.

Nigam, Kamal and Rayid Ghani 2000. Analyzing the effectiveness and applicability of co-training. *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM 2000)*, 86–93. ACM, McLean, Virginia, USA.

Nobata, Chikashi, Nigel Collier and Jun-ichi Tsujii 1999. Automatic term identification and classification in biology texts. *Proceedings of the Fifth Natural Language Pacific Rim Symposium (NLPRS 2000)*, 369–374. Beijing, China.

Olsson, Fredrik 2002. Requirements and design considerations for an open and general architecture for information refinement. Licentiate of Philosophy Thesis, Uppsala University, Uppsala. Available as RUUL No. 35 (Reports from Uppsala University, Department of Linguistics). ISBN: 91-973737-1-0, ISSN: 0280-1337.

Olsson, Fredrik and Katrin Tomanek 2008. An intrinsic stopping criterion for committee-based active learning. Submitted.

Osborne, Miles and Jason Baldridge 2004. Ensemble-based active learning for parse selection. *Proceedings of Human Language Technology Conference – the North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL 2004)*, 89–96. ACL, Boston, Massachusetts, USA.

Pereira, Fernando C. N., Naftali Tishby and Lillian Lee 1993. Distributional clustering of English words. *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, 183–190. ACL, Columbus, Ohio, USA.

Pierce, David and Claire Cardie 2001. Limitations of co-training for natural language learning from large datasets. *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, 1–9. Pittsburgh, Pennsylvania, USA.

Powell, Michael J. D. 1987. Radial basis functions for multivariable interpolation: A review. J. Mason and M. Cox (eds), *Algorithms for Approximation*, 143–167. New York, New York, USA: Oxford: Clarendon Press.

Quinlan, Ross J. 1993. *C4.5: Programs for machine learning*. San Mateo, California: Morgan Kaufmann.

Ramshaw, Lance A. and Mitchell P. Marcus 1995. Text chunking using transformation based learning. *Proceedings of the Third Workshop on Very Large Corpora*, 82–94. ACL, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Reichart, Roi and Ari Rappoport 2007. An ensemble method for selection of high quality parses. *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, 408–415. ACL, Prague, Czech Republic.

Ringger, Eric, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi and Deryle Lonsdale 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. *Proceedings of the Linguistic Annotation Workshop*, 101–108. ACL, Prague, Czech Republic.

Sassano, Manabu 2002. An empirical study of active learning with support vector machines for Japanese word segmentation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 505–512. ACL, Philadelphia, USA.

Schapire, Robert E. 1990. The strength of weak learnability. *Machine Learning* 5 (2): 197–227 (June).

Schapire, Robert E. 2003. The boosting approach to machine learning: An overview. D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick and B. Yu (eds), *Nonlinear Estimation and Classification*, Volume 171 of *Lecture Notes in Statistics*, 149–172. Springer.

Schapire, Robert E., Yoav Freund, Peter Bartlett and Wee Sun Lee 1998. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics* 26 (5): 1651–1686 (October).

Scheffer, Tobias, Christian Decomain and Stefan Wrobel 2001. Active hidden Markov models for information extraction. *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis (IDA-2001)*, 309–318. Lisbon, Portugal: Springer.

Schohn, Greg and David Cohn 2000. Less is more: Active learning with support vector machines. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, 839–846. Stanford University, Stanford, California, USA: Morgan Kaufmann.

Sekine, Satoshi 1998. NYU: Description of the Japanese NE system used for MET-2. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia, USA.

Sekine, Satoshi and Hitoshi Ishara 2000. IREX: IR & IE evaluation project in japanese. *Proceedings of the 2nd International Conference on Language Resources and Evaluation*. ELRA, Athens, Greece.

Sekine, Satoshi and Chikashi Nobata 2004. Definition, dictionary and tagger for extended named entities. *Proceedings of The Fourth International*

Conference on Language Resources and Evaluation (LREC). Lisbon, Portugal.

Seung, H. Sebastian, Manfred Opper and Haim Sompolinsky 1992. Query by committee. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 287–294. Pittsburgh, Pennsylvania, USA: ACM.

Shannon, Claude E. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27 (July and October): 379–423 and 623–656.

Shen, Dan, Jie Zhang, Jian Su, Guodong Zhou and Chew-Lim Tan 2004. Multi-criteria-based active learning for named entity recognition. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, 589–596. ACL, Barcelona, Spain.

