



Deliverable 3.4

Survey of state of the art methods for content acquisition and representation

Authors:

Björn Gambäck	Fredrik Olsson	Magnus Sahlgren	Enrico Zovato
SICS	SICS	SICS	Loquendo
gamback@sics.se	fredriko@sics.se	mange@sics.se	enrico.zovato@loquendo.com

Workpackage: WP3.4 Domain and Community Adaptability

Type: Report

Distribution: Public

Date: 30 September 2007

Status: Final

Deliverable Coordinator: Björn Gambäck, SICS

Reviewers:

Area Coordinator: Björn Gambäck, SICS & Jan Hajić, Charles University

Project Coordinator: Yorick Wilks, University of Sheffield

EU Project Officer: Michel Brochard

Table of Contents

Table of Contents	iii
Abstract	v
1 Introduction	1
1.1 Overview	2
1.2 Domain adaptation	3
1.3 Dimensionality reduction	5
1.3.1 Feature selection	5
1.3.2 Feature extraction	5
1.4 Content representation	6
2 Companions scenarios	7
2.1 Lexicon acquisition	7
2.1.1 Types of lexical acquisition	8
2.1.2 Lexicon acquisition based on distributional statistics	9
2.2 Information extraction	11
2.3 Speech synthesis	13
3 An experiment: Named Entity Recognition	15
3.1 Technical considerations	15
3.2 Data	16
3.3 Named Entity Recognition as token sequence labelling	16
3.3.1 Parameter optimization	17
3.3.2 Data set context size versions	17
3.3.3 Instance representation	18
3.3.4 Feature selection methods	19
3.3.5 Selected machine learning schemes	19
3.4 Results	21
3.4.1 Time to train	21

3.4.2	Time to test	23
3.4.3	Coverage	24
3.4.4	Combined results	25
4	Recommendations for Companions	27
4.1	Lexical Acquisition	27
4.2	Named Entity Recognition	27
	References	29
	Appendices	35
A	Parameter settings	37
A.1	Time to train	37
A.2	Time to test	39
A.3	Coverage	40
A.4	Combined results	42

Abstract

The survey reported in this deliverable has as its main goals to give an experimentally-driven overview of different machine learning techniques for content acquisition and representation and point to ways in which the techniques may be further integrated into the COMPANIONS prototypes. In particular we discuss the need for adaptive content representations in the COMPANIONS project and investigate the characteristics of methods for extracting and organising content with respect to learning curve, turn-around time, size of data, etc., focusing on learning strategies employing Active Learning in which the learning algorithm asks an oracle (e.g., a human) to classify instances that confuse the algorithm.

The topics of domain adaptation, dimensionality reduction, (semantic) content representation, and automatic content acquisition are introduced first, while paying particular attention to the idea of using “Bag-of-Concepts” (rather than the more traditional Bag-of-Words approach) for content representation and to *Random Indexing*. Then we discuss some application areas and COMPANIONS related scenarios where automatic methods will be of particular interest. These scenarios include automatic lexicon acquisition based on distributional statistics, information extraction, and text-to-speech synthesis.

A key part of the deliverable consists of hands-on experiments with machine learning algorithms applied to one of the COMPANIONS scenarios, namely *Named Entity Recognition* in order to provide recommendations regarding choice of data representation and ML algorithms for a COMPANION which employs learning components that continuously face new data and thus have to update or re-train their models. The conclusions we draw are that decision trees should be used while incremental learning methods should be avoided, that a small token context is enough, and that one should be careful when using automatic feature selection. We also argue that distributional statistics should be used to acquire lexical information, that a word-space methodology should be employed, and that Random Indexing should be used for word space dimensionality reduction.

Chapter 1

Introduction

“It is reasonable to think that a good representation is the key to effective domain adaptation” (Ben-David et al., 2007)

The work described here concerns **Objective 3.4** of the COMPANIONS project, that is, *Adaptation to other domains and communities*. In particular, this report relates to **Task 3.4.1: Survey of state of the art methods for domain adaptability and the acquisition and representation of semantic content in different modalities which forms the basis for the domain adaptability work in the project and in particular for the other tasks of WP3.4 that will aim to investigate and evaluate different strategies for identifying the semantic content conveyed by multiple input modalities; machine learning techniques for content acquisition including user-centric learning approaches; and representations of content in different modalities and languages.**

The particular survey reported in this deliverable has as its main goals to give an overview of different machine learning techniques for content acquisition and representation, and not “only” an overview in the form of a summary of the state-of-the-art as reported in other publications, but also an overview in the form of hands-on experiments using novel ML strategies in the application areas of the COMPANIONS project, and point to the ways the machine learning techniques may be further integrated into the COMPANIONS prototypes.

In short, the questions we want to investigate overall in **Workpackage 3.4: Domain and Community Adaptability** in general and in this report in particular concern issues such as:

- What is content?
- How can we acquire/extract content from text (and other modalities)?
- How can we represent (different kinds of) content?
- The need for adaptive content representations in the COMPANIONS project.
- Characteristics of methods for extracting and organising content.
- Active machine learning methods (relying on the user as an oracle).

1.1 Overview

The report is laid out as follows: This chapter introduces the topics of domain adaptation, dimensionality reduction, (semantic) content representation and automatic content acquisition. Concretely, we will see that a problem with using statistical algorithms for domain adaptation is that the algorithms often assume that source and target data are drawn from the same probability distribution and that a model built from source data can be directly applied to the target domain. However, this is typically not the case, and as a remedy we suggest using Active Learning (Cohn et al., 1994) in which the learning algorithm has some control over instances on which it trains: the algorithm explores the training data and asks an oracle (e.g., a human) to classify instances that confuse the learning algorithm. On the topic of dimensionality reduction we will discuss the problems of feature selection and feature extraction, while the theme of content representation will introduce the idea of using “Bag-of-Concepts” (rather than the more traditional Bag-of-Words approach). Two topics will be of particular interest in that section, the *Johnson-Lindenstrauss lemma* (which states that a vector space may be projected into a random subspace of appropriate dimensionality without corrupting the relative distances between points in the space) and *Random Indexing*.

Chapter 2 then discusses some application areas and COMPANIONS related scenarios where these automatic methods will be of particular interest. Such scenarios include Information Extraction and Text-to-Speech synthesis. An application area which will be discussed at length is *Lexicon Acquisition*, and then in particular lexicon acquisition based on distributional statistics (over large text data). That is, fully automatic methods that require a bare minimum of pre-processing and language specific tuning. These allow for very efficient implementation and are often implemented using vector algebra, normally assuming some variant of the *distributional hypothesis*, i.e., that words with similar distributional behavior have similar meanings.

In Chapter 3 we describe some experiments of the above-mentioned type, applied to one of the COMPANIONS scenarios, namely *Named Entity Recognition*. Here the purpose is to provide hands-on recommendations regarding choice of data representation and ML algorithms for a COMPANION which employs learning components that continuously face new data and thus have to update or re-train their models. We will discuss choice of features and learners/algorithms, and investigate the behaviour of the algorithms in terms of accuracy, time to learn, and time to classify, as well as combined overall results. It is clear that on the task at hand (as fast and correct as possible decide whether a token is a part of a name), the two tested decision tree learners (REPTree and J48) clearly out-perform the other learning algorithms.

Finally, Chapter 4 sums up the previous discussion by giving some recommendations for how the content acquisition and representation methods could be used within the COMPANIONS project (in particular during its second phase). With regards to Lexicon Acquisition we will argue that distributional statistics should be used to acquire lexical information, that word-space methodology should be used as implementation, and that Random Indexing should be used for word space dimensionality reduction. Regarding Named Entity Recognition our conclusions are that decision trees should be used while incremental learning methods should be avoided, that a small token context is enough, and that one should be careful when using automatic feature selection.

1.2 Domain adaptation

Domain adaptation is the task of porting an algorithm from one domain to another — e.g., porting a named entity recognizer from newswire texts to documents about health and fitness.

In such a scenario, we typically have access to fairly large amounts of annotated data in a *source* domain, which is the known domain in which we have access to both training data and key (in the example above, this would be the newswire texts labeled with named entities). Additionally, we may have access to a small annotated corpus in the *target* domain, which is the domain we want to apply our model to, and which might be entirely or partly unknown (i.e. the health and fitness documents). If we have access to annotated data in the target domain, we have a *supervised* setting, while if we only have access to *unannotated* data in the target domain, we have a *semi-supervised* setting.

The problem with domain adaptation is that many statistical learning algorithms assume that the source and target data are drawn from the same probability distribution, and that a model built from the source data can be directly applied to the target domain. This is typically not the case.

On the contrary, it is most often the case in language technology applications that we have access to annotated data in one specific domain (such as newswire texts), while our real-world applications operate on a radically different domain (such as image captions or health and fitness documents).

