UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of master's thesis by

MAHESH JOSHI

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Richard Maclin

Name of Faculty Adviser

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

Kernel Methods for Word Sense Disambiguation and Abbreviation Expansion in the Medical Domain

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Mahesh Joshi

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

August 2006

Abstract

Word Sense Disambiguation (WSD) is the problem of automatically deciding the correct meaning of an ambiguous word based on the surrounding context in which it appears. The automatic expansion of abbreviations having multiple possible expansions can be viewed as a special form of WSD with the multiple expansions acting as ”senses” of the ambiguous abbreviation. Both of these are significant problems, especially in domains such as medical text with abstracts of articles in scholarly medical journals and clinical notes taken by physicians.

Most popular approaches to WSD involve supervised machine learning methods which require a set of manually annotated examples of disambiguated words or abbreviations to learn patterns that help in disambiguating future unseen instances. However, manual annotation imposes a limit on the amount of labeled data that can be made available to the supervised machine learning algorithms such as Support Vector Machines (SVMs), because annotation requires significant human work. Kernel methods for SVMs provide an elegant framework to incorporate knowledge from unlabeled data into the SVM learners.

This thesis explores the application of kernel methods to two datasets from the medical domain, one containing ambiguous words and the other containing ambiguous abbreviations. We have developed two classes of semantic kernels - Latent Semantic Analysis (LSA) Kernels and Word Association Kernels (ASSOC) for SVMs, that are learned from unlabeled text containing the ambiguous words or abbreviations using unsupervised methods. We have found that our semantic kernels improve the accuracy of SVMs on the task of WSD in the medical domain. In particular, we find that our LSA kernels with unigram features and ASSOC kernels with bigram features perform better than off-the-shelf SVM learners and they are significantly better for five out of 11 ambiguous words, which have a balanced sense distribution and for nine out of ten abbreviations.

We also focus on the feature engineering aspect of the abbreviation expansion problem in the domain of clinical notes text. We propose a flexible window approach for capturing features and show that it significantly improves performance. We make use of features specific to clinical notes such as the gender code of the patient and show that this improves performance significantly in combination with Part-of-Speech features.

Acknowledgments

I would like to thank my advisors Dr. Richard Maclin and Dr. Ted Pedersen for their immense and continual guidance. I started off at the University of Minnesota Duluth with an empty slate with respect to the fields of Machine Learning and Natural Language Processing. Dr. Maclin and Dr. Pedersen have taught me the basics of how to go about doing research and have truly inspired in me the spirit required to do research, by setting examples and patiently guiding me on several matters, be they technical or non-technical. Special thanks are due to both of them, to Dr. Maclin for adjusting his schedules to facilitate early completion of my thesis defense and for several painstaking reviews to get the thesis into its current state, and to Dr. Pedersen for being actively involved in my research throughout my two years at UMD, encouraging and facilitating my attendance at leading international conferences and for being extremely accommodating about my defense schedule.

I would like to thank my thesis committee member Dr. Barry James, for the the many things that I have learned while taking his course "Probability," which provided me with a sound statistical background for my research.

I would like to take this opportunity to thank the Department of Computer Science at UMD for providing an encouraging environment for graduate studies. I would also like to thank the Director of Graduate Studies in Computer Science Dr. Carolyn Crouch and Dean James Riehl of College of Science and Engineering for the generous financial grants provided to support my conference travel.

I extend my gratitude to the department staff including Lori Lucia and Linda Meek for facilitating all official matters and to our system administrator Jim Luttinen for impeccably maintaining the computing facilities at the department.

I was fortunate to have had an opportunity to work as a summer intern at the Mayo Clinic, Division of Biomedical Informatics. I would like to thank my internship supervisor Dr. Serguei Pakhomov for providing the right balance of supervision and independence during my summer research. I would also like to thank Dr. Christopher Chute and James Buntrock for providing me this internship opportunity. Sincere thanks to the annotators Barbara Abbott, Debra Albrecht and Pauline Funk at the Mayo Clinic, Division of Biomedical Informatics for annotating the abbreviation datasets that I have used for my summer research. I would also like to thank the Mayo Clinic for permitting the use of clinical notes data for my summer research and inclusion of the results in my thesis work.

This work was carried out in part using hardware and software provided by the University of Minnesota Supercomputing Institute.

I am thankful to Dr. Alan Aronson and James Mork from the National Library of Medicine (NLM) for providing the unlabeled data associated with the NLM word sense disambiguation collection, which was extensively used for our experiments. I am also thankful to Dr. Hongfang Liu for providing the MEDLINE abbreviation data collection, which was used in our experiments.

Thanks to everyone in the Natural Language Processing group at UMD, including Dr. Pedersen, Anagha Kulkarni, Saiyam Kohli, Apurva Padhye, Jason Michelizzi and Pratheepan Raveendranathan. Discussions in the group meeting have helped me a lot in terms of getting others' feedback on my work, raising interesting questions and providing an excellent opportunity to build confidence in presenting my work.

I would like to thank Lisa Fitzpatrick from the Visualization and Digital Imaging lab at UMD for providing a very supportive work environment in the lab.

I would also like to thank all my friends and colleagues at the Department of Computer Science including Lalit Nookala, Aditya Polumetla, Murthy Ganapathybhotla, Saiyam Kohli, Vishal Bakshi, Sameer Atar, Kedar Bhumkar and others for attending my practice presentations for conference talks, providing feedback and for all the fun times we have had together.

Finally, although I cannot ever extend enough thanks to her, I would like to to express my love and thanks toward my dearest friend, philosopher and guide, my wife Anagha. She has been extremely supportive and patient throughout, encouraging in difficult times, genuinely interested and contributive in my work and certainly the most adorable person for me all along.

Contents

List of Figures

List of Tables

1 Introduction

The English language has many words that have multiple meanings. For example, the word *switch* in the sentence "*Turn off the main switch*" refers to an electrical object whereas in the sentence "*The hansom driver whipped the horse using a switch*" it refers to a flexible twig or rod[1]. As can be observed in the above examples, the correct sense is usually made clear by the context in which the word has been used. Specifically, in the first sentence, the words *turn, off* and *main* combined with some world knowledge of the person interpreting the sentence, for example, the fact that usually there is a main *switch* for electrical connections inside a house, help in disambiguating the word, that is, assigning the correct sense to the word. Similarly, in the second sentence the words *hansom, driver, whipped* and *horse* serve the purpose of defining the appropriate context which helps in understanding the correct sense of the word *switch*.

Word sense disambiguation (WSD) [23] is the problem of automatically assigning the appropriate meaning to a word having multiple senses. So in our example above, given the ambiguous word *switch*, WSD involves interpreting the surrounding context of the word and analyzing the properties exhibited by the context to determine the *right* sense of *switch*.

Automatic abbreviation expansion [44] is a special case of WSD where an ambiguous abbreviation is to be assigned the correct expansion based on its surrounding context. For example, the acronym *PVC* could have two possible expansions *Polyvinyl Chloride* and *Premature Ventricular Contraction*. An occurrence of *PVC* in the sentence "*Atenolol is an alternative drug in the treatment of PVC in patients with coronary heart disease*" can be expanded as *Premature Ventricular Contraction* based on the presence of the words *coronary* and *heart*. In this work, we will use the word *disambiguation* to refer to WSD as well as abbreviation expansion.

It may be sometimes perceived that ambiguity is less of a problem in more specialized domains, for example, the word *virus* can be ambiguous if the domain is not known, and can refer to a *computer virus* or a *disease causing agent*; but *virus* is not ambiguous if we know that the domain under consideration is that of medicine. However, we have observed that ambiguity still remains

[1]According to the Merriam-Webster Dictionary online: http://www.m-w.com/cgi-bin/dictionary?book=Dictionary&va=switch

a significant problem even in the specialized domain of medicine. For example, *radiation* could be used to mean the property of *electromagnetic radiation*, or as a synonym for *radiation therapy* for treatment of a disease. While both of these senses are somewhat related, the therapy relies on the radioactive property, there are also words like *cold*, which can mean the temperature of a room, or an illness. Thus, even more specialized domains exhibit a full range of ambiguities. Weeber et al., [54] have gathered a dataset containing 50 ambiguous words from the abstracts of scholarly articles from biomedicine related journals, which demonstrates the difficulty of this problem.

As noted by Weeber et al., [54], the linguistic interest in medical domain arises out of the need for better natural language processing (NLP) systems used for decision support or document indexing for information retrieval. Such NLP systems will perform better if they are capable of resolving ambiguities among terms. For example, with the ability to disambiguate senses, an information retrieval query for *radiation therapy* will ideally be capable of getting only those documents that contain the word *radiation* in the "medical treatment" sense.

Most work in word sense disambiguation has focused on general English [7, 19, 23, 30, 42, 43]. Here we propose to study word sense disambiguation in the medical domain and evaluate how well existing techniques perform, and introduce refinements of our own based on this experience. We initially present our baseline results originally published in Joshi et al., [28] and expand upon them using the idea of semi-supervised semantic kernels for Support Vector Machines.

Most popular approaches to the problems of WSD and abbreviation expansion make use of supervised machine learning methods, which require a set of manually labeled or sense-tagged training instances of the word or acronym to be disambiguated. The amount of labeled data required to generate a robust model (i.e., a model which is accurate and generalizes well) using any learning algorithm is usually quite large, the lack of which imposes a significant limitation on the knowledge that a learning algorithm can acquire. This is the known as the so-called knowledge acquisition bottleneck.

A large corpus of unlabeled text is easier to obtain as compared to manually labeled data for a supervised machine learning algorithm. Unsupervised learning methods can make use of unlabeled text to deduce similarity relationships among words and documents. Any machine learning algorithm that can make use of such similarity relationships has the potential to benefit from a large corpora of unlabeled text. Kernel methods for Support Vector Machines (SVMs) provide exactly such a mechanism to incorporate external knowledge from various sources to improve the accuracy of the SVM learner.

The goal of this thesis is to evaluate kernel methods for WSD using existing unsupervised methods proposed by Purandare and Pedersen [49] and their extension in the form of a variation in the data representation to evaluate the similarity of contexts of an ambiguous word. We have developed two classes of semantic kernels that incorporate unlabeled text in a semi-unsupervised fashion. The first class of kernels are the Latent Semantic Analysis (LSA) [29] kernels and the second are the Word Association kernels. The difference among these two types of kernels is the data representation of the unlabeled instances.

Our goal is also to evaluate the feature engineering aspect of the abbreviation expansion problem in the medical domain. This includes the idea of using a variable-sized context of the ambiguous word for identifying features. We call this the *flexible window* approach, which does not physically restrict the position of a feature with respect to the ambiguous word. Further, we have experimented with using features that are specific to the domain of clinical notes from physicians, such as the gender code of the patient and the department code of the department from where the clinical note originated.

Our general findings are that although kernel methods using unlabeled text can significantly improve performance of WSD in the medical domain, they are not advantageous under all circumstances. They are highly dependent on the quality of unlabeled data used for generating the kernels, rather than quantity. In fact, a large quantity of low quality data can degrade the performance of an SVM learner that makes use of such data. We have found that our semantic kernels improve the accuracy of SVMs on the task of WSD in the medical domain. In particular, both LSA kernels and Word Association kernels perform significantly better when the sense distribution for an ambiguous word is balanced, that is a comparable number of labeled examples for each sense of the ambiguous word are available.

Our feature engineering experiments show that the flexible window approach improves the performance significantly for the task of abbreviation expansion. We also find that increasing the window size beyond two or three words for abbreviations in the clinical notes does not significantly improve accuracy and hence localized features are the best for clinical notes, which is quite intuitive as clinical notes are typically transcripts of physicians' diagnosis notes and tend to have short sentences or sentence fragments providing a concise context. Finally, the use of features specific to clinical notes also improves accuracy for abbreviation expansion and in combination with other syntactic features such as Part-of-Speech tags they yield significant improvement over basic features such as words and word pairs in context of the ambiguous abbreviation.

Overall, the contributions of this thesis categorized into research oriented contributions and infrastructure oriented contributions are as follows:

Research Oriented Contributions

- Development of two classes of semantic kernels derived from unlabeled text
  - Latent Semantic Analysis based kernels
  - Word Association based kernels
- Evaluation of our semantic kernel on two medical domain datasets, demonstrating improvements over default linear kernel Support Vector Machines
- A novel flexible window approach for feature extraction
- Feature engineering for abbreviation expansion in clinical notes
  - Use of features specific to clinical notes, such as gender code, department code and section identifier, in combination with Part-of-Speech features
  - Evaluation of our flexible window approach, demonstrating significant improvement using the same

Infrastructure Oriented Contributions

- Development of two word sense disambiguation toolkits

  - WSDShell[2], which is a Perl based toolkit that wraps the functionality of Ngram Statistics Package (NSP) [5] and SenseTools[3] for feature identification and extraction and the WEKA Data Mining Toolkit [55] for supervised WSD.

  - WSDGate [27][4], which is a Java based toolkit that integrates in the GATE (General Architecture for Text Engineering) [10] environment, wrapping GATE and NSP for feature identification and extraction and WEKA for supervised WSD. This toolkit provides the flexible window feature extraction capability.

- Development of NSPGate[5], which is a Java based wrapper for NSP and integrates the NSP functionality in GATE.