Steedman, Mark, Rebecca Hwa, Stephen Clark, Miles Osborne, Anoop Sarkar, Julia Hockenmaier, Paul Ruhlen, Steven Baker and Jeremiah Crim 2003. Example selection for bootstrapping statistical parsers. *Proceedings of Human Language Technology Conference – North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL 2003)*, 157–164. ACL, Edmonton, Alberta, Canada.

Tang, Min, Xiaoqiang Luo and Salim Roukos 2002. Active learning for statistical natural language parsing. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, 120–127. ACL, Philadelphia, Pennsylvania, USA.

Tapanainen, Pasi and Timo Järvinen 1997. A non-projective dependency parser. *Proceedings of the Fifth Conference of Applied Natural Language Processing*, 64–71. ACL, Washington DC, USA: Morgan Kaufmann.

Tateisi, Yuka and Jun-ichi Tsujii 2004. Part-of-Speech Annotation of Biology Research Abstracts. *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, 1267–1270. ELRA, Lisbon, Portugal.

Thompson, Cynthia A., Mary Elaine Califf and Raymond J. Mooney 1999. Active learning for natural language parsing and information extraction. *Proceedings of the Sixteenth International Machine Learning Conference (ICML-99)*, 406–414. Bled, Slovenia.

Tjong Kim Sang, Erik F. 2002a. Introduction to the CoNLL-2002 shared task: Language independent named entity recognition. *Proceedings of the Conference on Computational Natural Language Learning*, 155–158. ACL, Taipei, Taiwan.

Tjong Kim Sang, Erik F. 2002b. Memory-based named entity recognition. *Pro-*

ceedings of the Sixth Conference on Computational Language Learning (CoNLL-2002), 203–206. ACL, Taipei, Taiwan.

Tjong Kim Sang, Erik F. and Fien De Meulder 2003. Introduction to the CoNLL-2003 shared task: Language independent named entity recognition. *Proceedings of the Conference on Computational Natural Language Learning*, 142–147. ACL, Edmonton, Alberta, Canada.

Tomanek, Katrin and Udo Hahn 2008. Approximating learning curves for active-learning-driven annotation. *Proceedings of Sixth International Conference on Language Resources and Evaluation (LREC 2008)*. ELRA, Marrakech, Morocco.

Tomanek, Katrin, Joachim Wermter and Udo Hahn 2007a. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 486–495. ACL, Prague, Czech Republic.

Tomanek, Katrin, Joachim Wermter and Udo Hahn 2007b. Efficient annotation with the jena annotation environment (JANE). *Proceedings of the Linguistic Annotation Workshop*, 9–16. ACL, Prague, Czech Republic.

Tong, Simon and Daphne Koller 2002. Support vector machine active learning with applications to text classification. *Journal of Machine Learning* 2 (March): 45–66.

Tur, Gokhan, Dilek Hakkani-Tür and Robert E. Schapire 2005. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication* 45 (2): 171–186 (February).

Vlachos, Andreas 2006. Active annotation. *Proceedings of the Workshop on Adaptive Text Extraction and Mining (ATEM 2006)*, 64–71. ACL, Trento, Italy.

Vlachos, Andreas 2008. A stopping criterion for active learning. *Computer, Speech and Language* 22 (3): 295–312 (July).

Voorhees, Ellen M. 2001. The Message Understanding Conference Scoring Software User's Manual. URL <http://www-nlpir.nist.gov/related_projects/muc/muc_sw/muc_sw_manual.html>.

Webb, Geoffrey I. 2000. Multiboosting: A technique for combining boosting and wagging. *Machine Learning* 40 (2): 159–196 (August).

Witten, Ian H. and Eibe Frank 2005. *Data mining: Practical machine learning tools with java implementations. 2nd edition.* San Fransisco: Morgan Kaufmann.

Wu, Wei-Lin, Ru-Zhan Lu, Jian-Yong Duan, Hui Liu, Feng Gao and Yu-Quan Chen 2006. A weakly supervised learning approach for spoken language

understanding. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, 199–207. ACL, Sydney, Australia.

Yangarber, Roman and Ralph Grishman 1997. Customization of information extraction systems. *Proceedings of the International Workshop on Lexically-Driven Information Extraction*, 1–11. Frascati, Italy.

Zhang, Kuo, Jie Tang, JuanZi Li and KeHong Wang 2005. Feature-correlation based multi-view detection. *Computational Science and Its Applications (ICCSA 2005)*, Lecture Notes in Computer Science, 1222–1230. Springer-Verlag.