Producing annotated data for the new domain can be very costly and time-consuming, and in many cases it is not feasible to demand that users provide annotations in unknown cases. One remedy to the problem of producing annotations for new data is to use an *active learning* approach. In active learning, the learner has some control over instances on which it trains (Cohn et al., 1994). The learner explores the training data and asks an oracle, e.g., a human, to classify any instance that confuses the learner as to which target class it belongs.

A prototypical example of a situation where active learning is suitable is that of data annotation; an annotation system employing an active learning strategy would only bother the user to review the instances for which the system is uncertain of what label the instances should have, whereas an “ordinary” annotation system would require the user to review all instances. One could say that the purpose of active learning is to re-order available training instances in order to achieve optimal generalization while considering as few training instances as possible.

It is anticipated that someone using a COMPANION will, knowingly or unknowingly, provide the system and any learning components therein with training data, e.g., by highlighting relevant text or marking images in a photo album. This process will most likely result in a relatively small number of training instances, exhibiting a skewed distribution among target classes. The user experience of the COMPANION as a whole will benefit from any on-line learning component being as unintrusive as possible, only asking the user for clarification in as few cases as possible, regarding such training examples that really matter. In this situation, there is clearly a need for rapidly reaching generalisations powerful enough to constrain the amount of training data a COMPANIONS user has to review.

Cohn et al. (1994) show that, for some situations, active learning is proven to give better generalizations, for a fixed number of training instances, than learning from arbitrarily or randomly ordered examples alone.

Active learning has been successfully employed in a range of language technology tasks, e.g., parsing (Thompson et al., 1999), corpus linguistics (Banko and Brill, 1986), named entity recognition (Shen et al., 2004), data annotation for the Semantic Web (Ciravegna et al., 2002), and information extraction (Finn and Kushmerick, 2003).

Active learning seems to be a viable path to decrease the amount of labeled instances needed to achieve a given target accuracy, as well as a means to obtain high quality training data while at the same time keeping down the cost of acquiring it.

Once we have access to data sets for the source and target domains, there are a number of different approaches to tackle the problem of domain adaptation. Daumé (2007) gives a nice summary:

- We can choose to ignore the target data all together, and just train a model on the source domain and hope that it will fit the target domain.
- We can also choose to disregard the source data, and merely train a model on what available target data we happen to have.
- We can use *both* the source and target data for training, treating them as belonging to the same domain.
- If the source domain is much larger than the target domain (as is typically the case), we can weight the examples in the two domains so that both domains get equal weight, despite differences in corpus size.
- We can train a model on the source domain, apply it to the target data, and use the output as additional features in a new model trained on the target data.
- We can use a model trained on the source data to initialize the weights (as a *prior*) for a second model that is trained on the target data (Chelba and Acero, 2006).
- We can interpolate the models trained on the source and target data separately.
- Lastly, as proposed by Daumé (2007), we can augment the feature space in such a way that we represent each feature as three separate features: one source-specific feature, one general feature, and one target-specific feature.

Daumé (2007) reports that the last and simple method outperforms all other domain adaptation methods on a number of different test setups.

1.3 Dimensionality reduction

The idea of producing a *general*, or *objective*, feature space is also the motivation for a number of approaches that utilize dimensionality-reduction techniques to restructure the feature space. The underlying idea is to use dimensionality reduction to learn a general feature representation that will be common for several domains. It is hypothesized that by applying dimensionality reduction, a model produced from the source data will generalize better to the target domain (Blitzer et al., 2006a).

Dimensionality reduction can be accomplished in many ways. With the risk of over-generalizing, we can say that there are two main approaches: *feature selection* and *feature extraction*.

1.3.1 Feature selection

Feature selection means that certain features are selected based on statistical measures, such as information gain, χ^2 , or mutual information (Yang and Pedersen, 1997).

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), while the other also involves the learning method under investigation (*wrapper* methods). Witten and Frank (2005) give a nice overview of the arguments for, and methods behind, feature selection.

1.3.2 Feature extraction

Feature extraction means that “artificial” features are created from the original ones, typically by using factorization techniques such as Independent Component Analysis (ICA) (Hyvärinen et al., 2001), Non-negative Matrix Factorization (NMF) (Lee and Seung, 2000), Singular Value Decomposition (SVD) or Principal Component Analysis (PCA) (Golub and Van Loan, 1996). Factorization is a mathematical process where an object is decomposed into the product of other objects, called factors. Thus, a matrix factorization is the right side of the equation $F = M_1 M_2 \dots M_m$.

The arguably most well-established and widely used matrix factorization technique in language technology research is SVD, in which the original matrix is decomposed into *three* smaller matrices, which can be multiplied to reproduce the original one. These smaller matrices contain the linearly independent factors of the original matrix (in the case of SVD, they are called “singular vectors” and “singular values”). If the smallest factors are disregarded when multiplying the smaller matrices, the result will be an *approximation* of the original matrix. This process is called *truncated* SVD, and is, as previously mentioned, the favored dimensionality-reduction method in many text-based information-access applications. In the context of domain adaptation, Blitzer et al. (2006b) is a nice example where SVD is used to produce *pivot features*, which are assumed to be common to several distinct domains.

1.4 Content representation

An interesting side-effect of using feature-extracting dimensionality-reduction methods is that when such approaches are applied to word representations, they are able to handle problems with vocabulary variation in the form of synonymy (that several words have similar meanings). This is accomplished when words that have similar features are grouped together in the factorization. Since these methods do not represent texts merely as collections of the words they contain, but rather as collections of the *concepts* they contain, Sahlgren and Cöster (2004) suggested that a more fitting label for these representations would be *Bag-of-Concepts* (BoC).

The ability to handle synonymy was the motivation for using SVD in the very successful Latent Semantic Indexing (LSI) (Deerwester et al., 1988; Landauer and Dumais, 1997) model of information retrieval, in which a terms-by-documents matrix is decomposed by SVD to a few hundred dimensions, thus facilitating the retrieval of documents that not necessarily contain any of the terms in the query. Since the initial success in information retrieval, LSI (or, equivalently, LSA) has been used in a number of natural language processing applications, including knowledge assessment (Wolfe et al., 1998), collaborative filtering (Hofmann and Puzicha, 1999), textual coherence detection (Foltz et al., 1998), and information filtering (Dumais, 2004).

A serious concern with the use of matrix factorization techniques like ICA, NMF, PCA and SVD is that they are computationally heavy, and thus are liable to efficiency and scalability issues. In many real-world applications that require online processing, it is not feasible to use models that rely on SVD or similar techniques to restructure the high-dimensional feature space. As an alternative to such computationally heavy dimensionality-reduction techniques, alternative approaches that rely on the Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984) have been suggested. The lemma states that a vector space can be projected into a random subspace of appropriate dimensionality without corrupting the relative distances between points in the space. Thus, the dimensionality d of a high-dimensional matrix M can be reduced to k (where $k \ll d$) by multiplying it with a random matrix R :

$$M_{w \times d} R_{d \times k} = M'_{w \times k}$$

If the random vectors d_i in matrix R are orthogonal, so that $R^T R = I$, then $F = F'$; if the random vectors are *nearly* orthogonal, then $F \approx F'$ in terms of the similarity of their rows. A very common choice for matrix R is to use Gaussian distribution for the elements of the random vectors. However, Achlioptas (2001) has shown that much simpler distributions (practically all zero mean distributions with unit variance) give a mapping that satisfies the lemma.

Dimensionality-reduction methods that rely on the Johnson-Lindenstrauss lemma such as Random Mapping (Kaski, 1999) and Random Indexing (Kanerva et al., 2000; Sahlgren, 2005) are considerably more efficient both in terms of CPU cycles and memory usage than methods based on matrix factorization. Random Indexing also has the benefit that it is an *incremental* method and does not employ a separate random projection phase; instead of first assembling a high-dimensional vector space and then reducing its dimensionality, Random Indexing *accumulates* a reduced-dimensional vector space by adding together sparse, high-dimensional random vectors.¹ Random Indexing is thus ideally suited for applications that require online processing.

¹See Sahlgren (2005) for an introduction to Random Indexing.

Chapter 2

Companions scenarios

In this chapter we will detail some of the scenarios where the use of content acquisition methods are of particular interest for the COMPANIONS project and for domain adaptability in general. First, Section 2.1 looks at lexicon acquisition, a prototypical task for any language processing system aimed at real life applications. Then, Section 2.2 gives a shorter overview of the application of learning methods to Information Extraction, a scenario which will also be the topic of Chapter 3. Finally, Section 2.3 singles out the task of grapheme to phoneme conversion in text to speech synthesis as a relevant scenario outside the pure text processing field.

2.1 Lexicon acquisition

Lexical resources are necessary for any type of natural language processing and language engineering applications. Whereas in the early days of language engineering most lexical information was hard-coded into the system, today most systems and applications rely on explicitly introduced and modularly designed lexica to function: examples range from applications such as automatic speech recognizers, dialogue systems, and writing aids to computational linguistic techniques such as word sense disambiguation systems.