- Extensions of scripts from the SenseClusters [49] [6] package to generate kernels from unlabeled data

- Addition of a new user defined kernel function to SVM Light [25], making use of the kernels derived using SenseClusters

[2]http://www.d.umn.edu/∼tpederse/wsdshell.html

[3]http://www.d.umn.edu/∼tpederse/sensetools.html

[4]http://wsdgate.sourceforge.net/

[5]http://nspgate.sourceforge.net/

[6]http://senseclusters.sourceforge.net/

2 Background

This chapter covers background topics including a formal introduction to the problems of word sense disambiguation and abbreviation expansion and some motivation as to why these problems are important in the medical domain. It then provides a general overview of the use of machine learning for solving these problems and introduces some commonly used features.

It then describes five specific machine learning algorithms including the naïve Bayes classifier, decision trees, decision lists, Support Vector Machines and the AdaBoostM1 ensemble approach. Finally it provides a description of kernel methods for Support Vector Machines and explains the concept of semantic kernels, which are directly related to our approach.

2.1 Word Sense Disambiguation

Many words in the English language are potentially ambiguous, that is they have multiple possible meanings. However, as humans, we rarely face the problem of ambiguity resolution since we can quickly decide the correct meaning based on the context in which the ambiguous word has been used. The same task however is extremely hard for computers to accomplish. Automatic methods to decide the correct meaning of an ambiguous word are known as Word Sense Disambiguation methods.

Word Sense Disambiguation (WSD) [23] is the process of assigning a unique correct meaning to an ambiguous word that occurs in a given context. The correct meaning is chosen from a set of possible meanings or senses for the ambiguous word, which are usually selected from some standard dictionary. Such a set of meanings for an ambiguous word is known as its *sense inventory*. For example, the New Oxford American Dictionary shows one possible sense inventory containing eight meanings for the ambiguous word *ring*:

1. a small circular band, typically of precious metal
2. a thin band or disk of rock or ice particles around a planet
3. a circular marking or pattern

4. an enclosed space typically surrounded by spectators
5. a group of people drawn together due to a shared interest or goal
6. a set of elements with two mathematical operations, addition and multiplication
7. make a clear resonant or vibrating sound
8. call by telephone

There are two things to note about this sense inventory. First, that the senses 1, 2 and 3 are actually a finer distinction of the broader sense of *ring* referring to the central idea of something being *circular*. The other senses can be distinguished from these and among themselves at a more coarse level. Second, the senses 1 through 6 can only be used when *ring* is a noun, whereas senses 7 and 8 can be used when *ring* is a verb.

WSD is further categorized into: (i) target-word sense disambiguation and (ii) all-words sense disambiguation. In target-word sense disambiguation (also known as a *lexical sample task*), the set of words to be disambiguated is decided in advance and for each such word, a set of sentences or paragraphs containing the ambiguous word are collected. These are known as *lexical samples* for the given ambiguous word. In an all-words sense disambiguation task, no such predetermined set of words is disambiguated, rather, the aim is to assign a meaning to each word in any given document or set of documents.

2.2 Abbreviation Expansion

The New Oxford American Dictionary defines an *abbreviation* as *a shortened form of a word or phrase*. For example "*abbr.*" is the abbreviation for the word "*abbreviation*. An *acronym* is defined in the New Oxford American Dictionary as as *a word formed from the initial letters of other words*. For example, *APC* is an acronym for *Atrial Premature Complexes*. Acronyms can therefore be considered as a special class of abbreviations. We will henceforth use the term "abbreviation" to refer to the broader category that includes acronyms.

A problem similar to WSD and also widely prevalent is that of automatic abbreviation expansion. Many abbreviations have multiple expansions and deciding the correct expansion of an abbreviation based on the context in which it appears is very much like deciding the correct sense of an ambiguous word depending on the context in which it is used. Automatic expansion of ambiguous acronyms is a challenge for computers, just like WSD. Abbreviation expansion is the problem of automatically deciding the correct expansion of an ambiguous abbreviation, based on its surrounding context. It is considered a variation of WSD, where the multiple expansions of an abbreviation can be considered as its "senses." However, the difference from WSD arises due to the fact that usually the expansions of an abbreviation have fairly coarse-grained distinctions and cannot form a hierarchy of meanings, which is quite common for word senses. An example of an ambiguous abbreviation in the abstracts of scholarly journal articles in the MEDLINE[7] bibliographical database is *APC*. The sense inventory for *APC* as defined in Liu et al., [35] is:

1. Antigen-Presenting Cells
2. Adenomatous Polyposis Coli
3. Atrial Premature Complexes
4. Aphidicholin
5. Activated Protein C

One point to note about sense inventories in general (whether for ambiguous words or for ambiguous abbreviations) is that the sense inventory used for the disambiguation task is usually dependent upon the corpus of data that is used for disambiguation - that is the different senses of an ambiguous word or abbreviation present in the experimental corpus decide its sense inventory for the task of disambiguation. For example, the same abbreviation *APC* had the following sense inventory with 10 expansions, for the experiments in Pakhomov [44]:

1. Adenomatous Polyposis Coli
2. Argon Plasma Coagulation
3. Atrial Premature Contraction

[7]http://www.nlm.nih.gov/pubs/factsheets/medline.html

4. Aspirin-Phenacetin-Caffeine
5. Activated Protein C
6. Adenomatous Polyposis gene
7. Allograft Prosthetic Composite
8. Anterior-Posterior Compression
9. Atrial Premature Complex
10. Antigen Presenting Cells

The difference in the two sense inventories above arises due to the different experimental data used by Liu et al., [35] and Pakhomov [44].

As mentioned before, the problem of abbreviation expansion is very similar to the problem of WSD and similar methods are applicable for solving both problems. Here onward we will refer to WSD as an umbrella term that refers to both word sense disambiguation as well as abbreviation expansion, unless stated otherwise explicitly.

2.3 WSD in the Medical Domain

Word Sense Disambiguation has been studied to a great extent in the domain of general English text; see Ide and Véronis [23] for an extensive overview. Although there have been a few studies on WSD and abbreviation expansion in the medical domain in the recent years [35, 44, 45], they are still comparatively much lesser than those for the domain of general English text. Schuemie et al., [51] contains an overview of WSD in the biomedical domain and mentions the scarcity of manually labeled data as one of the causes of less research being done in this domain. Therefore methods that can make use of small amounts of labeled data and can take advantage of the large amount of unlabeled data available in the domain in the form of abstracts of scholarly journal articles and in some cases data internal to the health organizations are of great value.

One might perceive that ambiguity is not a problem once we narrow down to a particular domain of text. However, the data collection of 50 ambiguous words from the abstracts of medical journal articles done by Weeber et al., [54] shows that even a specialized domain such as the medical domain can exhibit a full range of word ambiguities. In addition to that, it has long been known that acronyms and abbreviations are widely used in the medical domain and their ambiguous nature poses a serious problem [2]. Liu et al., [34] note that over 33% of the abbreviations with six characters or less in the UMLS [52] are ambiguous. In a further study, Liu et al., [32] show that 81% of the abbreviations in MEDLINE abstracts are ambiguous and have on average 16 senses. With such widely prevalent ambiguities and the sensitive nature of the medical domain, misinterpretation of an ambiguity can be potentially harmful by way of an incorrect drug being prescribed or incorrect diagnosis being done for a patient. Therefore WSD and abbreviation expansion are of great importance in this domain.

2.4 The Field of Machine Learning

Learning can be defined as improving one's performance on a given task with the aid of prior experience [39]. One way of making computers learn involves training machine learning algorithms with the help of an initial set of training data. The *experience* that the machine learning algorithms gain from the training data can then be applied to make predictions about previously unseen data. For example, given 100 sentences containing the ambiguous word *ring* along with the correct sense of *ring* in each of those sentences, one can train a machine learning algorithm such as the naïve Bayes classifier to disambiguate occurrences of the ambiguous word *ring*. Such a trained classifier can then take as input previously unseen sentences containing the word *ring*, and predict the correct sense of *ring* in those sentences.

Learning is further categorized as *supervised* or *unsupervised*. In *supervised* learning, a *teacher* provides the correct labels or outputs (such as the category or some numerical outcome) for the training data. Normally this means that the training data is manually labeled with the appropriate annotation, such as the occurrence of an ambiguous word in a sentence being assigned the correct sense. So the example above involving the ambiguous word *ring* and the naïve Bayes classifier is an example of *supervised* learning. In *unsupervised* learning, there is no teacher involved, the correct label for the training data instances is not available. One form of *unsupervised* learning is *clustering* where the goal of the machine learning algorithms is to partition the training data into coherent sets (known as *clusters*) of instances based on their similarity. Any new instance is evaluated by the algorithm to decide into what cluster it should be categorized. If in the above supervised learning example, the correct sense of the ambiguous word *ring* was not known in each of the 100 sentences, then the challenge for a clustering algorithm would be to firstly identify the number of senses of the word *ring* in the 100 sentences, and then cluster them into those many clusters. Given a new sentence containing the word *ring*, it should be assigned to one of the clusters formed from the training data (or even create a new cluster of its own if the algorithm concludes that the new sentence does not fit well into any of the existing clusters).

The advantage of supervised learning is that high accuracy can be obtained on unseen instances given that a sufficient amount of manually labeled training data is provided to generate a good model. The drawback of the supervised learning approach is that manually labeled data is highly expensive to generate in terms of time as well as money. Unsupervised methods benefit from the fact that they do not require manually labeled data. However, they usually suffer from low accuracy values on unseen instances. A hybrid approach to learning, popularly known as *semi-supervised* learning aims to capture the benefits of both supervised and unsupervised learning by making use of a small amount of labeled data to increase the accuracy and large amount of unlabeled data to effectively reduce the amount of labeled data required for generating accurate models.

2.5 Supervised Word Sense Disambiguation

Most popular approaches to WSD involve *supervised* machine learning. We now elaborate on some aspects of supervised machine learning with respect to WSD and introduce some terminology.

A set of data items is referred to as a *dataset*. Each element in the dataset is referred to as a data *instance*. For example, in the case of WSD, an instance can be a sentence containing the ambiguous word under consideration. Every instance has a finite number of properties associated with it, that are referred to as *attributes* or *features*. For example, in a WSD dataset, each instance might contain 10 attributes or features, which are the 5 words to the left and right of the ambiguous word. These features are popularly known as Bag-of-Word (BoW) features, which we discuss in the next subsection. The features in a dataset can have *discrete* or *continuous* values. BoW features are an example of discrete valued features. Discrete valued features are also referred to*nominal* features, where the possible set of values that they can take is known in advance. Continuous features on the other hand can take any real value and therefore their set of values cannot be enumerated or known in advance. As mentioned before, supervised learning requires an initial dataset for training the machine learning algorithms. This dataset is referred to as the *training data*. The implicit assumption in supervised learning is that the training data is *labeled*, that is every instance in the training data is associated with an output value or *label* that can be thought of as a special attribute or feature for each instance. For WSD, every instance in the training data should be assigned a label that corresponds to the correct sense of the ambiguous word that the instance contains or represents. Labels for a given dataset can have a finite number of discrete values (such as the different senses of an ambiguous word in a WSD dataset) or a continuous value (such as the temperature in a weather forecasting dataset). Machine learning algorithms make use of the instance attributes or features in the training data and generate a *model* to predict the label of any given instance. This model can be applied to unseen instances to predict their labels. Algorithms that can learn to predict discrete valued labels are called as *classification algorithms* or *classifiers*, whereas the algorithms that can learn to predict continuous valued labels are called *regression algorithms*. As the task of WSD only involves discrete valued labels for word senses, we use only classification algorithms.

Automatic methods for WSD rely primarily on the surrounding context of the word (i.e., the other words in its vicinity and their various properties) to identify distinguishing characteristics that enable disambiguation. This idea originally comes from Firth, [15] whose famous quote goes "*You shall know a word by the company it keeps.*" The properties of the surrounding context used by WSD methods are known as *features* used for the task of WSD. Several types of features have been proposed and used in the WSD literature. The most common type of features used are lexical features, where word tokens in the context of an ambiguous word are used as features. They are also known as the so-called Bag-of-Word (BoW) features since their position with respect to the ambiguous word does not matter, only their presence or absence is important. For example, in the sentence

"Frodo slips the *Ring* on and disappears."

containing the ambiguous word *Ring*, the BoW features are:

{Frodo,slips, the, on, and, disappears}

Note that although the above set lists the words in the order they appear in the sentence, the order does not matter for Bag-of-Word features.

Usually, function words or closed-class words such as articles (e.g., a, an, the), conjunctions (e.g., and, or) and prepositions (e.g., in, on) are excluded while identifying BoW features. Function words are also commonly known as *stop words*. The BoW features for the above sentence, excluding the function words would be:

{Frodo, slips, disappears}

Further, morphological variations of the BoW features are sometimes reduced to their root form and this can be useful in identifying similar features if the only difference is due to morphological variations. As an example,

"Frodo slips the *Ring* on and disappears."

will have BoW features as shown above, while

"Frodo slipped the *Ring* on and disappeared."

will have the features:

{Frodo, slipped, disappeared}.

If we analyze the similarity of these two contexts on the basis of number of overlapping words, we will find that only one out of three words overlaps. However, removing the morphological variations in both the sets of features will reduce them to the same set:

{Frodo, slip, disappear}

and yield an overlap of three out of three words, showing that the two contexts are indeed very similar.