Zhu, Jingbo and Eduard Hovy 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 783–790. ACL, Prague, Czech Republic.

Zhu, Jingbo, Huizhen Wang and Eduard Hovy 2008a. Learning a stopping criterion for active learning for word sense disambiguation and text classification. *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP 2008)*, 366–372. Hyderabad, India.

Zhu, Jingbo, Huizhen Wang and Eduard Hovy 2008b. Multi-criteria-based strategy to stop active learning for data annotation. *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, 1129–1136. ACL, Manchester, England.

A BASE LEARNER PARAMETER SETTINGS

The subject matter of this appendix is the base learner parameter scope and settings as explored and used in conjunction with the single token classification task described in chapter 8. The parameters are to be understood as the options set in Weka when defining the experiments in the Weka Experiment environment (Witten and Frank 2005). The commands in the listings have been cleaned up in order to increase readability, thus they do not constitute valid Weka command lines, in particular, the seeds that are set automatically by Weka have been removed.

A.1 Parameter scope

This section contains information about the scope of the parameters used for the base learners employed in the single token classification task described in chapter 8. Each subsection lists all parameters and their values for a given base learner. In addition to the configurations listed in this section, the two feature selection methods – Consistency-based feature selection and correlation-based feature selection – are invoked for each of the configurations, but the actual command lines are omitted, since listing those would merely mean repeating all configurations two times without actually adding or altering parameters.

In general, a number of parameter settings were tried out for each base learner on a small subset of the data in order to form an image of the influence of the parameters on classifier performance. The settings deemed interesting due to the resulting fluctuations in performance were then further investigated in full experiments involving all data.

In total, 216 combinations of base learners, parameter settings, and data sets were tested.

A.1.1 trees.REPTree

- trees.REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1
- trees.REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1 -P
- trees.REPTree -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- trees.REPTree -M 2 -V 0.0010 -N 6 -S 1 -L -1 -P
- trees.REPTree -M 2 -V 0.0010 -N 2 -S 1 -L -1
- trees.REPTree -M 2 -V 0.0010 -N 6 -S 1 -L -1

A.1.2 trees.J48

- trees.J48 -C 0.25 -M 2
- trees.J48 -C 0.05 -M 2
- trees.J48 -C 0.5 -M 2
- trees.J48 -S -C 0.25 -M 2

A.1.3 functions.RBFNetwork

- functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1

A.1.4 functions.Logistic

- functions.Logistic -R 1.0E-8 -M 50
- functions.Logistic -R 1.0E-8 -M 100
- functions.Logistic -R 1.0E-8 -M 200

A.1.5 bayes.NaiveBayes

- bayes.NaiveBayes -K
- bayes.NaiveBayes -D

A.1.6 bayes.NaiveBayesUpdateable

- classifiers.bayes.NaiveBayesUpdateable

A.1.7 rules.PART

- rules.PART -M 2 -C 0.25 -Q 1
- rules.PART -M 6 -C 0.25 -Q 1

A.1.8 rules.JRip

- rules.JRip -F 3 -N 2.0 -O 2 -S 1
- rules.JRip -F 3 -N 2.0 -O 2 -S 1
- rules.JRip -F 6 -N 2.0 -O 4 -S 1

A.1.9 lazy.IBk

- lazy.IBk -K 5 -W 0 -I -A weka.core.KDTree -A weka.core.EuclideanDistance -W 0.01 -L 40
- lazy.IBk -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance

A.2 Time to train

The listing in this section corresponds to the base learner listing in table 8.6 in chapter 8 such that each item in the list below provides elaborate information about the corresponding base learner entry in the table.

- **lazy.IBk** -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance
- **bayes.NaiveBayesUpdateable**
- **bayes.NaiveBayes** -K
- **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1
- **functions.Logistic** -R 1.0E-8 -M 50
- **trees.J48** -C 0.25 -M 2
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **lazy.IBk** -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.J48** -C 0.5 -M 2
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.Logistic** -R 1.0E-8 -M 50
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayes** -D

- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.PART** -M 2 -C 0.25 -Q 1
- **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **rules.PART** -M 6 -C 0.25 -Q 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.JRip** -F 3 -N 2.0 -O 2 -S 1
- **rules.JRip** -F 3 -N 2.0 -O 2 -S 1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **lazy.IBk** -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **trees.J48** -C 0.25 -M 2
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **functions.Logistic** -R 1.0E-8 -M 100
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **rules.PART** -M 2 -C 0.25 -Q 1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **rules.JRip** -F 3 -N 2.0 -O 2 -S 1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1

A.3 Time to test

The listing in this section corresponds to the base learner listing in table 8.7 in chapter 8 such that each item in the list below provides elaborate information about the corresponding base learner entry in the table.