In general, modern language processing theories tend to rely heavily on *lexicalization*, that is, to place more and more information in the lexicon, information that has traditionally resided in the grammar rules. The demand for automatic lexical acquisition has thus increased, even though the problems associated with acquiring such a large lexicon often are neglected or at least underestimated. Still, the need for large lexica is not new and the task of automated lexical acquisition has over the years been approached from several different directions, both with tools supporting the user in creating lexical entries and with tools that try to automate the acquisition process completely.

Multilingual applications, which are driven by modeling lexical correspondences between different languages, are obviously reliant on lexical resources to a high degree — the quality of the lexicon is the main bottleneck for quality of performance and coverage of service. While automatic text and speech translation have been the main multilingual tasks for most of the history of computational linguistics, today the recent awareness within the information access

field of the multilingual reality of information sources has made the availability of lexica an all the more critical system component.

However, machine readable lexica in general, and machine readable multilingual lexica in particular, are not easy to find and are laborious to build. Manual approaches to lexicon construction vouch for high quality results, but are time- and labour-consuming to build, costly and complex to maintain, and inherently static as to their nature: tuning an existing lexicon to a new domain is a complex task that risks compromising existing information and corrupting usefulness for previous application areas. Automatic lexicon acquisition techniques, on the other hand, promise to provide fast, cheap and dynamic alternatives to manual approaches, but have yet to prove their viability. In addition to this, they typically require sizeable computational resources.

2.1.1 Types of lexical acquisition

The problem of automatic lexical acquisition can be defined more formally as

Definition 1 (Automatic lexical acquisition)

Lexical acquisition is the task a program performs when it from language data automatically adds a new lexeme or phrase together with its associated lexical information to the lexicon of a language processing system.

Where *language data* may be any (language-based) information source, such as a plain text, a machine readable dictionary, etc.

One of the first systems that aimed at constructing lexical entries automatically from raw text was FOUL-UP (Granger, 1977). FOUL-UP was an integral part of the SAM system (Schank, 1975) developed to extend SAM's lexicon by inferring restrictions placed on unknown words by instantiating scripts that matched the sentences containing the unknown words. This built on a number of assumptions which in general do not hold, in particular: that all the information needed to create an entry is contained in one text; that no morphological information is needed; that specific (hand-coded) scripts covering the domain can be made available, etc. Instead, Jacobs and Zernik (1988) showed the need to consult a variety of knowledge sources such as morphological, syntactic, semantic, and contextual knowledge when determining a new lexical entry.

Semi-automatic lexical acquisition on the other hand refers to approaches to solve the acquisition problem that have created a range of tools that require various degrees of *interactive* support when new lexical entries are created, either from raw text material or from machine readable dictionaries.

If the lexical acquisition system is designed with a specific application in mind, the task is some-what simplified. Such a system was TEAM (Grosz et al., 1987), which was designed specifically as a front-end for databases of a particular kind. This meant that lexical acquisition in TEAM was essentially a matter of determining the English counterparts of particular database relations, and that the possibilities for word behaviours were constrained by the kinds of relations that existed in the databases. Thus TEAM was able to allow the user to volunteer

a sentence from which, with the help of some hard-wired auxiliary questions, it inferred the syntactic and semantic characteristics of the way a verb and its arguments mapped into the database.

The success of a lexical acquisition project is very much dependent on a correct choice of information sources. Several approaches to the problem have chosen to use whatever information can be extracted from already available machine readable dictionaries. This can be attempted both by automatic methods or semi-automatic ones. Even though the information in such a dictionary is not of a form which can be directly used in a language processing system, the task of creating new lexical entries from it might be easier than from raw text. The definitions in the dictionaries often follow a very strict syntax, for example.

Most approaches to automatic lexical acquisition do, however, attempt to create a new lexicon directly from text corpora. The problem is simplified if part-of-speech tagged corpora are available. When processing untagged text on the other hand, we cannot normally use all information in the text (i.e., the whole text, possibly including semantics), but have to restrict ourselves in one way or the other. A person trying to determine the grammatical category of an unknown word would normally look at the meaning of the word, as it appears in a particular context. A state of the art automatic lexical acquisition system does, however, not have access to much semantics. It has to manage with distributional statistics, syntactic information only, or even just with the physical appearance of the word (that is, the actual letters of the unknown word itself will provide some information, for example in the form of affixes).

Syntactic information is available in a tagged text, but part of it may also be found in untagged text, for example, in the form of cooccurrence statistics. The grammatical categories in a sequence of words obviously make a pattern. The “windows” through which we look at the sequence can be of different sizes and are commonly referred to as *N-grams*, where the ‘N’ denotes the number of words in the window. Keeping the size as small as possible is normally desirable as this reduces the complexity of the knowledge representation patterns which have to be created, while the available information more easily can be generalized.

Another way to look at the context is by *clustering* of the words into related groups. The Edinburgh Knowledge Acquisition Workbench (Mikheev and Finch, 1995) used bigram statistics. It also sorted terms with the same head-word by length and then used information on semantic categories available in WordNet (Miller, 1990) to cluster the modifiers into groups. The similarities between different contexts in which words occur were used to induce the similarities between the words themselves.

2.1.2 Lexicon acquisition based on distributional statistics

A particularly promising approach to automatic lexicon acquisition is a methodology based on collecting *distributional statistics* over large text data. This is attractive since it is fully automatic and requires a bare minimum of preprocessing and language specific tuning. In addition to that, techniques based on collecting distributional statistics often allow for very efficient implementation. Such approaches normally assume some variant of the so-called *distributional hypothesis*, which states that words with similar distributional behavior have similar meanings (Rubenstein and Goodenough, 1965; Schütze and Pedersen, 1995). See Sahlgren (2006)

for a thorough discussion about the origins, implications and implementations of this hypothesis.

Lexicon acquisition techniques based on distributional data are often implemented using vector algebra, and are sometimes referred to as *word space models* (Schütze, 1993; Sahlgren, 2006). In such models, the (text) data is collected in a words-by-contexts matrix which is populated with (normalized and/or weighted) frequency counts. The point of this representation is that the series of frequency counts for each word can be seen as a high-dimensional vector (usually referred to as a *context vector*) in context space. By computing vector similarity between such context vectors, it is possible to quantify (distributional \approx semantic) similarity between words. Sahlgren (2006) demonstrates that using words as contexts — i.e., populating the matrix by counting co-occurrences within a window of word tokens, like in the Hyperspace Analogue to Language (HAL) model (Lund et al., 1995) — produces *paradigmatic* word spaces, while using text regions — i.e., sentences or documents, as in LSA (Landauer and Dumais, 1997) — as contexts produces *syntagmatic* word spaces.

The word-space methodology can be used to acquire lexica in the following manner: distributional statistics is collected, context vectors are produced, and a lexicon is compiled by entering words with similar context vectors (as determined by computing, e.g., the cosine of the angles between the vectors¹) under the same entry in the lexicon. Note that a word space in itself can be seen as an *un-compiled* lexicon, in which similarities between words can be computed on the fly.

The word-space methodology can also be used to produce *multilingual* lexica, in which case aligned parallel data is used for collecting the distributional information. The underlying assumption is that “... words that are translations of each other are more likely to appear in corresponding bitext regions than other pairs of words” (Melamed, 2000), and that collecting distributional information from parallelized data would reveal such translation pairs. Sahlgren and Karlgren (2005) describe an approach where a word-space model is used to accumulate a bilingual vector space by using the aligned pair of text segments in the parallel data as contexts. The result is that words from the different languages are effectively located in the same vector space, and correspondences are found between them by measuring the distance between their context vectors.

Sahlgren’s and Karlgren’s approach is conceptually different from other work using parallel data, in the sense that their approach can be called a *context-based* approach, while other approaches (Brown et al., 1988; Brown et al., 1990; Melamed, 2000) have typically been more *co-occurrence-based*. To illustrate the difference between these approaches, consider Table 2.1:

T_s	Context	T_t
a a a b c c	1	x v y z z z z
a d e	2	v w z
a a a c	3	x x v

Table 2.1: An illustration of the *context-based* approach vs. the *co-occurrence-based* approach.

¹The cosine of the angles between two vectors is given by: $d_{\cos}(x, y) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| |\vec{y}|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$

Two good examples of co-occurrence-based approaches are Brown’s original model (Brown et al., 1988; Brown et al., 1990) and Melamed’s somewhat later model (Melamed, 2000). Brown’s co-occurrence measure is proportional to the product of their joint frequencies in each step; Melamed’s dampens the measure by only taking the smaller of the frequencies. In this case, Brown’s model would find a and z having the closest cooccurrence score — with the score for context one majorizing everything else; Melamed’s model would find a and v the closest, with the number of contexts they engage in majorizing everything else; Sahlgren’s and Karlgren’s model, on the other hand, will find that a has the closest cooccurrence score with contexts $T_s(1)$ and $T_s(3)$, and that x also has the same context model, and that they thus are the most closely corresponding words.