Another variation on the BoW features is limiting the size of the context from which BoW features are identified. For example, if we limit the context to a window of two words around the ambiguous word *Ring* above, then after eliminating function words and removing morphological variations, the feature vector that we will obtain is:

{slip}

Apart from single-word tokens, two or more contiguous words in the context of an ambiguous word are also used as lexical features. Such features are termed as *collocations*. Continuing with the example used above, the two-word collocation features for the same are:

{Frodo slips, slips the, the Ring, Ring on, on and, and disappears}.

Note that we have not excluded the stop words in this case. There can be various approaches to decide which collocations should be eliminated based on the presence of stop words in them. One approach can be to only eliminate a collocation if all of its words are stop words. The above feature set would then reduce to:

{Frodo slips, slips the, the Ring, Ring on, and disappears}

Another approach can be removing all the collocations in which any of the words is a stop word. This would yield a very small feature set of just one collocation "Frodo slips" in our example. Other variations of this second criterion can be applied to collocations with more than two words (such as eliminating only those three-word collocations where two or more words are stop words).

A more general category of lexical features is the *ngram* feature. An ngram is a set of *one or more* word tokens, where the "n" stands for the number of word tokens in the ngram. So a single word token is a 1-gram or *unigram*, a two-word set is a 2-gram or *bigram*, a three-word set is a 3-gram or *trigram* and so on. The difference between multi-word ngrams and collocations is that ngrams do not require that the words they are composed of be contiguous. There can be zero or more words in between a pair of words in an ngram. In that sense, collocations are a special case of ngrams where there are no words permitted in between the constituent words. To restrict the number of ngram features and keep them semantically coherent, usually a limit is imposed on the number of words permitted in between two constituent words of an ngram. To clarify this with an example, the bigram features in the sentence "Frodo slips the *Ring* on and disappears" with the limit of at most two intermediate words permitted in their constituent words, and the restriction that no word should be a stop word would be:

{Frodo slips, Frodo Ring, slips Ring, Ring disappears}

Linguistic features of the context of an ambiguous word are also commonly used for WSD. Among linguistic features, Part-of-Speech (POS) tags of the word tokens in close vicinity of the ambiguous word (usually up to 2 words to the left and right of the ambiguous word, including the POS tag of the ambiguous word itself) are fairly common. POS tags are special unique identifiers for Parts of Speech of a language. For example, the POS tag for the part of speech *noun* in the POS tag set used by Hepple [22] is NN. The POS tagged version of our sentence:

> "Frodo slips the *Ring* on and disappears."

using the Hepple POS tagger [22] implemented in GATE (General Architecture for Text Engineering) [10][8] is:

> "Frodo/NNP slips/VBZ the/DT Ring/NNP on/IN and/CC disappears/VBZ ./."

where the part of speech tags for each word token appear after the "/" and:

NNP stands for *proper noun - singular*,
VBZ stands for *verb - 3rd person singular present*,
DT stands for *determiner*,
IN stands for *preposition or subordinating conjunction*,
CC stands for *coordinating conjunction* and
. stands for the *literal* period.

So the POS tag features for our example, in a window of size two around the ambiguous word *Ring* would be: $\{$VBZ, DT, NNP, IN, CC$\}$. Usually POS tags of stop words are not skipped, because they may contain linguistics clues which can be useful in disambiguation and secondly, removal of stop words can adversely affect the reliability of POS tagging.

Another type of linguistic feature used for WSD are syntactic features obtained by performing a shallow parse of the context of an ambiguous word. Parsing of a sentence involves identifying higher level linguistic phrases using POS tagged word tokens. Shallow parsing or chunking as proposed by Abney [1] identifies simple phrases such as a *Noun phrase*, *Prepositional phrase* or a *Verb phrase* in a sentence as opposed to a full parse yielding a detailed parse tree. Additionally

[8]http://gate.ac.uk

some shallow parsers also include *Subject-Object* relationship identification [11]. In WSD, features such as whether the ambiguous word belongs to the Noun phrase or the Verb phrase or whether it is the subject or object of the main verb can be used. The shallow parse for our example as given by the Memory-Based Shallow Parser[9] proposed in [11] is as follows:

For the phrasal analysis:

[NP Frodo/NNP NP] [VP slips/VBZ VP] [NP the/DT Ring/NNP NP] [Prep on/IN Prep] and/CC [VP disappears/VBZ VP] ./.

For the subject-object analysis:

[NP1Subject Frodo/NNP NP1Subject] [VP1 slips/VBZ VP1] [NP1Object the/DT Ring/NNP NP1Object] [P on/IN P] and/CC [VP disappears/VBZ VP] ./.

If we assume that we are interested in the following four ideas - whether the ambiguous word is a part of the noun phrase, whether it is a part of the verb phrase, whether it is part of the subject or whether it is a part of the object in the sentence. Each of these features is represented as a binary value, 0 for "no" and 1 for "yes." Then our syntactic features would be: $\{1, 0, 0, 1\}$.

Features using semantic knowledge can also be derived from an ontology such as WordNet [14] has been explored to obtain additional features for WSD. In one such approach, Mihalcea and Moldovan [37] use *unambiguous* words from the set of synonyms in WordNet for a given sense of the ambiguous word or information from the gloss (i.e., definition) of that sense from WordNet to create a search query and retrieve more data for that sense from the World Wide Web (WWW). Such an approach helps to create an augmented set of BoW features for the task of WSD. Patwardhan et al., [46] have made use of measures of semantic relatedness derived using WordNet to improve the accuracy on the task of WSD.

Next we discuss some of the popular classification algorithms in supervised machine learning and then introduce Support Vector Machines and kernel methods which are the focus of this thesis.

[9]http://ilk.uvt.nl/cgi-bin/tstchunk/demo.pl

2.6 Machine Learning Algorithms

In this thesis we use the following machine learning algorithms: Naïve Bayes Classifier, Decision Trees, Decision Lists, Support Vector Machines, Boosting Algorithms and Kernel Methods. The naïve Bayes classifier, decision trees and decision lists have all been shown to perform well on the task of WSD [6, 18, 41, 47, 56] and therefore establish a competitive baseline for comparing our methods. Support Vector Machines have been recently shown to also perform well on WSD and similar tasks [7, 30, 43]. The Boosting approach was included to compare our method with an ensemble methodology. Each of these methods is described in the following sub-sections.

2.6.1 Naïve Bayes Classifier

The naïve Bayes classifier is one of the simplest and most popular machine learning algorithms. It is based on the Bayes' rule for conditional probabilities, which states that:

$$P(Y = y_i|X = x_j) = \frac{P(X = x_j|Y = y_i)P(Y = y_i)}{\sum_k P(X = x_j|Y = y_k)P(Y = y_k)}$$

This essentially states that the conditional probability or the *posterior* probability $P(Y|X)$ can be found by taking the product of the conditional probability $P(X|Y)$ and the unconditional probability or the *prior* probability $P(Y)$, and dividing this product by the total probability that X has the given value over all possible values of Y. In the case of the naïve Bayes classifier, Y represents the output or the label for each instance of the dataset and X represents each instance in the dataset. Since X can have multiple attributes, we represent it as a vector $\langle X_1, X_2, X_3, ..., X_n\rangle$ with *n* features in general. Using this, the Bayes' rule equation for calculating the probability of any class value y_i for a given instance X becomes:

$$P(Y = y_i|X_1, X_2...X_n) \quad = \quad \frac{P(X_1, X_2...X_n|Y = y_i)P(Y = y_i)}{\sum_k P(X_1, X_2...X_n|Y = y_k)P(Y = y_k)}$$

The class value y_i assigned to an instance is the one which has the maximum probability according to the equation above. Since the denominator term remains the same for probability calculation of all class values, we can assign the class value using the following equation:

$$Y_{out} = \arg\max_{y_i \epsilon Y} P(X_1, X_2...X_n|Y = y_i)P(Y = y_i)$$

that is, the class value is the one which maximizes the numerator of the Bayes' rule equation shown earlier.

The "naïve" part of the naïve Bayes classifier is that it makes the simplifying assumption that all the features of an instance are conditionally independent given its label Y. Therefore using the rule of conditional independence of probabilities, the above equation reduces for a naïve Bayes classifier to:

$$Y_{nb} = \underset{y_i \epsilon Y}{\arg\max}\, P(Y = y_i) \prod_{j=1}^{n} P(X_j | Y = y_i)$$

Given a new unseen instance to classify, the naïve Bayes classifier calculates the probability of each class value given the features of the new instance and then assigns it the class value that has the maximal probability.

2.6.2 Decision Trees

Decision trees are also one of the most intuitive and popular machine learning algorithms. They are based on the idea of information gain from information theory. A decision tree is a top-down hierarchy of test conditions on the attributes of a dataset. Every node in a decision tree is a test of some attribute of the given instance, to categorize it into some subset depending on the value of the attribute for that instance. Every such non-leaf node in the decision tree (that tests an attribute) has as many branches or child nodes as the number of different values for the attribute being evaluated at that node. The tree is built starting from the root node, which tests the attribute that provides maximum information gain for the entire dataset, and the process continues recursively along each branch, until no further classification is required (usually within some tolerable level of error, so that the process can stop even if the dataset contains error).

Information gain is defined in terms of the entropy difference of a parent node in the decision tree and the weighted average of the entropies of its child nodes. Entropy of a node can be seen as a measure of "*impurity*" of a node in terms of the proportion of instances it contains of the different classes. The more balanced the proportion of different classes, the more the set is impure and hence the high entropy. So a set of instances with two class values will have maximum entropy if half of the instances are of one class and the other half are of the second class. So a set of instances with two class values will have minimum entropy when all the instances are of the same class (either the first or the second). Mathematically, entropy is the weighted average of negative logarithms (to the base 2) of the probabilities of class values in the set of data instances at the given node. Therefore the entropy of a node *N* is:

$$E(N) = \sum -p_i log_2 p_i$$

where p_i is the probability of class value *i*. For example, let us assume that for the node *N* being currently processed while building a decision tree there are *m* instances of the positive class (+) and *n* instances of the negative class (-). Then entropy at that node *N* is given by:

$$E(N) = -\left(\frac{m}{m+n}\right) log_2 \left(\frac{m}{m+n}\right) - \left(\frac{n}{m+n}\right) log_2 \left(\frac{n}{m+n}\right)$$

To define information gain, let us assume the following: node N has $|N|$ instances and entropy $E(N)$ is defined as above. Information gain is evaluated with respect to some attribute of the dataset. Let us assume that the attribute being considered currently is *A* and that it has v distinct values in the dataset. Therefore as discussed earlier, if the current node evaluates attribute *A*, then we will have v branches and therefore v child nodes of *N*. Let us name these child nodes $A_i, 1 \leq i \leq v$. Depending upon their value for the attribute *A*, the $|N|$ instances are divided among the child nodes A_i. Let us assume that the number of instances at child node A_i is $|A_i|$ and the entropy at the child node A_i is $E(A_i)$. Now, we define information gain using attribute *A* at node *N* as:

$$Gain(N, A) = E(N) - \sum_i \frac{|A_i|}{|N|} E(A_i)$$

A decision tree is constructed recursively by evaluating the information gain of each attribute for the set of data instances at the current node. For the root node, the information gain of all attributes over the entire dataset must be determined. Then the attribute with the maximum information gain is selected as the attribute to be used as the test for the current node. For all non-root nodes, the information gain of only those attributes that have not already been used in the parent branch of current node is evaluated. The leaf nodes do not evaluate any attribute, and in the best case contain instances of just one class and are therefore "pure." In the event the dataset contains an error or is not separable using the decision tree algorithm, the leaf nodes may not be pure.

Given a new unseen instance to classify, a decision tree begins by evaluating the instance for the attribute at the root node and “passes” the instance down the appropriate branch in the decision tree, until it reaches a leaf node. If the leaf node is pure, then the instance is assigned the same class as that of all the nodes in the leaf node. If the node is not pure, then one approach can be to assign the new instance the class that is most frequent among the instances at the leaf node.

2.6.3 Decision Lists

Decision list learning is a rule based approach, and is similar in concept to decision trees. The aim of the decision list learner is to discover a set of “if....then” or “switch....case” conditions that test attributes of the data instances and assign them a class value based on the first rule that *matches* or *covers* the data instance. If none of the discovered rules matches a given instance, then the most frequently occurring class in the training dataset is assigned as its class value. The rules are learned in an iterative and incremental fashion using the features of the training data. One rule is learned at a time and the set of examples covered by that rule are then eliminated from further analysis. Rules can be learned using different kinds of heuristic search procedures and ordered using evaluation criteria such as training sample accuracy (the rule that gives the best accuracy takes precedence) or information gain as in the case of decision trees. The final decision list is ordered, so that any new unseen instance to be classified is tested with each of the rules in the decision list in order from top to bottom, and the first rule that covers the instance decides the output class.