- **trees.REPTree** -M 2 -V 0.0010 -N 3 -S 1 -L -1 -P
- **trees.J48** -C 0.05 -M 2
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **trees.J48** -C 0.25 -M 2

- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.J48** -C 0.5 -M 2
- **rules.JRip** -F 6 -N 2.0 -O 2 -S 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.REPTree** -M 2 -V 0.0010 -N 3 -S 1 -L -1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.JRip** -F 6 -N 2.0 -O 2 -S 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **rules.JRip** -F 6 -N 2.0 -O 2 -S 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.PART** -M 2 -C 0.25 -Q 1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **rules.PART** -M 6 -C 0.25 -Q 1
- **rules.PART** -M 6 -C 0.25 -Q 1
- **functions.Logistic** -R 1.0E-8 -M 100
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.Logistic** -R 1.0E-8 -M 200
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **functions.Logistic** -R 1.0E-8 -M 100
- **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **bayes.NaiveBayesUpdateable**
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **lazy.IBk** -K 5 -W 0 -I -A weka.core.KDTree -A weka.core.EuclideanDistance -W 0.01 -L 40

- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **lazy.IBk** -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance
- **lazy.IBk** -K 5 -W 0 -I -A weka.core.KDTree -A weka.core.EuclideanDistance -W 0.01 -L 40

A.4 Accuracy

The listing in this section corresponds to the base learner listing in table 8.8 in chapter 8 such that each item in the list below provides elaborate information about the corresponding base learner entry in the table.

- **rules.JRip** -F 3 -N 2.0 -O 2 -S 1
- **rules.PART** -M 2 -C 0.25 -Q 1
- **trees.REPTree** -M 2 -V 0.0010 -N 3 -S 1 -L -1 -P
- **trees.J48** -C 0.5 -M 2
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.J48** -C 0.5 -M 2
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.PART** -M 2 -C 0.25 -Q 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **lazy.IBk** -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance
- **lazy.IBk** -K 5 -W 0 -I -A weka.core.KDTree -A weka.core.EuclideanDistance -W 0.01 -L 40
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **trees.J48** -C 0.5 -M 2
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **rules.PART** -M 2 -C 0.25 -Q 1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **lazy.IBk** -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.JRip** -F 6 -N 2.0 -O 4 -S 1
- **functions.Logistic** -R 1.0E-8 -M 200
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayes** -D

- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **rules.JRip** -F 6 -N 2.0 -O 4 -S 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayes** -D
- **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.Logistic** -R 1.0E-8 -M 50
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **functions.Logistic** -R 1.0E-8 -M 100
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **bayes.NaiveBayesUpdateable**

A.5 Combined results

The listing in this section corresponds to the base learner listing in table 8.9 in chapter 8 such that each item in the list below provides elaborate information about the corresponding base learner entry in the table.

- **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **trees.J48** -C 0.05 -M 2
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.J48** -C 0.5 -M 2
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **rules.JRip** -F 3 -N 2.0 -O 2 -S 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.PART** -M 2 -C 0.25 -Q 1
- **rules.PART** -M 6 -C 0.25 -Q 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.J48** -C 0.25 -M 2
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.JRip** -F 3 -N 2.0 -O 2 -S 1

- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **functions.Logistic** -R 1.0E-8 -M 100
- **lazy.IBk** -K 5 -W 0 -I -A weka.core.KDTree -A weka.core.EuclideanDistance -W 0.01 -L 40
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **lazy.IBk** -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance
- **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.Logistic** -R 1.0E-8 -M 50
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.PART** -M 2 -C 0.25 -Q 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **rules.JRip** -F 6 -N 2.0 -O 2 -S 1
- **bayes.NaiveBayesUpdateable**
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **lazy.IBk** -K 2 -W 0 -A weka.core.LinearNN -A weka.core.EuclideanDistance
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **functions.Logistic** -R 1.0E-8 -M 100
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S BestFirst -D 1 -N 5 -W **functions.RBFNetwork** -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1

Awaiting Enlightenment