Sahlgren and Karlgren (2005) evaluate their method using freely available online bilingual lexica as gold standards, and report results of around 60% overlap. In the monolingual setting, word-space models have been used for a number of vocabulary acquisition tasks, including synonym extraction (Rapp, 2003; Levy et al., 1998), thesaurus comparison (Sahlgren, 2006), association tests (Lund et al., 1995; Sahlgren, 2006) and query expansion for information retrieval (Sahlgren et al., 2003), just to name a few.

2.2 Information extraction

Information extraction (IE) is concerned with recognising and extracting entities and relations between them from text, and to create a structured representation of those. Introductions to information extraction are given by, e.g., Cowie and Lehnert (1996), Grishman (1997), and Appelt and Israel (1999). The core definition of IE has evolved during the Message Understanding Conference (MUC) series (Science Applications International Corporation, 2000).

After the MUC series, there has been other forums for evaluating information extraction systems, e.g., the Translingual Information Detection, Extraction and Summarization (TIDES) (LDC, 2007), and Automatic Content Extraction (ACE) (NIST, 2007) programs sponsored by DARPA, and the Pascal Network of Excellence (PASCAL, 2007) sponsored by the EU.

Figure 2.1 illustrates the organisation of a typical IE system. Usually, IE systems are constituted of 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 2.1, text is fed to the system and passed through a *lexical analysis* phase which involves segmenting the document into, e.g., sentences and tokens. The tokens are then analysed in terms of, e.g., part-of-speech and syntactic functions. Next, the *name recognition* module harvests the text for names of, e.g., persons, organisations, places, monetary expressions, and dates. The *partial syntax* step includes identifying noun 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 prepositional phrase attachment. *Reference resolution* and *discourse analysis* 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.

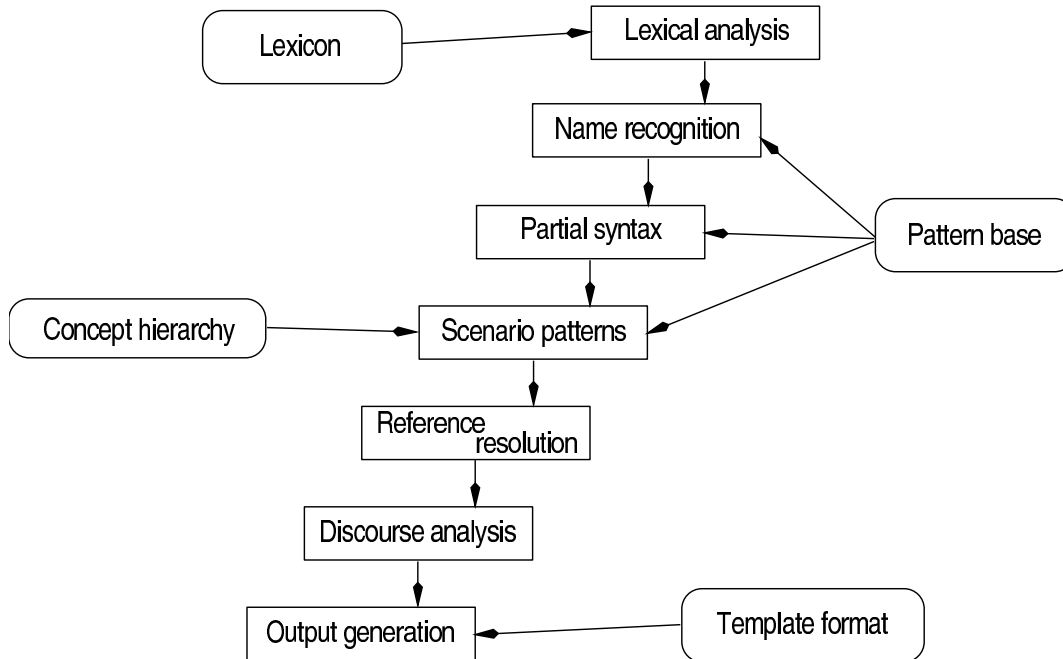


Figure 2.1: The organisation of a typical information extraction system, adopted from Yangarber and Grishman (1997).

Some of the modules are less domain and scenario dependent, such as name recognition and simple phrase grouping, while others are very specific to the information need that a given system is designed to fulfill.

Initially, information extraction systems were conceived as systems of rules, implemented by hand, partly relying on the designer’s intuition. Since then, mainly due to domain adaptation issues, there has been a shift towards using machine learning for discovering relevant patterns in existing data. Turmo et al. (2006) present an extensive survey focusing on adaptive systems for extracting information from text.

We anticipate that there will be COMPANIONS needing to have information extraction capabilities so as to, e.g., be able to extract, store and work with names of persons and places while interacting with a user concerning the user’s photo album. We further assume that the COMPANION will employ learning strategies to adapt to the user’s use of language and to the domain in which the COMPANION -user interaction is to take place. To accommodate for these issues, data-driven methods for creating new named entity recognizers, or adapt existing ones, will be investigated and reported on in Chapter 3.

2.3 Speech synthesis

Even though most of this report is concentrated on text processing and text-related issues, in general, machine learning type strategies will be applicable to many areas necessary for the successful construction of COMPANION type systems, in particular within speech processing and speech production. Here we concentrate on one such task, the task of Grapheme to Phoneme conversion in speech synthesis. This can be accomplished following two approaches. The first one is based on rules and is suitable for languages whose phonetic is quite “regular” in the sense that it is based on simple conversion rules with few exceptions. The second approach is based on machine learning techniques and is suitable for languages, like English, whose phonetic transcription can hardly be managed by simple conversion rules.

The second strategy exploits a language-independent learning scheme, trained on large phonetic lexicons of inflected forms. Inductive learning techniques like this are able to find out the common characteristics in the data and generalize them. For this task, Loquendo TTS exploits Classification And Regression Trees, CART (Breiman et al., 1984) that have proven useful to yield phonetic transcription with lexical stress assignment and in the past have been successfully used to predict prosodic patterns. This solution heavily relies on accurate phonetic lexicons, that contain graphemic strings and their corresponding phonetic representation.

First, a forced alignment between graphemes and phonemes is performed. Null symbols are used in those cases where more than one phoneme is associated to a grapheme and vice-versa. The alignment is based on a statistical model that provides the most likely associations between the two entities, through the application of a dynamic programming algorithm. The distance metric is given by the probability that the grapheme g will be transcribed as the phoneme p , that is $P(p|g)$. The best alignment is given by the maximum global likelihood. The probabilities are re-estimated at each step and the process is iterated until a converging association model is found.

After the lexicon’s alignment, the training database for CART has to be set up, i.e., a set of patterns associated with a number of features (numerical or categorical) and a target value. Starting from these data, the CART training algorithm builds up the root node (one for each grapheme) and then, selecting the features that better discriminate the output phonemes, creates a splitting rule associated with the root. This procedure is extended to the children until splitting rules can be found. The leaves of the tree are the phonemes, in this way every path from the root node to a leaf represents a rule in which the number of tests corresponds to the number of nodes involved in the path. Recent improvements have been introduced in this module by training the system with revised and incremented lexicons. Moreover, the new module is more flexible and scalable in terms of number of rules extracted, and comprises a new feature to manually add rules with higher priority than the learned ones. This allows a fine tuning to better manage exceptions in the grapheme to phoneme conversion.

Chapter 3

An experiment: Named Entity Recognition

Machine learning for named entity recognition (NER) is far from a new activity, and it is exactly the widespread work on this task that makes it attractive in a COMPANIONS context — an experiment in this setting, addressing a well-known task, using a well-known data set, opens up for comparisons along other dimensions than are usually explored. While most previous work has been concerned with reaching the highest numbers in terms of recall, precision and F-score, the angle of approach in this experiment is slightly different — in addition to correctly learn from available data, a COMPANION will most likely also be constrained in terms of the time it has to its disposal for learning and classification. Thus, the focus of this experiment will be on comparing several machine learning schemes, deployed on the same task, wrt learning time, classification time, as well as performance in term of recall, precision, and F-score. For each learning scheme, variations in parameter settings, data representation, and automatic feature selection methods will be investigated.

The main purpose of this experiment is to provide the COMPANIONS project with hands-on recommendations regarding choice of data representation and machine learning algorithms suitable for a setting where a COMPANION employs (on-line) learning components which continuously face new training data, and thus have to update or re-train their models in order to keep pace with the user.

3.1 Technical considerations

The machine learning toolbox of choice for this experiment is Weka (Witten and Frank, 2005) which is a freely available, and large, collection of machine learning algorithms implemented in Java. The algorithms can be utilized via a provided API or via Weka's own graphical user interface. What is particularly useful in Weka is the possibility to define machine learning experiments in the Weka Experimenter environment 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 analyse the results using various statistical methods. Weka is available on the Internet at <http://www.cs.waikato.ac.nz/ml/weka/>

Weka require the data it operates on to be in what is known as the Attribute Relation File Format, or ARFF for short. 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 3.3.3 for an elaboration on the features chosen). The English Functional Dependency Grammar (EN-FDG, version 3.6) from Connexor Oy is used for linguistic analysis of the texts in order to decide the values for some of those features.