2.6.4 Support Vector Machines

Support Vector Machines (SVMs) are machine learning algorithms that have their roots in statistical learning theory [53] and can be applied to classification as well as regression problems. The SVM formulation for classification is designed to handle only two-class problems, but there are extensions to this basic formulation that handle the multi-class classification problems such as WSD. In its basic binary formulation, given an N-dimensional dataset (i.e., a dataset with N features for each instance) the aim of the SVM learner is to find an N-dimensional linear separating boundary between the two classes in the dataset. This linear separating boundary is a *hyperplane*. To improve the

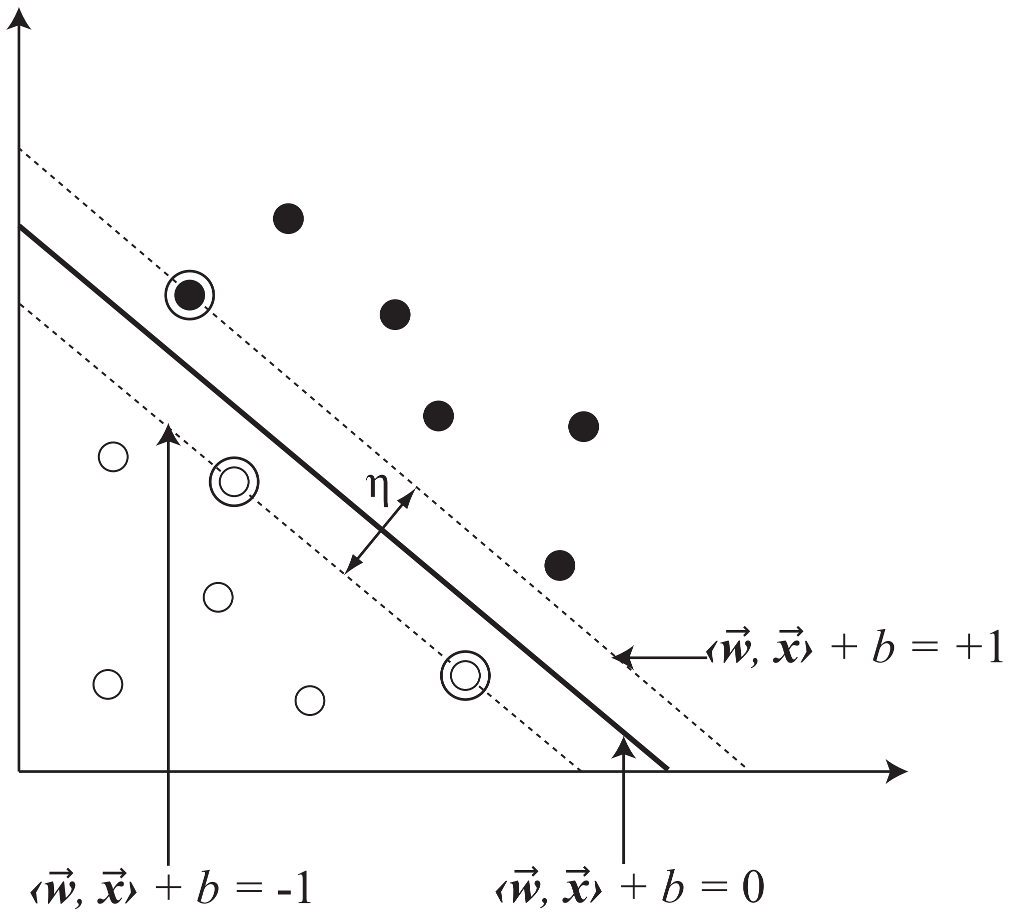
η
$\langle\vec{w}, \vec{x}\rangle + b = +1$
$\langle\vec{w}, \vec{x}\rangle + b = -1$
$\langle\vec{w}, \vec{x}\rangle + b = 0$

Figure 1: **Linear *hyperplane* of Support Vector Machine for a 2-dimensional dataset**

generalization ability of the SVM model (i.e., to classify future unseen examples with minimum error), the hyperplane should be selected such that it is located as far as possible from the data instances on its both sides. Let us consider a simple 2-dimensional dataset that is linearly separable to illustrate the terms related to SVMs and their formulation.

In Figure 1, we have a two dimensional dataset with two features represented along the two perpendicular axes. The solid circles are instances belonging to the positive class and the empty ones are those belonging to the negative class. The bold solid line that separates the two types of instances is the SVM *hyperplane*. Any hyperplane can be represented by the generic equation:

$$\langle\vec{\mathbf{w}}, \vec{\mathbf{x}}\rangle + b = 0$$

where $\vec{\mathbf{w}}$ is an *N*-dimensional weight vector, $\vec{\mathbf{x}}$ is an *N*-dimensional vector representing any point on the hyperplane and b is the distance of the hyperplane from the origin. $\langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle$ is the *dot product* or *inner product* of $\vec{\mathbf{w}}$ and $\vec{\mathbf{x}}$ in the *N*-dimensional space, that is:

$$\langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle = \sum_{i=1}^{N} w_i x_i$$

where w_i and x_i are the components of $\vec{\mathbf{w}}$ and $\vec{\mathbf{x}}$. In this basic formulation, learning an SVM model is learning the *N* components of the weight vector $\vec{\mathbf{w}}$ and the offset b. The two dotted lines parallel to the separating hyperplane in Figure 1 pass through the data instances nearest to the hyperplane on both sides. These are hyperplanes with equations $\langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle + b = +1$ (on the side of positive instances) and $\langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle + b = -1$ (on the side of the negative instances). The distance between these two hyperplanes that are parallel to the separating hyperplane is known as the *margin* of the SVM classifier and is represented by η. As mentioned earlier, to maximize the generalization capability of SVMs, the separating hyperplane should be at the farthest possible distance from both these hyperplanes passing through the data instances on both sides. As a result, the separating hyperplane is exactly in the center of these two hyperplanes and it should separate the data as much as possible. The goal of maximizing the generalization capability of SVM translates directly into maximizing the margin η. It can be shown that

$$\eta = \frac{2}{||\vec{\mathbf{w}}||}$$

where $||\vec{\mathbf{w}}||$ is the Euclidean length of the weight vector $\vec{\mathbf{w}}$ or the *2-Norm* of $\vec{\mathbf{w}}$, that is

$$||\vec{\mathbf{w}}|| = \sqrt{(w_1^2 + w_2^2 + ... + w_N^2)}$$

where w_i represents the *i*th component of the *N*-dimensional weight vector $\vec{\mathbf{w}}$. Note that we want the hyperplane to be defined such that for the instances of the positive class, $\langle \vec{\mathbf{w}}.\vec{\mathbf{x}} \rangle + b \geq +1$ should hold and for the instances of the negative class, $\langle \vec{\mathbf{w}}.\vec{\mathbf{x}} \rangle + b \leq -1$ should hold. Assuming there are *m* instances in the dataset, if we represent $y_i, 1 \leq i \leq m$ as the class of an instance with $y_i = +1$ for positive instances and $y_i = -1$ for negative instances, then we can combine the above two conditions on the hyperplane into the single set of constraints:

$$y_i(\langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle + b) \geq +1$$

We therefore have to maximize the margin subject to the constraints above. Although this actually means maximizing $\frac{2}{||\vec{\mathbf{w}}||}$ subject to the constraints $y_i(\langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle + b) \geq +1$, for practical purposes it is convenient to convert the problem to an equivalent form:

$$minimize \quad \frac{1}{2}||\vec{\mathbf{w}}||^2$$

$$such\ that \ \ y_i(\langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle + b) \geq +1, 1 \leq i \leq N$$

Note that $||\vec{\mathbf{w}}||^2$ is the same as $\langle \vec{\mathbf{w}}, \vec{\mathbf{w}} \rangle$, from the definition of $||\vec{\mathbf{w}}||$ above. This is a classical quadratic minimization problem where a quadratic expression is to be minimized subject to a set of linear constraints. This is also known as the *primal* form of the SVM formulation. The solution to the quadratic minimization yields the values of $\vec{\mathbf{w}}$ and b. Any new unseen instance x_{new} is then evaluated as follows: the function $\langle \vec{\mathbf{w}}, \vec{\mathbf{x}}_{new} \rangle + b$ is evaluated and y_{new} is assigned +1 if the function evaluates to a positive value, else y_{new} is assigned the value -1.

Using the Lagrangian theory for constrained optimization problems, the above quadratic minimization problem can be converted to an equivalent problem that has simpler constraints. This form is known as the *dual* form of SVM formulation and is as follows:

$$maximize \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j \rangle$$

$$such\ that\ \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i y_i = 0$$

Now the problem has been converted to learning just the α_i values ($1 \leq i \leq m$, one for each instance in the dataset) instead of learning the *N* components of the weight vector $\vec{\mathbf{w}}$ and b. These α's are known as the *Lagrange multipliers* and intuitively, they determine the importance of an instance in deciding the separating hyperplane. The instances in the training dataset for which α_i is non-zero are known as *support vectors* as these are precisely the instances that “support” the solution, so that even if the the other instances in the dataset are removed and the model is learned using only these *support vectors*, the same separating hyperplane will be learned. So support vectors determine the equation of the separating hyperplane. Referring back to Figure 1, the support vectors are the circled instances on the dotted hyperplanes parallel to the separating hyperplane.

It can be shown that the weight vector $\vec{\mathbf{w}}$ can be constructed using a linear combination of the training instances, with their corresponding Lagrange multipliers as coefficients, that is:

$$\vec{\mathbf{w}} = \sum_{i=1}^{m} \alpha_i y_i \vec{\mathbf{x}}_i$$

or equivalently

$$\vec{\mathbf{w}} = \sum_{i\epsilon\{Support\ Vectors\}} \alpha_i y_i \vec{\mathbf{x}}_i$$

as the Lagrange multipliers for the instances that are not support vectors are zeroes.

For classifying a new unseen instance x_{new}, we need to evaluate the function $\langle \vec{\mathbf{w}}, \vec{\mathbf{x}}_{new} \rangle + b$, but we can do so without explicitly evaluating $\vec{\mathbf{w}}$ by replacing the above value of $\vec{\mathbf{w}}$ in this function. So we need to evaluate:

$$\sum_{i\epsilon\{Support\ Vectors\}} \alpha_i y_i \langle \vec{\mathbf{x}}_i, \vec{\mathbf{x}}_{new} \rangle + b$$

We still need b to evaluate new instances. It can be found using any one of the support vectors x_{sv}, as we know that for support vectors from the positive class:

$$\sum_{i\epsilon\{Support\ Vectors\}} \alpha_i y_i \langle \vec{\mathbf{x}}_i, \vec{\mathbf{x}}_{sv} \rangle + b = +1$$

and for support vectors from the negative class:

$$\sum_{i\epsilon\{Support\ Vectors\}} \alpha_i y_i \langle \vec{\mathbf{x}}_i, \vec{\mathbf{x}}_{sv} \rangle + b = -1$$

So b can be easily calculated once we have learned the α's.

So far we have only analyzed a linearly separable dataset with no errors or outliers. Both the primal and the dual formulations of SVM shown above can be extended to permit some amount of error on the training data by adding error terms (known as *slack variables*) in the formulations. For the dual formulation, it can be shown that this results in only one additional constraint on the Lagrange multipliers α in terms of a a new parameter *C* which serves as an upper bound on their value. This parameter *C* is known as the *trade-off* parameter between the error and the SVM margin as selecting a large value for *C* forces the SVM margin to be smaller and vice-versa. Cristianini and Shawe-Taylor [8] gives all of the formulations mentioned above along with their proofs.

2.6.5 Boosting Algorithms

Boosting algorithms in machine learning are based on the idea of combining several simple classifiers (such as a single node decision tree, also known as a *decision stump*) to form one classifier ensemble that yields significantly better accuracy than its component classifiers yield individually. The argument in favor of boosting algorithms is that it is easier to build several simple classifiers that correctly classify small subsets of the training data and combine them together rather than trying to build one complex classifier that correctly classifies all the training instances. The AdaBoostM1 algorithm ([16, 17]) is an example of a boosting algorithm. Assuming that there is a simple, weak learning algorithm available, the AdaBoostM1 algorithm repeatedly invokes this weak learner (the number of iterations are fixed using a parameter T) to classify a random sample of instances drawn from the training data using a probability distribution D. Initially D is a *uniform* distribution so that all the instances have equal probability of being selected. As the algorithm proceeds through iterations, D is updated to reduce the probabilities of instances that are correctly classified by the weak learner in the current iteration and correspondingly increase the probabilities associated with the mis-classified examples. This forces the weak learner in subsequent iterations to focus on the instances that were mis-classified in previous iterations. Once the T iterations are complete, the T models of the weak learner are combined to create an ensemble that predicts the classification of a new unseen instance based on a weighted vote of the T models. Intuitively, lower weight is given to the classifier models from earlier iterations and higher weight is given to the classifier models from latter iterations as they focus on getting the hard-to-classify instances correct. The label that is predicted with the highest weighted vote is assigned to the new instance.

2.6.6 Kernel Methods

The SVM formulations presented above can find separating hyperplanes for linearly separable data. However, if the data is not separable linearly, that is, if the data is such that it requires a non-linear separating boundary for accurate classification, then the formulations above need to be adapted to such situations.

One of the possible alternatives is to transform the input data using some transformation function $\Phi(.)$ such that in the transformed space, the data instances are linearly separable. Consider the hypothetical example shown in Figure 2 where a 2-dimensional dataset that is not linearly separable is transformed using some function $\Phi(.)$ into a 2-dimensional space where it becomes linearly separable. The original space where the data is not linearly separable is referred to as the *input space* and

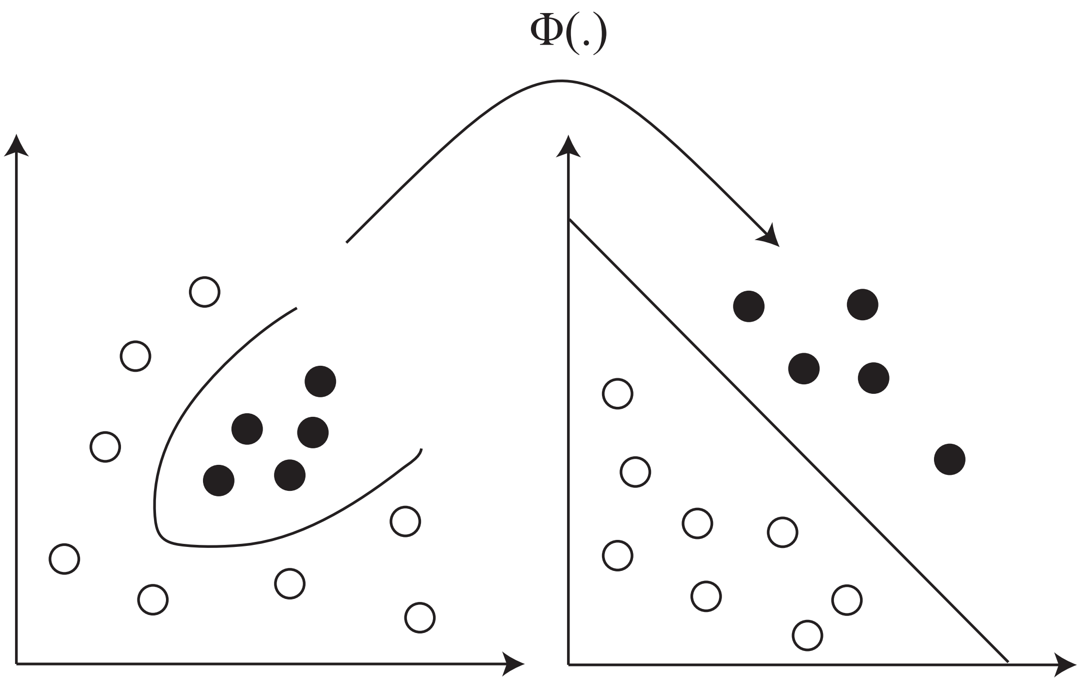

Figure 2: **A 2-dimensional linearly non-separable dataset transformed using $\Phi(.)$ into a linearly separable dataset**

the transformed space where the data becomes linearly separable is referred as the *feature space*. Normally the feature space has higher number of dimensions compared to the input space, and in some cases its dimensions are infinite.

The problem in this approach is that it is computationally prohibitive to transform a dataset into a very high-dimensional (possibly infinite) feature space and then perform quadratic optimization calculations in such a high-dimensional space. This is exactly where the *kernel functions* are useful.

Notice that in the dual formulation of the SVM, the actual data instances appear only in the form of their inner products, $\langle \vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j \rangle$. Thus if we are able to calculate the inner products of the transformed data instances in the high-dimensional feature space *directly* using some function, instead of actually transforming the data points and then calculating their inner product then the computational complexity will be reduced while still learning a linear separation in the high-dimensional feature space. A kernel function is exactly such a function and it can be defined such that the following holds, for some transformation function $\Phi(.)$, which may not be necessarily known:

$$K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = \langle \Phi(\vec{\mathbf{x}}_i), \Phi(\vec{\mathbf{x}}_j) \rangle$$

Consider for example a simple transformation function $\Phi(.)$ that operates on a 2-dimensional vector $\vec{\mathbf{x}} = (x_1, x_2)$ and converts it into a 3-dimensional vector such that $\Phi(\vec{\mathbf{x}}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$. Now given two vectors $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ in input space, their inner product in this 3-dimensional feature space will be equal to:

$$\begin{aligned}
\langle \Phi(\vec{\mathbf{x}}), \Phi(\vec{\mathbf{y}}) \rangle &= (x_1^2, x_2^2, \sqrt{2}x_1x_2)(y_1^2, y_2^2, \sqrt{2}y_1y_2) \\
&= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\
&= (x_1y_1 + x_2y_2)^2 \\
&= \langle \vec{\mathbf{x}}, \vec{\mathbf{y}} \rangle^2
\end{aligned}$$

This shows that defining a kernel function $K(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ such that:

$$K(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = \langle \vec{\mathbf{x}}, \vec{\mathbf{y}} \rangle^2$$

achieves the effect of finding an inner product in the 3-dimensional feature space defined by:

$$\Phi(\vec{\mathbf{x}}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

It simply computes the inner product of the data instances in input space and squares that result. This is the so-called *kernel trick*, where a kernel function directly gives the inner product of input data instances in a very high-dimensional transformed feature space by performing the transformation and inner product calculation in high-dimensional feature space *implicitly* at a much lower computational cost.

In the example mentioned above, the kernel function was arrived upon with the knowledge of the transformation function $\Phi(.)$. It is not necessary that this be the case. One can define a kernel function without having to know the actual transformation that leads to the feature space in which the kernel function evaluates the inner product. In such a case, to verify that the kernel function is "valid" (i.e., it leads to a sound definition of inner product in some transformed space) the function needs to have certain properties.

Before we mention the properties that a function should have in order to be a valid kernel function, it will be useful to define a *kernel matrix*. First, let us embed the kernel function $K(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ in the dual formulation of the SVM to transform it into:

$$maximize \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j)$$

$$such\ that\ \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i y_i = 0$$

Here we have just replaced the inner product $\langle \vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j \rangle$ in the original dual formulation with the kernel function $K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j)$. Now we can see that the kernel function will be evaluated for every pair of training instances when solving the quadratic optimization problem to learn an SVM model. The *kernel matrix* (also known as the *Gram matrix*) is a square matrix of size *m-by-m* (m is the number of training instances), where the entry in the ith row and jth column is $K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j)$. Thus a kernel matrix stores the kernel evaluations for all pairs of training examples. A function is a valid kernel function if the kernel matrix generated using that function is a *symmetric positive semi-definite* matrix (i.e., a symmetric matrix whose every eigenvalue is non-negative).

While the computational advantage is an appealing aspect of the kernel methods, another equally interesting aspect is the fact that any quadratic optimization problem using a kernel function can use any well-defined kernel matrix for learning a decision surface. From this perspective, we need not look at the entries in the kernel matrix necessarily as inner products of some vectors. A more general perspective can be that the values of the kernel matrix are similarity values between pairs of objects and these objects need not be representable in the form of vectors, but can be of any type as long as a there is some well-defined measure of similarity between them. For example, one can have kernel matrices generated out of similarity values between parse trees, where the similarity score could be the number of overlapping identical branches in the two parse trees. This idea is what makes kernel methods extremely powerful for introducing prior knowledge into SVM learners, to aid the task of classification.

2.7 Semantic Kernels

Our approach is based on the idea of latent semantic kernels [9] and domain kernels [19] for WSD. The classical Vector Space Model (VSM) from information retrieval literature is the basic building block for both these approaches. First, we introduce some concepts and terminology related to the VSM that we shall utilize later. Initially, we use the conventional information retrieval terminology and then translate that into the WSD terminology.

Given a set D containing m documents, the Vector Space Model [50] involves constructing a term-by-document matrix where all the unique set of terms (which can be single words, collocations or word phrases, or any other features that can be identified from the corpus of documents) in the entire set D of documents are represented as rows of the matrix and the documents themselves are represented as the columns of this matrix. If there are n unique terms in the set of terms T in all the documents in D then the representation is an $n \times m$ matrix. The cell value in the ith row and jth column is the number of times the term t_i occurs in the document d_j. The reason this model is known as the "Vector Space Model" is that in the term-by-document matrix, each of the m documents is represented as an n dimensional column vector, with each of its components representing the term frequency (TF) of the respective term in the given document. In one variation of this model, the matrix values are not term frequencies in the corresponding documents, but a value known as the *TF-IDF* score – which stands for the product of the *term frequency* and the *inverse document frequency* (IDF). *Term frequency* is the same as before – the number of times a term occurs in a given document. *Inverse document frequency* is a measure of how informative a given term is to help identify a subset of documents from D - the smaller the subset of documents identified, the more informative is the term. Mathematically IDF of a term t_i that occurs in m_i documents out of the total m documents is defined as:

$$IDF(t_i) = log_2(\frac{m}{m_i})$$

One can see that as m_i decreases (i.e., the term occurs in fewer number of documents), its IDF value increases and as m_i approaches m, the IDF value approaches 0, that is, a term that occurs in all of the documents in D is not considered informative.

Given this VSM, there can be several methods to analyze the similarity of two documents using their corresponding document vectors. One of the commonly employed mechanism is finding the cosine value of the angle between the two document vectors. The cosine value of the angle between two vectors is in the interval $[-1, +1]$, -1 being the value when the vectors are pointing in opposite directions (i.e., they are 180 degrees apart and therefore most unlike or dissimilar from each other) and +1 being the value when the vectors are pointing in the same direction (i.e. they are 0 degrees apart and therefore most alike or similar to each other). The cosine value between two document vectors $\vec{d_1}$ and $\vec{d_2}$, each n-dimensional, is defined as:

$$cos(\vec{d_1}, \vec{d_2}) = \frac{\langle \vec{d_1}, \vec{d_2} \rangle}{||\vec{d_1}||.||\vec{d_2}||}$$

where the numerator is the *inner product* of the document vectors and is defined as previously in the introduction to SVMs and the denominator is the product of the Euclidean lengths of the two document vectors, again as defined earlier in the SVM introduction. Note that the formulation for the cosine of two vectors can be re-arranged to view it simply as the inner product or dot product of their normalized unit vectors:

$$cos(\vec{d_1}, \vec{d_2}) = \left\langle \frac{\vec{d_1}}{||\vec{d_1}||}, \frac{\vec{d_2}}{||\vec{d_2}||} \right\rangle$$

So if the vectors are already normalized, then the cosine evaluates to the same value as the dot product of the two vectors. So that,

$$cos(\vec{d_1}, \vec{d_2}) = \langle \vec{d_1}, \vec{d_2} \rangle \; if ||\vec{d_1}|| = 1 \; and \; ||\vec{d_2}|| = 1$$

An essential property of a measure of similarity is that it should be symmetric, so that *similarity(a, b)* is the same as *similarity(b, a)*. This is true for the cosine measure since $cos(\vec{d_1}, \vec{d_2})$ is equal to $cos(\vec{d_1}, \vec{d_2})$ which is the cosine of the smaller angle between the two vectors. Another commonly used measure of similarity is simply the inner product or the dot product of the document vectors. This is exactly the numerator of the cosine measure above – it is not normalized using the Euclidean vector lengths in the denominator.

Recalling the concept of kernels as a measure of similarity between instances in a dataset, one can see that an $m \times m$ matrix K created out of similarity values between every pair of documents in D (such that $K_{i,j}$ represents the similarity between document vectors $\vec{d_i}$ and $\vec{d_j}$ and $K_{j,i} = K_{i,j}$) is a kernel matrix for the set of documents D and can be used to apply the SVM algorithm on these set of documents. This kernel K and the polynomial and Gaussian extensions to it (which are simply mathematical transformations on the basic kernel K) were proposed by Joachims [24] for text categorization of documents into a predefined set of topics. If the similarity measure used is the inner product, then this kernel matrix K is the same as the $m \times m$ resultant matrix after multiplying the transpose of D (D^T) and D, that is:

$$K_{m\times m} = D^T_{m\times n} D_{n\times m}$$

This is the basic VSM kernel.