3.2 Data

The data at hand is the training portion of the named entity part of the MUC-7 corpus which consists of 100 news items regarding air crashes (Linguistic Data Consortium, 2001). In the named entity recognition task in MUC-7, the aim is to recognise and categorise occurrences of names of *persons*, *organizations*, *locations* (usually referred to as ENAMEX), *monetary expressions*, *percentages* (NUMEX), and *dates*, *times* (TIMEX) that appear in the text. Bagga (1998) presents an analysis of, e.g., the MUC-7 data wrt complexity and information extraction.

The part of the MUC-7 corpus used in this experiment contains 4969 ENAMEX, 1441 TIMEX, and 140 NUMEX instances. When processed with the Functional Dependency Grammar, the total number of tokens in the corpus adds up to 93073. Names vary in length from one token (3296 instances) to 12 tokens (1 instance).

3.3 Named Entity Recognition as token sequence labelling

Instead of trying to recognise and categorise whole names, which often spans more than a single token, the task of named entity recognition is commonly redefined as a token sequence labelling task; for each token, the task is to decide whether the token is a part of a name. Following the IOB tag scheme proposed by, e.g., Ramshaw and Marcus (1995) for sequence labelling, the initial task of assigning one of the $7 + 1$ classes of names to chunks of the input can be turned into a $14 + 1$ class task in which each token is assigned a tag indicating if it is located in (I), outside (O), or in the beginning (B) of any of the classes of names. $B-X$ 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 named entity.

Thus the task addressed in this experiment is that of assigning one of the tags *OUT*, *B-PERSON*, *I-PERSON*, *B-LOCATION*, *I-LOCATION*, *B-ORGANIZATION*, *I-ORGANIZATION*, *B-MONEY*, *I-MONEY*, *B-PERCENT*, *I-PERCENT*, *B-TIME*, *I-TIME*,

B-DATE, and *I-DATE* to each token in the input. Broken down to tokens, the data exhibits the characteristics outlined in Table 3.1.

CLASS	OCCURRENCES
B-PERSON	0
I-PERSON	2321
B-LOCATION	0
I-LOCATION	2340
B-ORGANIZATION	6
I-ORGANIZATION	4289
B-TIME	0
I-TIME	642
B-DATE	1
I-DATE	2501
B-MONETARY	0
I-MONETARY	238
B-PERCENT	0
I-PERCENT	72
OUT	80663

Table 3.1: The part of the MUC-7 corpus used for this task broken down to tokens and their class.

In pursuing the task there are a number of choices to be made that might have a great impact on the final result. For each machine learning scheme, there are the issues of *parameter optimization*, *feature selection* and *data set context size versions*, each of which adds to the problem in a combinatorial way. Since it is not feasible to exhaustively explore the entire space of possible combinations, informed choices have to be made along the way.

3.3.1 Parameter optimization

Each machine learning method has its own parameters that need to be tweaked to obtain the best result wrt to the task and data at hand.

Daelemans and Hoste (2002) show that *“the interaction between algorithm parameter settings and feature selection within a single algorithm often accounts for a higher variation in results than differences between different algorithms or information sources”*.

In the present experiment, the parameter settings for each 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. Once the identification of parameters and approximate parameter values range is decided, a classifier set-up for each such combination is defined in conjunction with the full data sets.

3.3.2 Data set context size versions

In a token sequence labelling task, each token can be represented in terms of its context. As used here, the notion of context is closely related to the issue of instance representation. For

instance, a context window size of 0 means that a token is represented by means of itself, 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 experiment at hand, data sets corresponding to context window sizes of *intrinsic*, 0 to 5 (including) are used. The *intrinsic* context is an instance representation in which some of the features referencing, or depending on, a token's context have been removed (see Section 3.3.3).

3.3.3 Instance representation

The task of named entity recognition has been thoroughly investigated, e.g. in the series of Message Understanding Conferences during the 90's, and some of the important features used to represent data to make the distinction between names and non-names possible are described in a number of papers, e.g., in Bikel et al. (1999), Borthwick et al. (1998), Zhou and Su (1999). In this experiment, we have chosen to use those features that have contributed to a good result as reported by others, as well as a range of other features that are due to the EN-FDG. The full list of features used to represent each token is available in the table below.

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

Table 3.2: The features used to represent a token.

All attributes except for the target class are numeric. The target class is a nominal attribute, which can take any of these values: OUT, B-PERSON, I-PERSON, B-LOCATION, I-LOCATION, B-ORGANIZATION, I-ORGANIZATION, B-MONEY, I-MONEY, B-PERCENT, I-PERCENT, B-TIME, I-TIME, B-DATE, and I-DATE. The above features are calcu-

lated for each token in the context window size, e.g., 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

$$2 * |\text{features in the table} - 1| + |\text{features in the table}| + 2 * |\text{features in the table} - 3|$$

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.

In the *intrinsic* representation of instances (see Section 3.3.2), a token is by the features outlined in Table 3.2, except for *class of previous previous*, *class of previous*, and *is first in sentence*. We are aware of that there are more than these four features that can be argued to be extrinsic.

3.3.4 Feature selection methods

Selecting features refers to (at least) two different stages of setting up a machine learning experiment. The first one being at the time of deciding on the representation of the instances over which the learners will learn, and the second one being when the task is to select among the features in a given set in order to improve classifier performance in some respect, e.g., training time or classification accuracy. This section is about the latter.

The two feature selection methods used in the present experiment both belong to the class of filter methods (as mentioned in Section 1.3).

Automatic selection of the features used for learning include ways of deciding on the amount of information a given feature, or set of features, contribute to the overall process (see, e.g., Witten and Frank (2005) for an overview of feature selection methods). Weka provides a range of such feature selection schemes. Hall and Holmes (2003) investigate the performance of some feature selection techniques on a number of widely used data-sets. Since learning speed, which might also include feature selection, is of particular interest to the experiment presented in this chapter, following the conclusions drawn by Hall and Holmes (2003) relevant to us is that when speed is a concern, *Relief*, *consistency-based subset evaluation* and *correlation-based feature selection* seem to be viable options. The former two are better if it is anticipated that the data contains attributes that interact in a way useful for the learning schemes. If, on the other hand, such attribute interaction is not expected, the latter feature selection method should be used since it is faster. In this experiment, we have settled for *ConsistencySubsetEval* (Liu and Setiono, 1996) and *CfsSubsetEval* (Hall, 1999). Feature selection has been applied on the data containing instances represented in the widest context, that is, a context of five tokens (as explained in Section 3.3.2).

3.3.5 Selected machine learning schemes

The machine learning methods under scrutinization stem from different families of learners, as outlined in Table 3.3. The experiment aims at covering the major families of machine learning techniques, although we arguably fall short of that goal by not yet having included Support Vector Machines.

Intuitively, what is needed throughout this experiment is methods of learning that allows 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 *NaiveBayesUpdateable* (John and Langley, 1995), and a Nearest Neighbor learner, *IBk* (Aha et al., 1991). Since incrementality is anticipated to come at a cost, other methods are given attention too. 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 fast but memory eating 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).

After initial experiments on smaller data sets, the *NaiveBayes* implementation of a Bayesian (John and Langley, 1995) learner gave results that indicated that it would be worthwhile to use it on the full data sets as well.

There evidently is a very large range of Artificial Neural Network configurations to test, the one included here *Radial Basis Function* (Powell, 1987), a two layer feedforward network that differs from a multilayer perceptron in the way that the hidden units perform computations. The RBF network was chosen since it seemed to learn faster than the multilayer perceptron network on a subset of the training data.

The representative of the Meta learning family was selected in a fashion similar to that of the neural network; *MultiBoostAB* (Webb, 2000) outperformed *AdaBoostM1* wrt both accuracy and time in initial experiments conducted on a subset of the data. The meta learner is set up to use with *NaiveBayes* classifiers since the problem at hand is a multi-class one (if it had been a binary classification task, using decision stumps might have sufficed). *NaiveBayes* seemed to be as accurate as *J48*, only faster.

ALGORITHM FAMILY	WEKA IMPLEMENTATION
Incremental learner	NaiveBayesUpdateable
Decision tree	J48 REPTree
Rule learner	JRip PART
Bayesian learner	NaiveBayes
Lazy learner	IBk
Artificial Neural Network	Radial Basis Function
Meta learner	MultiBoostAB

Table 3.3: The machine learning families and Weka implementations used in the experiment.

Each machine learner in the right column in Table 3.3 was run on each one of the seven data sets representing the context sizes introduced in Section 3.3.2, using various parameter settings specific to the individual learning methods. In addition, the two feature selection techniques introduced in Section 3.3.4 were used with each machine learner on the data set representing the largest context size. In total, 207 experiments was run, using 10-fold cross-validation for calculating the resulting scores.