A major disadvantage of the VSM is that any similarity measure based purely on the document vectors in the VSM takes into account just the lexical overlap of the terms across documents. Two documents appear to be more similar if more terms overlap. This does not take into account the semantic relationships among terms, such as the fact that two words are synonyms of each other (and hence should ideally increase the similarity of the documents in which they occur) or that an ambiguous word is used in two documents in different senses (and thus should not contribute to increasing the similarity of the documents). To incorporate semantic relationships in evaluating document similarities, Cristianini et al., [9] propose using the concept of Latent Semantic Indexing (LSI) [12] from the information retrieval theory to build Latent Semantic Kernels (LSK). LSI involves Principal Component Analysis (PCA) of the term-by-document VSM matrix by means of a process known as Singular Value Decomposition (SVD). The goal of PCA in a task such as document categorization is to extract the important concepts from the entire corpus of documents and modify the document vectors in such a way that they represent the similarity or difference of the documents with respect to the important concepts. This can be achieved by dimensionality reduction after decomposing the term-by-document matrix using SVD. SVD decomposes a matrix D as follows:

$$D = U\Sigma V^T$$

where U and V are orthogonal matrices (i.e., $UU^T = VV^T = I$, the identity matrix) and Σ is a diagonal matrix (not necessarily square) such that the diagonal elements of Σ are *singular values* of D and the columns of U are *left singular vectors* of D corresponding to the singular values in Σ. They are in fact the *eigenvectors* of the term-term similarity matrix DD^T. Similarly, columns of V are the *right singular vectors* of the matrix D and eigenvectors of the document-document similarity matrix D^TD. For an matrix D with n rows (representing n terms) and m columns (representing m documents), that is, for a matrix of size $n \times m$, the *full* SVD is defined as:

$$D_{n\times m} = U_{n\times n}\Sigma_{n\times m}V^T_{m\times m} \tag{1}$$

A variation known as the *thin* or *economy sized* SVD is defined as:

$$D_{n\times m} = U_{n\times m}\Sigma_{m\times m}V^T_{m\times m} \tag{2}$$

A *compact* SVD is defined as:

$$D_{n\times m} = U_{n\times r}\Sigma_{r\times r}V^T_{m\times r} \tag{3}$$

Here r represents the *rank* of the matrix D and $r \leq min(n, m)$. Note again that Σ can be a rectangular matrix (as in the full SVD when $n \neq m$), the only condition is that all its non-diagonal elements ($\Sigma_{i,j}, i \neq j$) are zeros. Finally, the most commonly used SVD formulation is the *truncated* SVD:

$$D_{n\times m} = U_{n\times k}\Sigma_{k\times k}V^T_{m\times k} \tag{4}$$

where only the first k columns of U and V along with the k singular values from Σ from the full SVD are selected such that $k \ll m$. These and further details about SVD can be found in Bai et al.,(editors) [3][10]. In all of the above formulations of SVD, the dimensions of the matrix reconstructed after multiplying U, Σ and V^T are still $n \times m$. The process however is known as *dimensionality reduction* because of the fact that fewer number of singular values are used to reconstruct a modified matrix D. Intuitively one can think of selecting k singular values as selecting the first k predominant concepts in the set of documents and then re-evaluating the vector components of all documents. This process results in some loss of information due to the compaction to k dimensions, but also helps identify the k principle semantic concepts. It indirectly takes into account co-occurrence information of terms such that if two terms t_i and t_j often occur together in many documents, then they also induce similarity between documents d_x and d_y even if only t_i appears in d_x and only t_j appears in d_y. An illustrative example of this can be found in Deerwester et al., [12] and Landauer et al., [29]. This adds a *semantic* perspective to the similarity analysis, which is lacking in the original VSM.

Now we consider the thin SVD from Equation 2 and explain the formulation of the Latent Semantic Kernel in [9]. Let I_k denote a size $m \times m$ identity matrix with elements beyond $I_{k,k}$ equal to zero. Now let $U_k = U_{n\times m}I_k$ be the matrix of left singular vectors with only the first k columns of $U_{n\times m}$ non-zero. Now a kernel similar to the basic VSM kernel, but generated using U_k is:

$$K_{semantic} = (U_k^T D)^T U_k^T D$$

[10]Also available online at the URL `http://www.cs.utk.edu/~dongarra/etemplates/book.html`

This is the basic form of *Latent Semantic Kernel* (LSK) proposed in Cristianini et. al., [9] and it captures the similarity of documents in a space with reduced dimensions as only k left singular vectors are used in evaluating the similarity. One can verify that within a very small error, the above kernel matrix is the same as the one generated by reconstructing D_k by multiplying the component matrices obtained after *truncated* SVD of D and then computing $K = D_k^T D_k$.

Consider again the thin SVD formulation from Equation 2. The concept of *domain kernels* [19] is based on a *domain matrix* created from the SVD of a term-by-document matrix as follows:

$$D_{LSA} = I^N_{n \times n} U_{n \times k} \sqrt{\Sigma_{k \times k}}$$

where $I^N_{n \times n}$ is defined as a normalizing diagonal matrix with $I^N_{i,i}$ being the Euclidean length of the ith row of the matrix $U_{n \times k}\sqrt{\Sigma_{k \times k}}$ and $\sqrt{\Sigma_{k \times k}}$ simply denotes a matrix consisting of the square roots of individual cell values of $\Sigma_{k \times k}$. Each row in this domain matrix represents a term in the reduced k-dimensional feature space instead of the original m-dimensional feature space and is referred to as a *domain vector* for the corresponding term. The idea is that the m documents have collapsed into the k most meaningful *domains* at a conceptual level and the relevance of term i in domain j is represented by the value $D_{LSA i,j}$ of the domain matrix. Given this domain model, the kernel function for similarity between any two terms is simply the dot product of their domain vectors:

$$K(t_i, t_j) = \langle \vec{t_i}, \vec{t_j} \rangle$$

where $\vec{t_i}$ and $\vec{t_j}$ represent the ith and jth rows of the domain matrix D_{LSA} (i.e., the domain vectors for t_i and t_j) and correspond to the representation of the terms t_i and t_j respectively in the domain model. The kernel function for evaluating the similarity of two documents initially requires creation of domain vectors for the documents. A domain vector for a document is created by transforming the document vector in the original VSM into a lower dimensional feature space as follows:

$$\vec{d_i'} = \vec{d_i} I^{IDF} D_{LSA}$$

where I^{IDF} represents the square diagonal matrix containing inverse document frequencies of the n terms and D_{LSA} is the domain matrix described above. Now the kernel function for any two documents is the dot product of their domain vectors:

$$K(d_i, d_j) = \langle \vec{d_i'}, \vec{d_j'} \rangle$$

At this point, we will translate the VSM terminology introduced so far into corresponding concepts related to WSD, which is fairly straightforward and intuitive. The concept of *documents* translates to instances of ambiguous words such as sentences or paragraphs, more commonly known as *contexts* in WSD. The *terms* in the VSM translate into *features* for the task of WSD. A term-by-document matrix becomes a *feature-by-context* matrix. The notion of similarity between contexts is identical to that of similarity between documents, with the intuitive understanding that more the similarity of two contexts of an ambiguous word, more is the chance that it has the same sense in those two contexts. This is a due to the distributional hypothesis proposed by Harris [21] and the contextual similarity hypothesis by Miller and Charles [38].

2.8 Unsupervised Learning

The unsupervised learning part of our approach is captured by knowledge about similarity of two contexts, which we evaluate using kernel matrices generated from unlabeled instances of an ambiguous word. Generation of one class of kernel matrices in our approach is done using the transpose of the first-order context representation as presented by Purandare and Pedersen in [49] and that of the second class of kernels is done using the second-order context representation used by Purandare and Pedersen [49]. Both these representations and the methods using them work in a completely unsupervised fashion, evaluating the similarity of the contexts under consideration. They are described in more detail in Purandare [48].

The first-order context representation in [49] is a context-by-feature representation of the corpus containing the ambiguous word. A list of features such as unigrams or bigrams or co-occurrences are extracted from a corpus containing multiple instances of an ambiguous word. Every context is then represented along the rows of a matrix and every identified feature is represented along the columns of the matrix. The cell values are frequency counts of the number of times a feature appears in a given context, or binary values indicating presence or absence of a feature in a given context. The rows of this matrix therefore represent context vectors with the features as their dimensions.

The second-order context representation creates a word-by-word matrix out of bigram or co-occurrence features identified from the set of contexts of ambiguous words. For bigram features, the first word in the bigram forms a row of the matrix and the second word forms a column of the matrix, and the corresponding cell value stores a statistical score of association for that bigram. For co-occurrence features, all the unique words are extracted and are listed along the rows as well as columns of a square matrix, and the cell values again contain a score of association for the co-occurrence. The rows of a word-by-word matrix (created using either bigrams or co-occurrences) represent word vectors with the words that they co-occur with as dimensions. Any context is then represented as a vector by averaging the word vectors for words that are present in that context.

Both the first and second order context representations and clustering of data using these representations are implemented in the freely available SenseClusters[11] package.

We elaborate on the exact process of kernel matrix creation using context representations of unlabeled data in the next chapter.

[11]http://senseclusters.sourceforge.net/

3 Semi-Supervised Semantic Kernels for Word Sense Disambiguation

In this chapter we present an approach based on semi-supervised learning, that makes use of unlabeled data and we elaborate on our SVM kernel formulations.

3.1 Motivation

While supervised machine learning methods represent the state-of-the-art in WSD, they face the problem of the so-called "Knowledge Acquisition Bottleneck." As mentioned earlier, supervised methods require a set of labeled instances of an ambiguous word in order to learn a model to disambiguate new instances. Ng [42] estimates that approximately 3.2 million labeled instances of ambiguous words (3200 ambiguous words with 1,000 labeled instances of each) would be required to build a robust and general-purpose WSD system that can significantly improve the accuracy over the baseline measure of assigning the most frequent sense. He further mentions that this is roughly equivalent to 16 man-years worth of effort in annotating the data with the correct sense for all the 3.2 million instances. It also empirically establishes for WSD the fact that the availability of more training data for a learning algorithm leads to better accuracy on novel instances. This clearly shows the "data-hungry" nature of the supervised machine learning algorithms. In order to be able to build classifier models that are highly accurate as well as generalize well (i.e., have low error on unseen data instances), the training data size should be sufficiently large to allow the algorithms to acquire knowledge of the majority of the disambiguating aspects of a large number of ambiguous words and their senses. But the requirement to annotate such a large number of instances imposes a practical limitation on the amount of labeled training data that can be made available. This is known as the Knowledge Acquisition Bottleneck.

With the advent of the World Wide Web and the exponential increase in digital storage of content such as newspaper articles, scholarly publications and medical records of patients, there is comparatively more data in a raw form than labeled data. Such a free-flowing corpus of text is often referred to as "unlabeled data." Recent efforts in NLP literature [19, 20] have focused on making use of unlabeled data to improvise upon the classical supervised machine learning tasks such as word sense disambiguation and text categorization. This approach is generally referred to as "semi-supervised learning." While having fully unsupervised methods to acquire knowledge from unlabeled data is very appealing, in practice the lower accuracy of unsupervised methods poses a significant limitation. The aim of semi-supervised methods is to combine the good properties of unsupervised and supervised learning approaches - the broad coverage provided by unsupervised methods using plentiful unlabeled data and the accuracy provided by supervised methods using a small amount of labeled data. However, to our knowledge relatively few studies have been done in the NLP literature related to the medical domain that make use of semi-supervised approaches [44, 45] and more specifically there has not been any research in applying task-specific SVM kernels to the problem of WSD and abbreviation disambiguation in medical domain. This thesis focuses on applying existing unsupervised methods by Purandare and Pedersen [49] and extensions to these methods to generate task-specific SVM kernels for the problem of WSD and abbreviation disambiguation in the medical domain.

3.2 Semantic Kernels

Our approach combines unsupervised knowledge from unlabeled corpora of text and supervised knowledge from labeled training instances using kernel methods for SVMs. We assume the availability of a large set of unlabeled instances of an ambiguous word for unsupervised learning to create feature vectors that are used in evaluating kernel matrices for the instances from labeled data. The process in general proceeds as follows: First, features are identified from the unlabeled data using unsupervised methods. Second, each feature is represented using a feature vector which encodes the properties of the corresponding feature, which are derived from the unlabeled data. The set of feature vectors for all identified features represents the knowledge gained from unlabeled corpora. Third, this knowledge is used to create a vector representation for each of the contexts available in the labeled data. Fourth, the similarity between every pair of context vectors is calculated. A square matrix created out of these similarities of every pair of labeled contexts is the kernel matrix that we utilize for running SVMs on the labeled instances. One thing to note here is that we *do not* utilize the labeled instances as a part of unsupervised learning. The features from labeled data are identified separately and are used in the supervised learning process of SVMs in addition to the kernel matrix. While evaluating the actual kernel function, the similarity values from the unsupervised kernel matrix are added to the similarity values computed using the default linear kernel on features from labeled instances - thus providing additional input to analyze similarity of the contexts. All of our kernels are generated using the unsupervised methods provided in the SenseClusters[12] package and extensions of these methods that we have created.

The features from labeled data are available to all machine learning algorithms, however SVMs can make use of the unsupervised knowledge in the form of kernels in a very elegant way. So the kernel matrices learned from the unlabeled data form the key elements of our methodology and represent the semi-supervised learning aspect of our methods. Based on the idea of latent semantic kernels and on the concept of domain kernels, we define two classes of kernels that are extensions of these existing kernels with respect to the type of features used and more crucially, with respect to the type of feature representation used for learning from the unlabeled data.

3.2.1 Latent Semantic Analysis Kernels

The first class of kernels that we introduce is the Latent Semantic Analysis (*LSA*) kernel, which is a generalization of the domain kernel by Gliozzo et al., [19], involving the use of features other than unigrams or Bag-of-Words to learn from unlabeled data. Initially we create a first-order context representation of the unlabeled contexts as described in Purandare and Pedersen [49]. This yields a context-by-feature matrix, where the features along the columns of the matrix can be unigrams, bigrams or two-word co-occurrences, each selected using different feature selection criteria. We then take the transpose of this context-by-feature matrix, which yields a feature-by-context matrix. Note that this representation is similar to the LSA representation proposed in Deerwester et al., [12] and therefore we call this class of kernels LSA kernels. The rows of this transposed first-order context representation are the feature vectors used for representing any two contexts whose similarity is to be evaluated using unsupervised methods. We explain each kernel in this category, based on the features used in our LSA representation below.

Unigram LSA Kernels Our unigram kernels are similar to the domain kernels introduced by Gliozzo et al., [19]. We start with an unlabeled corpus containing m instances of the ambiguous

[12]http://senseclusters.sourceforge.net/

word. Unigram features are then identified for these m instances using the Ngram Statistics Package [5] (NSP). Feature selection is done by eliminating a predefined set of stop words or functional words and also by removing words that occur fewer times in the m contexts than some frequency cut-off value (which is a parameter and takes the values of 2, 5, 10 and 15 for our experiments). Assuming that n unigrams are identified as features, a *unigram-by-context* matrix is created as the transpose of the first-order context-by-feature matrix, so that for the m instances of the ambiguous word and the n unigrams identified across all the instances, a matrix D of size $n \times m$ is generated.

Consider for example, the following set of five unlabeled contexts (c_1 through c_5) for the word *immunosuppression*:

$\mathbf{c_1}$: The patient received a second graft and despite an optimal six-antigen-match and different *immunosuppression* with tacrolimus, thrombosis recurred by the fifth postoperative day.

$\mathbf{c_2}$: This study included five groups: allografts without immunosuppression (group A, n = 12), allografts with immunosuppression (group B, n = 13), autografts without *immunosuppression* (group C, n = 11), autografts with immunosuppression (group D, n = 12), and autografts treated by 45 minutes of pretransplant warm ischemia to induce acute graft pancreatitis (group E, n = 14).

$\mathbf{c_3}$: STUDY DESIGN: Ninety-eight consecutive patients (59 children, 39 adults) with a panoply of indications received 104 allografts under tacrolimus-based *immunosuppression*: intestine only (n = 37); liver and intestine (n = 50); or multivisceral (n = 17).

$\mathbf{c_4}$: If this *immunosuppression* is critical for the development of most skin tumors, then its prevention should result in prevention of photocarcinogenesis.

$\mathbf{c_5}$: However, excessive exposure to UV can overwhelm the cutaneous antioxidant capacity, leading to oxidative damage and ultimately to skin cancer, *immunosuppression* and premature skin aging.

Assume that the following unigram features are identified from these five sentences: *graft, antigen, tacrolimus, thrombosis, allografts, autografts, liver, intestine, skin, tumors, photocarcinogenesis, cutaneous, cancer*. In this example, the number of features $n = 13$ and the number of unlabeled contexts $m = 5$. The $n \times m$ feature-by-context matrix D for this example is as shown in

Table 1: **A unigram-by-context matrix used for unigram LSA kernels.**

	$\mathbf{c_1}$	$\mathbf{c_2}$	$\mathbf{c_3}$	$\mathbf{c_4}$	$\mathbf{c_5}$
graft	1	1	0	0	0
antigen	1	0	0	0	0
tacrolimus	1	0	1	0	0
thrombosis	1	0	0	0	0
allografts	0	2	1	0	0
autografts	0	3	0	0	0
intestine	0	0	2	0	0
liver	0	0	1	0	0
skin	0	0	0	1	2
tumors	0	0	0	1	0
photocarcinogenesis	0	0	0	1	0
cutaneous	0	0	0	0	1
cancer	0	0	0	0	1

Table 1. The cell values in this matrix are frequency counts of the number of times a feature appears in a context.

Truncated SVD of D yields $D = U_{n \times k} \Sigma_{k \times k} V^T_{m \times k}$. We then define the domain matrix as $D_{LSA} = U_{n \times k} \sqrt{\Sigma_{k \times k}}$. Once the domain matrix is defined, then any arbitrary context can be represented in the k-dimensional domain space as the average of its feature vectors that form the rows of the domain matrix. Mathematically:

$$\vec{c_i} = \frac{\sum_{k \epsilon features\ of\ c_i} \vec{w}_{ik}}{number\ of\ features\ of\ c_i}$$

where w_{ik} is a feature identified for the context c_i and $\vec{w}_{ik}$ represents its corresponding feature vector as a row in the domain matrix. For example consider the following new context c_6 of the word *immunosuppression*:

c_6: Predictors and outcome of early- versus late-onset major bacterial infections in liver transplant recipients receiving tacrolimus (FK506) as primary *immunosuppression*.

Since our unigram-by-context matrix from Table 1 is quite small originally, we do not show the truncated SVD and the creation of the domain matrix D_{LSA} from it. Instead let us assume that for this example $D_{LSA} = D$. Now, the two words *liver* and *tacrolimus* in the new context c_6 are present in our domain matrix. Therefore, a context vector for c_6 is created as follows:

$$\begin{aligned}
\vec{c_6} &= \frac{\overrightarrow{liver} + \overrightarrow{tacrolimus}}{2} \\
&= \frac{[0\;0\;1\;0\;0] + [1\;0\;1\;0\;0]}{2} \\
&= \frac{[1\;0\;2\;0\;0]}{2} \\
&= [0.5\;0\;1\;0\;0]
\end{aligned}$$

Any feature of c_i that is not present in the domain matrix cannot be used in evaluation of the kernel. However, since the kernel matrix represents a source of unsupervised knowledge in our case, we can rely on the supervised part of our algorithm to utilize such features that are found in the labeled training data, but not in the unlabeled data. Now the semantic kernel function to evaluate similarity of two arbitrary contexts is defined as:

$$K_{semantic}(c_i, c_j) = cos(\vec{c_i}, \vec{c_j})$$

where $\vec{c_i}$ and $\vec{c_j}$ are the two context vectors, created from the average of feature vectors as shown above.

The unigram LSA kernel is then defined as the sum of the semantic kernel above and the default linear kernel of SVMs, that evaluates the dot product of the supervised feature vectors of c_i and c_j. The supervised feature vectors of c_i and c_j are not the same as $\vec{c_i}$ and $\vec{c_j}$, since they are derived purely on the basis of labeled data. Mathematically, the unigram LSA kernel is defined as:

$$K_{unigram} = K_{linear}(c_i, c_j) + K_{semantic}(c_i, c_j)$$

The linear kernel represents the supervised learning component of our unigram LSA kernel and

Table 2: **A two-by-two contingency table for the bigram "star wars."**

	$wars$	$\overline{wars}$	
$star$	$n_{11} = 50$	$n_{12} = 25$	$n_{1+} = 75$
$\overline{star}$	$n_{21} = 75$	$n_{22} = 10,000$	$n_{2+} = 10,075$
	$n_{+1} = 125$	$n_{+2} = 10,025$	$n_{++} = 10,150$

the semantic kernel represents the unsupervised learning component. This unigram kernel function is used to solve the dual formulation of SVMs with kernels, shown in Section 2.6.6.

Bigram LSA Kernels Bigrams are pairs of words (possibly non-contiguous) in the context of the ambiguous word. Non-contiguous bigrams can contain one or more words in between the two words of the bigram. We control the number of words permitted in between two words of a bigram using a parameter of NSP.

The bigram kernel is created similarly to the unigram kernel except for the fact that in addition to using a list of stop words and frequency cut-off values for feature selection, a statistical measure of association of significance of bigrams is also used to eliminate spurious bigrams that are merely co-occurrences happening by chance. A statistical measure of association for bigrams determines to what degree two words are related to each other as opposed to being random co-occurrences. Applying a cut-off to the value obtained by a statistical measure of association forms a statistical test of association, which asserts with some degree of confidence the significance of a bigram (i.e., it not being a random co-occurrence).

The measure of association that we use is the Log Likelihood (LL) measure, which is available in NSP. The Log Likelihood score for a bigram [13] compares the actual counts of co-occurrence of a pair of words in the given corpus to the randomly expected co-occurrence count based on the frequency count of the individual words of the bigram. Consider a contrived example of a corpus containing the bigram "star wars," with frequency values as shown in Table 2.

Table 2 is known as a "contingency table" for a bigram and is interpreted as follows. n_{11} is the number of times the words "star" and "wars" occur together in the corpus. n_{12} is the number of times the word “star” occurs with some word other than “wars” in the corpus. n_{21} is the number of times the word “wars” occurs with some word other than “star” and n_{22} is the number of words other than “star” and “wars.” n_{1+} is the individual frequency count of the word “star” and n_{+1} is the individual frequency count of the word “wars.” n_{2+} is the number of words other than “star” and n_{+2} is the number of words other than “wars” in the corpus. Lastly, n_{++} is the total number of words in the corpus. The LL score (G^2) of the bigram in such a contingency table is defined as:

$$G^2 = \sum_{i,j} 2 * n_{ij} * log \frac{n_{ij}}{e_{ij}}$$

where e_{ij} represents the randomly expected frequency of co-occurrence of the two words in the bigram, which is calculated using the individual frequency count of each word in the bigram as follows:

$$e_{ij} = \frac{n_{i+} * n_{+j}}{n_{++}}$$

The LL score defined above can be shown to be distributed with a Chi-Squared (χ^2) distribution with degree of freedom equal to one. We therefore apply a cut-off of 3.842 to the LL score, which corresponds to 95% confidence in the significance of a bigram. Once the bigram features are identified, a bigram-by-context matrix D is created similar to the unigram-by-context matrix. Truncated SVD of D yields a domain matrix D_{LSA}. The context vector for any arbitrary context is created similar to the unigram LSA kernel, but using bigram features from the new context, and averaging their corresponding feature vectors from the domain matrix.

The semantic kernel function to evaluate similarity of two arbitrary contexts is again defined as:

$$K_{semantic}(c_i, c_j) = cos(\vec{c_i}, \vec{c_j})$$

where $\vec{c_i}$ and $\vec{c_j}$ are the two context vectors, created from the average of feature vectors as mentioned above.

The bigram LSA kernel is then defined as the sum of the semantic kernel above and the default linear kernel of SVMs, that evaluates the dot product of the supervised feature vectors of c_i and c_j.

Mathematically, the bigram LSA kernel is defined as:

$$K_{bigram} = K_{linear}(c_i, c_j) + K_{semantic}(c_i, c_j)$$

Co-occurrence LSA Kernels Co-occurrences are *unordered* bigrams. For example the two bigrams features “fine wine” and “wine fine” will form just one co-occurrence feature “fine wine.” Essentially any instance of the bigrams “fine wine” or “wine fine” will be treated as an instance of the co-occurrence “fine wine.” Also note that since the order does not matter, we could also have called the co-occurrence feature above “wine fine” as long as we have a consistent and unique way to refer to it. While taking frequency counts for co-occurrences, 4 occurrences of the bigram “fine wine” and 1 occurrence of the bigram “wine fine” will count at 5 occurrences of the co-occurrence “fine wine.”

Apart from the difference mentioned above, co-occurrence kernels are created exactly in the same manner as bigram kernels, with similar feature selection criteria and kernel function.

3.2.2 Word Association Kernels

Our word association kernels are based on the second-order context representation proposed by Purandare and Pedersen [49], which makes use of bigram or co-occurrence features.

Bigram Association Kernels Given a corpus of m documents, bigram features are identified for that corpus using the feature selection criteria mentioned above for bigram LSA kernels. This yields a list of bigrams and their LL scores of association. For every bigram (w_i, w_j), its first word w_i is represented along the row of an association matrix D and its second word w_j is represented along the column, with the matrix entry $D_{i,j}$ containing the score of association of the bigram (zero for all i, j pairs that were not identified as bigram features in the given corpus. The matrix D is therefore a set of feature vectors corresponding to the words from the corpus, showing what other words they co-occur with and their corresponding score of association. For example, for the set of five unlabeled contexts of the word *immunosuppression* shown earlier for unigram LSA kernels, if we choose bigram features where at most one word is permitted in between the two words of a bigram,

Table 3: **A word-by-word association matrix created using bigram features**

	graft	pancreatitis	ischemia	cancer	tumors
acute	4.0761	4.7322	0.0000	0.0000	0.0000
induce	4.0761	0.0000	0.0000	0.0000	0.0000
graft	0.0000	4.7322	0.0000	0.0000	0.0000
pretransplant	0.0000	0.0000	6.7902	0.0000	0.0000
warm	0.0000	6.7902	0.0000	0.0000	0.0000
skin	0.0000	0.0000	0.0000	5.0875	7.8252

then among several other bigrams, the following bigrams occur as features (with their corresponding LL score of association shown in parentheses): *acute graft (4.0761), induce graft (4.0761), acute pancreatitis (4.7322), graft pancreatitis (4.7322), pretransplant ischemia (6.7902), warm ischemia (6.7902), skin cancer (5.0875), skin tumors (7.8252).* Using these bigram features, the association matrix D is defined as in Table 3.

The difference from the unigrams LSA kernels introduced earlier is firstly that the words is this new representation are not unigrams, but the first half of bigrams, and secondly the dimensions of the feature vectors for these words are not contexts, but other significantly associated words from the corpus, that is the second half of the bigrams.

For bigrams features, this matrix D can be rectangular (as in our example) since the list of unique first words and unique second words of all bigram pairs may not be of the same size. Assuming that n bigrams are identified as features from the unlabeled corpus and that the number of unique first words is p and the number of unique second words is q, the size of D is $p \times q$, and thus each feature vector is initially q-dimensional. After applying SVD to the matrix D as in the case of unigram LSA kernels, these feature vectors are reduced to k dimensions. Using the k-dimensional feature vectors from the matrix D_k, any context from the labeled data can be represented as the average of the feature vectors from the rows of D_k, similar to the method used in unigram LSA kernels. The semantic kernel for any two contexts can be evaluated as the cosine between the two corresponding context vectors, just as in the case of unigram LSA kernels. The bigram association kernel function for two arbitrary contexts c_i and c_j is then defined as:

$$K_{ASSOC-bigram} = K_{linear}(c_i, c_j) + K_{semantic}(c_i, c_j)$$

4 Experiments

In this chapter we first describe the medical domain datasets that we have used for our experiments. Next we describe our experiments including their methodology and results followed by a discussion for each of the experiments. The goal of our experiments is to initially establish a competitive baseline using purely supervised learning algorithms and the best combination of features that yield the highest accuracy using these off-the-shelf algorithms. We then extensively explore the use of semantic kernels learned from unlabeled data for one of our datasets and compare the performance of our kernels with that of the off-the-shelf algorithms. Next we apply the best kernel methods from our first dataset to a second dataset and test the generality of our approach. In our experiments with abbreviations, we also focus on the feature engineering aspect, by making use of domain specific features and using a novel flexible window feature extraction approach and evaluating their contribution to the overall accuracy for abbreviation expansion.