3.4 Results

Note that the results reported are on the token sequence labelling task, not on recognizing entire names. Thus, the recall, precision, and F-score values below might not be comparable to the name tagging results reported by others.

The tables included in this part contain only the best classifiers, that is, the results are only reported for one instance of, e.g., the *IBk* nearest neighbor although several configurations were explored. What matters is the relative order in which the classifiers occur wrt the time required to train on data, the time required to test the classifiers, and the measures of coverage, i.e., the percent correctly classified tokens. Recall, precision, 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 percent of correctly classifier tokens are compared between each classifier.

The listing below serves as a legend to reading the parts of the result tables that has been abbreviated.

- **Ctx** – the context in which each token in the input data is represented. Possible values is *intr* (intrinsic), *c0* - *c5* where the number indicates the size of the context used to represent each token in the data, and *c5-fs* which denotes a context of five tokens on each side which has been reduced with one of the feature selection methods prior to training the classifier.
- **Train/Test** – CPU time required for classifier to train and test on data.
- **PC** – The percent correctly classified tokens.
- **Prec/Rec/F** – Precision, recall, and F-score
- **NBUd** – NaiveBayesUpdateable
- **NB** – NaiveBayes
- **REPT** – REPTree
- **MB** – MultiBoostAB
- **CfsSE** – Correlation-based feature-selection subset evaluation
- **ConsSE** – Consistency-based feature-selection subset evaluation
- **Rank** – The square rank sum assigned to a classifier based on its ranking when training, testing, and its performance in terms of percent correctly classified tokens in the test data.

3.4.1 Time to train

As anticipated, the list of machine learning methods presented in Table 3.4 requiring the least time to train on the training part of the MUC-7 data contain the lazy learner in the top

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
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 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
CfsSE MB	c5-fs	609,11	1,64	95,63	0,99	0,98	0,98
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
MB	intr	2138,36	138,94	91,25	0,98	0,95	0,97
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 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 MB	c5-fs	8080,15	0,89	96,35	0,97	0,99	0,98
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 3.4: Classifiers ordered according to the time required to train on data.

position. 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. Stepping up a couple of orders of magnitudes of training time, we find the *NaiveBayesUpdateable*, *NaiveBayes* and *REPTree* learners. Approximately ten times slower are the *J48* decision tree learner. Then follows a range of learning methods that have been invoked after *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 reduced prior to the learning phase. For instance, the *PART* method learns faster when the 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.

The trend among the two automatic feature selection methods employed in this experiment, is that *correlation-based feature selection* method outperforms *consistency-based feature selection* wrt to training time in all cases. Overall, the classifiers all learn faster on the data represented by relatively few features.

3.4.2 Time to test

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
ConsSE NBUD	c5-fs	6178,32	0,37	86,21	0,91	0,96	0,93
NB	intr	20,99	0,61	92,15	0,99	0,94	0,96
ConsSE MB	c5-fs	8080,15	0,89	96,35	0,97	0,99	0,98
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	0,96
CfsSE MB	c5-fs	609,11	1,64	95,63	0,99	0,98	0,98
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
MB	intr	234,25	6,49	92,77	0,99	0,94	0,97
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 3.5: Classifiers ordered according to the time required to apply to data.

In terms of time required to apply a classifier to the test portion of the data, Table 3.5 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 classifiers that stand out in a bad way; compared to any of the other machine learning methods under scrutinization, the memory-based learner is really a poor performer when it comes to being applied to the test data.

Contrary to the case of training the classifiers (Section 3.4.1), it is not at all clear cut which of the two feature selection methods performs the best; the *consistency-based feature selection* method is better in five of nine cases.

Classifiers trained on a relatively small context (*intr*, *c0*, and *c5-fs*) are faster to apply to the test data than those trained on larger contexts.

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
ConsSE MB	c5-fs	8080,15	0,89	96,35	0,97	0,99	0,98
CfsSE Jrip	c5-fs	3018,82	0,20	96,34	0,98	0,99	0,98
CfsSE MB	c5-fs	609,11	1,64	95,63	0,99	0,98	0,98
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
MB	c1	831,6	18,93	93,96	0,99	0,95	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 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 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 3.6: Classifiers ordered according to number of percent correctly classified tokens in the input data.

3.4.3 Coverage

What is striking in Table 3.6 is that, overall, classifiers seem to perform better when trained on small contexts. The best performer is the *Jrip* rule learner when trained on intrinsic token representations. Most of the classifiers manage to classify more than 95 percent of the tokens in test portion of the data correctly. Three classifiers scored below 90 percent; the *NaiveBayesUpdatable* with and without automatic feature selection. Looking at Table 3.6, it is obvious why precision, recall, and F-score are not good for discriminating between the classifiers – the majority of results along these dimensions is in the vicinity of 0,98 or 0,99.

The *correlation-based feature selection* method is better than the *consistency-based feature selection* method in six of nine cases.

Remember that this is a token sequence labelling task and that the coverage results are not directly comparable with most other results reported for named entity recognition.

3.4.4 Combined results

The results from the three complete listings of the training time, the testing time, and the coverage were combined into one list using squared rank sum (see Table 3.7). By using rank sum, some of the items in the combined result list ended up with the same rank sum. In order to favor low ranks, i.e., items high on the lists, the squared rank sum approach was used. Essentially, 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 .

RANK	CLASSIFIER	CTX	TRAIN	TEST	PC	PREC	REC	F
1386	REPT	c0	122,31	0,10	97,68	0,99	0,99	0,99
2561	J48	c0	189,65	0,07	97,65	0,99	0,99	0,99
5565	CfsSE J48	c5-fs	367,69	0,11	97,28	0,99	0,99	0,99
7521	CfsSE REPT	c5-fs	446,37	0,12	97,32	0,99	0,99	0,99
9774	CfsSE PART	c5-fs	504,79	0,23	97,19	0,99	0,99	0,99
10298	Ibk	c0	4,20	305,02	97,12	0,99	0,99	0,99
12201	NB	c0	22,22	0,65	93,05	0,99	0,94	0,97
13706	CfsSE Ibk	c5-fs	284,92	340,04	97,17	0,99	0,99	0,99
14225	PART	c0	1295,23	0,30	97,43	0,99	0,99	0,99
16266	MB	c0	235,64	6,84	93,84	0,99	0,95	0,97
17221	CfsSE NB	c5-fs	435,46	0,17	93,13	0,99	0,95	0,97
18349	CfsSE MB	c5-fs	609,11	1,64	95,63	0,99	0,98	0,98
19466	Jrip	intr	4154,58	0,13	97,97	0,98	0,99	0,99
20110	CfsSE Jrip	c5-fs	1858,39	0,19	96,33	0,97	0,99	0,98
24221	NBUd	c0	10,04	5,71	81,28	0,99	0,85	0,92
25805	ConsSE J48	c5-fs	6040,74	0,11	96,67	0,98	0,99	0,98
27339	CfsSE NBUd	c5-fs	360,75	1,67	85,34	0,99	0,89	0,94
28874	ConsSE PART	c5-fs	6434,96	0,33	96,39	0,98	0,99	0,98
29053	RBF	intr	1249,56	2,81	91,30	0,96	0,98	0,97
29406	ConsSE REPT	c5-fs	7913,13	0,12	96,82	0,98	0,99	0,99
31656	ConsSE NB	c5-fs	6654,77	0,15	95,47	0,96	0,99	0,98
33267	ConsSE MB	c5-fs	8080,15	0,89	96,35	0,97	0,99	0,98
33434	ConsSE Ibk	c5-fs	5613,11	167,30	96,36	0,98	0,99	0,98
37757	ConsSE Jrip	c5-fs	9056,25	0,25	94,82	0,96	0,99	0,97
39717	CfsSE RBF	c5-fs	7499,59	1,41	91,79	0,92	1,00	0,96
43448	ConsSE NBUd	c5-fs	6178,32	0,37	86,21	0,91	0,96	0,93
49706	ConsSE RBF	c5-fs	18383,74	1,01	90,01	0,91	0,99	0,95

Table 3.7: Classifiers ordered according to squared rank sum.

The task at hand in this experiment — to as fast and correct as possible decide whether a token is a part of a name — can successfully be approached using decision trees. The two tree learners used, *REPTree* and *J48*, occupy the four first positions in Table 3.7.

The *correlation-based feature selection* method wins over *consistency-based feature selection* in all cases.

Given the feature representation described in Section 3.3.3, the classifiers all seem to prefer small context sizes.

General recommendations based on the outcome of this experiment for the future work with information extraction and named entity recognition within the COMPANIONS project is given in Chapter 4.

Chapter 4

Recommendations for Companions

4.1 Lexical Acquisition

Our recommendations for a potential lexicon acquisition module in COMPANIONS are the following:

- **Use distributional statistics to acquire lexical information.** The distributional methodology is a very attractive alternative for lexicon acquisition, since it is fully automatic, can be very efficiently implemented, is language independent, and requires a bare minimum of preprocessing.
- **Use the word-space methodology as implementation.** Using linear algebra as implementational basis allows us the benefit of a whole menagerie of tools to manipulate the representations. Also, a word space effectively constitutes an implicit lexical repository that can be updated real-time and consulted when lexical information is needed.
- **Use Random Indexing for dimensionality reduction in the word space.** In any system that requires on-line processing, efficiency and scalability are vital aspects. Matrix factorization techniques like SVD or PCA are known to be computationally expensive, and so are not ideal choices for such systems. Random Indexing, on the other hand, is designed to be scalable, efficient, and comparatively robust in comparison with matrix factorization techniques.

4.2 Named Entity Recognition

The following observations and recommendations stem from the named entity recognition experiment presented in Chapter 3, while they may apply to other data sets and other tasks as well, they should be used with utmost care in situations not resembling NER in textual data. Furthermore, the recommendations are based on the assumption that learning will take place on-line, i.e., while the user is waiting, or near on-line, e.g., when one of two systems are trained

in the background while the other one operates on-line (thus, the time required to train a system is still an important factor, but not as crucial as in on-line operation).

- **Use decision trees.** The decision tree learners used in the experiment outperformed the other classifiers when the results from measuring training time, testing time, and correctly classified tokens were combined.
- **A small token context is enough.** It seems as in treating named entity recognition as a token sequence labelling task and using the features introduced in Section 3.3.3, the machine learning methods performed well when using a small context.
- **Be careful when using automatic feature selection.** The cases where automatic feature selection methods have been used did not make it all the way to the top of the result lists. Thus, intital generation of large feature vectors with the intention of reducing the dimensionality using *correlation-based* or *consistency-based* feature selection prior to learning is not a viable path. The features used must be motivated at the time of the task definition.
- **Avoid using incremental learning methods.** The incremental learning methods examined here are *NaiveBayesUpdateable*, an incremental version of a naïve Bayesian learner, and *IBk*, a nearest neighbor lazy learner. Both methods learn fast, are relatively slow to apply on test data, and perform badly wrt to percent correctly classified tokens.

References

- Achlioptas, Dimitris. 2001. Database-friendly random projections. In *Symposium on Principles of Database Systems*.
- Aha, David W.; Kibler, Dennis and Albert, Marc K. 1991. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.
- Appelt, Douglas E. and Israel, David J. 1999. Introduction to Information Extraction Technology. Tutorial at the 16th International Joint Conference on Artificial Intelligence, August.
- Bagga, Amit. 1998. Analyzing the complexity of a domain with respect to an information extraction task. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*.
- Banko, Michele and Brill, Eric. 1986. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 11th International Conference on Computational Linguistics*, pages 26–33, Bonn, Germany, August. ACL.
- Ben-David, Shai; Blitzer, John; Crammer, Koby and Pereira, Fernando. 2007. Analysis of representations for domain adaptation. In Schölkopf, B.; Platt, J. and Hoffman, T. editors, *Advances in Neural Information Processing Systems 19*, pages 137–144. MIT Press, Cambridge, MA.
- Bikel, Daniel M.; Schwartz, Richard and Weischedel, Ralph M. 1999. An algorithm that learns what’s in a name. *Machine Learning*, 34:211–231.
- Blitzer, John; McDonald, Ryan and Pereira, Fernando. 2006a. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July. Association for Computational Linguistics.
- Blitzer, John; McDonald, Ryan and Pereira, Fernando. 2006b. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing, EMNLP’06*, Sydney, Australia.
- Borthwick, Andrew; Sterling, John; Agichtein, Eugene and Grishman, Ralph. 1998. NYU: Description of the MENE named entity system as used in MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Fairfax, VA, USA, April 29 - May 1.
- Breiman, Leo; Friedman, Jerome H.; Olshen, Richard A. and Stone, Charles J. 1984. *Classification And Regression Trees*. Brooks/Cole Publishing, Monterey, California.
- Brown, Peter; Cocke, John; Della Pietra, Stephen; Della Pietra, Vincent; Jelinek, Frederick; Mercer, Robert and Roossin, Paul. 1988. A statistical approach to language translation. In *Proceedings of the 12th Annual Conference on Computational Linguistics, COLING’88*. International Committee on Computational Linguistics.
- Brown, Peter; Cocke, John; Della Pietra, Stephen; Della Pietra, Vincent; Jelinek, Frederick; Lafferty, John; Mercer, Robert and Roossin, Paul. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

- Chelba, C. and Acero, A. 2006. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.
- Ciravegna, Fabio; Petrelli, Daniela and Wilks, Yorick. 2002. User-system cooperation in document annotation based on information extraction. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*, Sigüenza, Spain. Springer Verlag.
- Cohen, William W. 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, Tahoe City, California, July.
- Cohn, David; Atlas, Les and Ladner, Richard. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.
- Cowie, Jim and Lehnert, Wendy. 1996. Information Extraction. *Communications of the ACM*, 39(1):80–91, January.
- Daelemans, Walter and Hoste, Véronique. 2002. Evaluation of machine learning methods for natural language processing tasks. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation on Applied Natural Language Processing*, Las Palmas, Canary Islands, Spain, May. ELRA.
- Daumé, III, Hal. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic, June. ACL.
- Deerwester, Scott; Dumais, Susan; Landauer, Thomas; Furnas, George and Beck, Laura. 1988. Improving information retrieval with latent semantic indexing. In *Proceedings of the 51st Annual Meeting of the American Society for Information Science*, volume 25, pages 36–40, Atlanta, Georgia. Learned Information, Inc.
- Dumais, Susan. 2004. Latent semantic analysis. *Annual Review of Information Science and Technology (ARIST)*, 38:189–230.
- Finn, Aidan and Kushmerick, Nicolas. 2003. Active learning selection strategies for information extraction. In *Proceedings of the International Workshop on Adaptive Text Extraction and Mining held in conjunction with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Catvat, Dubrovnik, Croatia, September 22.
- Foltz, P.; Kintsch, W. and Landauer, T. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2/3):285–307.
- Frank, Eibe and Witten, Ian H. 1998. Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 144–151.
- Golub, Gene and Van Loan, Charles. 1996. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, USA.

- Granger, R. H. 1977. FOUL-UP: A program that figures out meanings of words from context. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 172–178, Cambridge, Massachusetts. IJCAI, Morgan Kaufmann.
- Grishman, Ralph. 1997. Information Extraction: Techniques and challenges. In Pazienza, Maria Teresa editor, *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, number 1299 in Lecture Notes in Artificial Intelligence, chapter 2, pages 10–27. Springer-Verlag, Berlin, Germany.
- Grosz, Barbara J.; Appelt, Douglas E.; Martin, Paul and Pereira, Fernando C. N. 1987. TEAM: An experiment in the design of transportable natural-language interfaces. *Artificial Intelligence*, 32:173–243.
- Hall, Mark A. and Holmes, Geoffrey. 2003. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering*, 15(3), May/June.
- Hall, Mark A. 1999. *Correlation-based Feature Subset Selection for Machine Learning*. Ph.D. thesis, Department of Computer Science, University of Waikato.
- Hofmann, Thomas and Puzicha, Jan. 1999. Latent class models for collaborative filtering. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI'99*, pages 688–693, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Hyvärinen, Aapo; Karhunen, Juha and Oja, Erkki. 2001. *Independent Component Analysis*. Wiley-Interscience.
- Jacobs, P. and Zernik, U. 1988. Acquiring lexical knowledge from text: A case study. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 739–744, Saint Paul, Minnesota, August. AAAI, MIT Press.
- John, George H. and Langley, Pat. 1995. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, CA, USA.
- Johnson, William and Lindenstrauss, Joram. 1984. Extensions of lipshitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206.
- Kanerva, Pentti; Kristofersson, Jan and Holst, Anders. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society, CogSci'00*, page 1036. Erlbaum.
- Kaski, Samuel. 1999. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN'98*, pages 413–418. IEEE Service Center.
- Landauer, Thomas and Dumais, Susan. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.
- LDC, . 2007. Ldc projects: Tides. Webpage.

- Lee, Daniel and Seung, Sebastian. 2000. Algorithms for non-negative matrix factorization. In *Proceedings of the 2000 Conference on Advances in Neural Information Processing Systems, NIPS'00*, pages 556–562.
- Levy, Joseph; Bullinaria, John and Patel, Malti. 1998. Explorations in the derivation of word co-occurrence statistics. *South Pacific Journal of Psychology*, 10(1):99–111.
- Linguistic Data Consortium, . 2001. Message understanding conference (muc) 7. LDC2001T02. FTP FILE. Philadelphia: Linguistic Data Consortium.
- Liu, H. and Setiono, R. 1996. A probabilistic approach to feature selection - a filter solution. In *Proceedings of the 13th International Conference on Machine Learning*, pages 319–327.
- Lund, Kevin; Burgess, Curt and Atchley, Ruth Ann. 1995. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society, CogSci'95*, pages 660–665. Erlbaum.
- Melamed, D. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.
- Mikheev, Andrei and Finch, Steven. 1995. Towards a workbench for acquisition of domain knowledge from natural language. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, pages 194–201, University College of Dublin, Dublin, Ireland, March. ACL.
- Miller, George A. 1990. WordNet: An online lexical database. *International Journal of Lexicography*, 3(4):235–312. Special issue.
- NIST, . 2007. Automatic content extraction (ace). Webpage.
- PASCAL, . 2007. Pattern analysis, statistical modelling and computational learning (pascal). Webpage.
- Powell, M. J. D. 1987. Radial basis functions for multivariable interpolation: A review. In Mason, J. and Cox, M. editors, *Algorithms for approximation*, pages 143–167. Oxford: Clarendon Press.
- Quinlan, J. Ross. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California.
- Ramshaw, Lance A. and Marcus, Mitchell P. 1995. Text Chunking using Transformation Based Learning. In *ACL Third Workshop on Very Large Corpora*, pages 82–94, June.
- Rapp, Reinhard. 2003. Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of MT Summit IX*, pages 315–322.
- Rubenstein, Herbert and Goodenough, John. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.
- Sahlgren, Magnus and Cöster, Richard. 2004. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING'04*, pages 487–493.

- Sahlgren, Magnus and Karlgren, Jussi. 2005. Automatic bilingual lexicon acquisition using random indexing of parallel corpora. *Journal of Natural Language Engineering*, 11(3):327–341.
- Sahlgren, Magnus; Karlgren, Jussi; Cöster, Rickard and Järvinen, Timo. 2003. SICS at CLEF 2002: Automatic query expansion using random indexing. In Peters, Carol; Braschler, Martin; Gonzalo, Julio and Kluck, Michael editors, *Advances in Cross-Language Information Retrieval, 3rd Workshop of the Cross-Language Evaluation Forum, CLEF'02. Rome, Italy, September 19-20, 2002, Revised Papers*, Lecture Notes in Computer Science, pages 311–320. Springer.
- Sahlgren, Magnus. 2005. An introduction to random indexing. In Witschel, H.F. editor, *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE'05, Copenhagen, Denmark, August 16, 2005*, volume 87 of *TermNet News: Newsletter of International Cooperation in Terminology*.
- Sahlgren, Magnus. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD Dissertation, Department of Linguistics, Stockholm University.
- Schank, Roger C. 1975. SAM — a story understander. Research Report 43, Dept. of Computer Science, Yale University, New Haven, Connecticut.
- Schütze, Hinrich and Pedersen, Jan. 1995. Information retrieval based on word senses. In *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175.
- Schütze, Hinrich. 1993. Word space. In *Proceedings of the 1993 Conference on Advances in Neural Information Processing Systems, NIPS'93*, pages 895–902, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Science Applications International Corporation, . 2000. The message understanding conference series (MUC). Webpage.
- Shen, Dan; Zhang, Jie; Su, Jian; Zhou, Guodong and Tan, Chew-Lim. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting and the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 589–596, Barcelona, Spain, July. ACL.
- Thompson, Cynthia A.; Califf, Mary Elaine and Mooney, Raymond J. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning*, Bled, Slovenia, June.
- Turmo, Jordi; Ageno, Alicia and Català, Neus. 2006. Adaptive information extraction. *ACM Computing Surveys*, 38(2), July.
- Webb, Geoffrey I. 2000. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2).
- Witten, Ian H. and Frank, Eibe. 2005. *Data Mining: Practical machine learning tools with Java implementations. 2nd Edition*. Morgan Kaufmann, San Francisco.

Wolfe, Michael; Schreiner, M.E.; Rehder, Bob; Laham, Darrell; Foltz, Peter; Kintsch, Walter and Landauer, Thomas. 1998. Learning from text: Matching readers and text by latent semantic analysis. *Discourse Processes*, 25:309–336.

Yang, Yiming and Pedersen, Jan. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning, ICML'97*, pages 412–420.

Yangarber, Roman and Grishman, Ralph. 1997. Customization of Information Extraction systems. In *Proceedings of the International Workshop on Lexically Driven Information Extraction*, Frascati, Italy.

Zhou, Guodong and Su, Jian. 1999. Machine learning-based named entity recognition via effective integration of various evidences. *Natural Language Engineering*, 11(2):189–206.

Appendices

Appendix A

Parameter settings

This appendix describes the parameter settings used for each of the “best” machine learning methods employed in Chapter 3. The parameters are to be understood as the options set in Weka when defining the experiments in the Weka Experimenter environment.

A.1 Time to train

The listing in this section corresponds to table 3.4 in Chapter 3. The commands in the listing have been cleaned-up in order to improve readability, thus they do not constitute valid Weka command lines. Where applicable, the seeds used have been set as they appear in the Weka Experimenter.

- **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
- **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 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**
- **meta.AttributeSelectedClassifier -E CfsSubsetEval -S BestFirst -D 1 -N 5 -W meta.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -D**
- **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**
- **meta.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -K**
- **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 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 meta.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -D**
- **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.2 Time to test

The listing in this section corresponds to table 3.5 in Chapter 3. The commands in the listing have been cleaned-up in order to improve readability, thus they do not constitute valid Weka command lines. Where applicable, the seeds used have been set as they appear in the Weka Experimenter.

- **trees.REPTree** -M 2 -V 0.0010 -N 3 -S 1 -L -1 -P -8.5624434286215393E18
- **trees.J48** -C 0.05 -M 2 -2.17733168393644448E17
- **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
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **bayes.NaiveBayesUpdateable**
- **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **ConsistencySubsetEval** -S BestFirst -D 1 -N 5 -W **meta.MultiBoostAB** -C 3 -P 100 -S 1 -I 10 -W **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 meta.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -D**
- **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.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -D**
- **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.3 Coverage

The listing in this section corresponds to table 3.6 in Chapter 3. The commands in the listing have been cleaned-up in order to improve readability, thus they do not constitute valid Weka command lines. Where applicable, the seeds used have been set as they appear in the Weka Experimenter.

- **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 ConsistencySubsetEval -S BestFirst -D 1 -N 5 -W meta.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -D**
- **meta.AttributeSelectedClassifier -E CfsSubsetEval -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 meta.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -D**
- **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.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -D**
- **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.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 bayes.NaiveBayesUpdateable**
- **meta.AttributeSelectedClassifier -E CfsSubsetEval -S BestFirst -D 1 -N 5 -W bayes.NaiveBayesUpdateable**
- **bayes.NaiveBayesUpdateable**

A.4 Combined results

The listing in this section corresponds to table 3.7 in Chapter 3. The commands in the listing have been cleaned-up in order to improve readability, thus they do not constitute valid Weka command lines. Where applicable, the seeds used have been set as they appear in the Weka Experimenter.

- **trees.REPTree** -M 2 -V 0.0010 -N 2 -S 1 -L -1 -P
- **trees.J48** -S -C 0.25 -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
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S **BestFirst** -D 1 -N 5 -W **rules.PART** -M 2 -C 0.25 -Q 1
- **lazy.IBk** -K 5 -W 0 -I -A **weka.core.KDTree** -A **weka.core.EuclideanDistance** -W 0.01 -L 40
- **bayes.NaiveBayes** -D
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S **BestFirst** -D 1 -N 5 -W 0 -W **lazy.IBk** -K 2 -W 0 -A **weka.core.LinearNN** -A **weka.core.EuclideanDistance**
- **rules.PART** -M 6 -C 0.25 -Q 1
- **meta.MultiBoostAB** -C 3 -P 100 -S 1 -I 10 -W **bayes.NaiveBayes** -D
- **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 **meta.MultiBoostAB** -C 3 -P 100 -S 1 -I 10 -W **bayes.NaiveBayes** -D
- **rules.JRip** -F 3 -N 2.0 -O 2 -S 1
- **meta.AttributeSelectedClassifier** -E **CfsSubsetEval** -S **BestFirst** -D 1 -N 5 -W **rules.JRip** -F 3 -N 2.0 -O 2 -S 1
- **bayes.NaiveBayesUpdateable**
- **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 **bayes.NaiveBayesUpdateable**

- **meta.AttributeSelectedClassifier -E ConsistencySubsetEval -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**
- **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 ConsistencySubsetEval -S BestFirst -D 1 -N 5 -W meta.MultiBoostAB -C 3 -P 100 -S 1 -I 10 -W bayes.NaiveBayes -D**
- **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 rules.JRip -F 3 -N 2.0 -O 2 -S 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 ConsistencySubsetEval -S BestFirst -D 1 -N 5 -W bayes.NaiveBayesUpdateable**
- **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**