4.1 Experimental Data

This section presents a description of three medical domain datasets for word sense disambiguation and automatic abbreviation expansion. First, the U.S National Library of Medicine (NLM) Word Sense Disambiguation (WSD) collection containing 50 ambiguous words; second, the MEDLINE Abbreviation (MA) collection containing 15 ambiguous abbreviations; and third, the Mayo Clinic Abbreviation (MCA) collection containing 16 ambiguous abbreviations. We have described the full datasets in each of the sub-sections below, however we make use of only a particular subset of each dataset for some of our experiments. The exact choice of a dataset or its subset for every experiment will be described in the corresponding experiment description.

4.1.1 NLM Dataset

The first dataset and the one that we have used most extensively is the biomedical word sense disambiguation test collection developed by Weeber et al., [54]. This WSD test collection is available from the National Library of Medicine (NLM)[13]. It contains 50 words that correspond to 50 ambiguous concepts from the UMLS [52] that are most frequently encountered in the 1998 collection of article abstracts in MEDLINE [36].

The UMLS consists of three knowledge sources related to biomedicine and health: First, the meta-thesaurus of biomedical and health related concepts such as names of diseases or agents causing them, for example *Chronic Obstructive Airway Disease* and *Virus*. Second, the UMLS includes a semantic network which provides a classification of these concepts and relationships among them. The relationships can be *hierarchical* as in "Acquired Abnormality **IsA** Anatomical Abnormality", which means that "Acquired Abnormality" is a type of "Anatomical Abnormality" or *associative* as in "Acquired Abnormality **affects** Cell Function." Third, the UMLS includes a SPECIALIST lexicon containing biomedical terms with their syntactic, morphological, and orthographic information. MEDLINE is a bibliographic database containing abstracts and references of several articles from journals related to life science. The NLM WSD collection developed by Weeber et al., [54] consists of 50 frequently encountered ambiguous words in the MEDLINE 1998 collection. While most of the words appear predominantly in noun form, there are also cases where they appear as adjectives or verbs. For example, the word *Japanese* occurs by itself as a noun meaning *the Japanese language* or *the Japanese people*, but more often as an adjective to describe people as in *the Japanese researchers* or *the Japanese patients*. Some words appear as verbs in their morphological variations, for example *discharge* appears as *discharged* and *determination* as *determined*. Each of the words has 100 randomly selected instances from the abstracts of 409,337 MEDLINE citations. Each instance provides two contexts for the associated word – the sentence from the abstract that contains the ambiguous word and the abstract of the article. The average size of the sentence context is 26 words and that of the abstract context is 220 words. The data is available in plain text format and follows some predefined formatting rules. Figure 3 shows a typical instance of an ambiguous term in the NLM WSD data collection. One of the datasets used by Liu et al., [35] and the dataset used by Leroy and Rindflesch [31] were subsets of this collection.

In addition to this manually annotated data collection, an unlabeled set of instances for each of the 50 words was made available by the National Library of Medicine. Each instance is an article

[13]http://wsd.nlm.nih.gov/

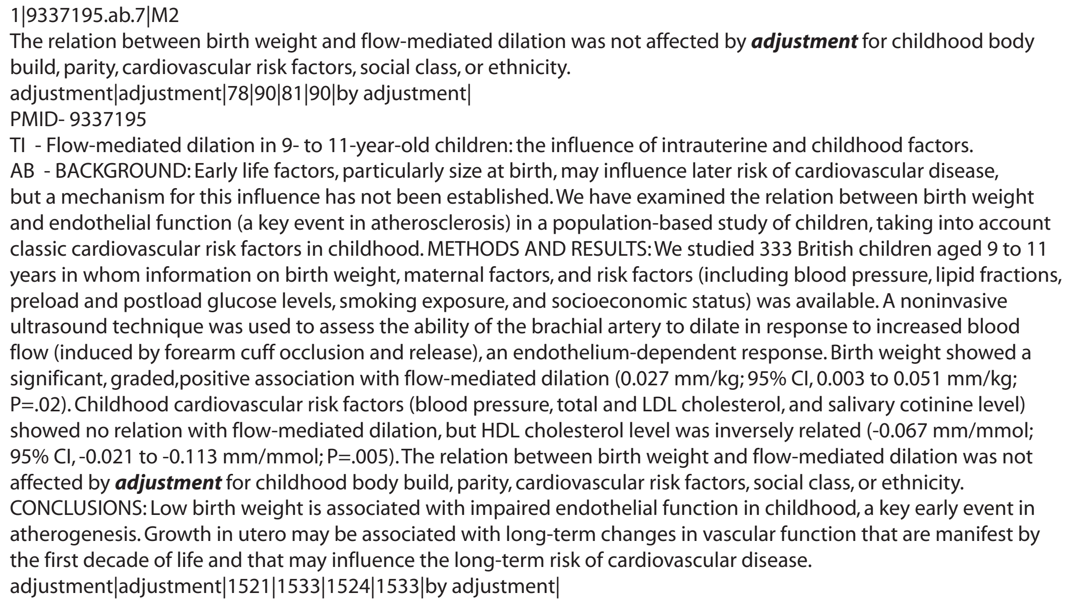
1|9337195.ab.7|M2
The relation between birth weight and flow-mediated dilation was not affected by ***adjustment*** for childhood body build, parity, cardiovascular risk factors, social class, or ethnicity.
adjustment|adjustment|78|90|81|90|by adjustment|
PMID- 9337195
TI - Flow-mediated dilation in 9- to 11-year-old children: the influence of intrauterine and childhood factors.
AB - BACKGROUND: Early life factors, particularly size at birth, may influence later risk of cardiovascular disease, but a mechanism for this influence has not been established. We have examined the relation between birth weight and endothelial function (a key event in atherosclerosis) in a population-based study of children, taking into account classic cardiovascular risk factors in childhood. METHODS AND RESULTS: We studied 333 British children aged 9 to 11 years in whom information on birth weight, maternal factors, and risk factors (including blood pressure, lipid fractions, preload and postload glucose levels, smoking exposure, and socioeconomic status) was available. A noninvasive ultrasound technique was used to assess the ability of the brachial artery to dilate in response to increased blood flow (induced by forearm cuff occlusion and release), an endothelium-dependent response. Birth weight showed a significant, graded, positive association with flow-mediated dilation (0.027 mm/kg; 95% CI, 0.003 to 0.051 mm/kg; P=.02). Childhood cardiovascular risk factors (blood pressure, total and LDL cholesterol, and salivary cotinine level) showed no relation with flow-mediated dilation, but HDL cholesterol level was inversely related (-0.067 mm/mmol; 95% CI, -0.021 to -0.113 mm/mmol; P=.005). The relation between birth weight and flow-mediated dilation was not affected by ***adjustment*** for childhood body build, parity, cardiovascular risk factors, social class, or ethnicity. CONCLUSIONS: Low birth weight is associated with impaired endothelial function in childhood, a key early event in atherogenesis. Growth in utero may be associated with long-term changes in vascular function that are manifest by the first decade of life and that may influence the long-term risk of cardiovascular disease.
adjustment|adjustment|1521|1533|1524|1533|by adjustment|

Figure 3: **A typical instance of an ambiguous term in the NLM WSD data collection. The example above shows an instance of the term *adjustment*.**

abstract containing the ambiguous word. The number of instances per word varies from as low as 402 for the word *mosaic* to 28,256 for the word *determination*. There are a total of 419,136 unlabeled instances for the 50 ambiguous words and the average size of an instance is 217 words.

Tables 4 and 5 show the distribution of different senses for each word in the collection and also the number of unlabeled instances available for each word. *M1* through *M5* are different sense labels for the senses of a word as defined in the UMLS repository. The last-but-one column with the sense label *None* stands for any sense other than those corresponding to labels *M1* through *M5*. The number of senses in the second column counts *None* as one of the senses and includes senses defined in UMLS even if they do not occur in the set of 100 labeled instances. A few salient features that can be observed from the distribution are as follows. Not every word has five senses defined in UMLS. Most of them have just two. A sense frequency count of zero in the table means that no instances of the corresponding sense were found in the set of 100 instances, but the sense is defined in UMLS. A '-' for some sense label for a word means that the sense label is not defined for the

Table 4: **The sense distribution for the ambiguous terms in the NLM WSD collection, the sense frequencies are out of 100. The last column shows the number of unlabeled instances available for each word.**

Word	**# Senses**	**Sense tag frequency**						**# Unlabeled instances**
		M1	M2	M3	M4	M5	None	
adjustment	4	18	62	13	-	-	7	1772
association	3	0	0	-	-	-	100	12182
blood_pressure	4	54	2	44	-	-	0	4767
cold	6	86	6	1	0	2	5	1417
condition	3	90	2	-	-	-	8	19384
culture	3	11	89	-	-	-	0	15398
degree	3	63	2	-	-	-	35	17694
depression	3	85	0	-	-	-	15	4335
determination	3	0	79	-	-	-	21	28256
discharge	3	1	74	-	-	-	25	3056
energy	3	1	99	-	-	-	0	5723
evaluation	3	50	50	-	-	-	0	11523
extraction	3	82	5	-	-	-	13	7092
failure	3	4	25	-	-	-	71	10049
fat	3	2	71	-	-	-	27	5940
fit	3	0	18	-	-	-	82	2426
fluid	3	100	0	-	-	-	0	6759
frequency	3	94	0	-	-	-	6	12157
ganglion	3	7	93	-	-	-	0	1184
glucose	3	91	9	-	-	-	0	5896
growth	3	37	63	-	-	-	0	20794
immunosuppression	3	59	41	-	-	-	0	954
implantation	3	17	81	-	-	-	2	2302
inhibition	3	1	98	-	-	-	1	13557
japanese	3	6	73	-	-	-	21	1662

word. For example, *cold* has 6 senses, and one of them (corresponding to the label *M4*) does not occur even once in the set of 100 labeled instances, whereas *adjustment* has only four senses, *M4* and *M5* are not defined for it. Every word has *None* as one of the possible senses, which means that while manually tagging the data instances, an instance which cannot be categorized into any of the known concepts as defined in UMLS can be assigned this default sense. This sense therefore actually represents the “unknown” sense, which is not present in UMLS. However we adhere to the sense tag “None” used by Weeber et al., [54]. Although the machine learning methods will see all the instances of an ambiguous word with the sense tag “None” as having the *same* sense, the

Table 5: **The sense distribution for the ambiguous terms in the NLM WSD collection (continued from Table 4). The word *mosaic* has two senses that are very closely related and were assigned the same label *M2*. The last column shows the number of unlabeled instances available for each word.**

Word	# Senses	Sense tag frequency						# Unlabeled instances
		M1	M2	M3	M4	M5	None	
lead	3	27	2	-	-	-	71	7088
man	4	58	1	33	-	-	8	3852
mole	4	83	1	0	-	-	16	1797
mosaic	4	45	**52**	*	0	-	3	402
nutrition	4	45	16	28	-	-	11	2637
pathology	3	14	85	-	-	-	1	3217
pressure	4	96	0	0	-	-	4	8410
radiation	3	61	37	-	-	-	2	4386
reduction	3	2	9	-	-	-	89	15853
repair	3	52	16	-	-	-	32	3960
resistance	3	3	0	-	-	-	97	8814
scale	4	0	65	0	-	-	35	5719
secretion	3	1	99	-	-	-	0	7228
sensitivity	4	49	1	1	-	-	49	21304
sex	4	15	5	80	-	-	0	6522
single	3	1	99	-	-	-	0	22447
strains	3	1	92	-	-	-	7	6741
support	3	8	2	-	-	-	90	16849
surgery	3	2	98	-	-	-	0	14767
transient	3	99	1	-	-	-	0	5526
transport	3	93	1	-	-	-	6	5527
ultrasound	3	84	16	-	-	-	0	3082
variation	3	20	80	-	-	-	0	7781
weight	3	24	29	-	-	-	47	14121
white	3	41	49	-	-	-	10	4467

features present in such instances will often be an entirely random mixture representing multiple other unknown senses. This effect will be more pronounced in the cases where the *None* sense covers almost 50 percent of the instances or greater. These instances introduce significant noise into the data. Therefore, for such words the performance of machine learning methods might degrade. The *majority sense* for an ambiguous word is the sense that occurs most frequently in a given set of labeled data. Half of the words in the NLM dataset have a majority sense that covers 80 percent of the instances, making their sense distribution highly skewed. Finally, a note regarding the word

Table 6: **The sense distribution for the ambiguous abbreviations in the MEDLINE Abbreviation dataset.**

Abbreviation	Expansions	Count	Dominance(%)
APC (2310)	Antigen Presenting Cells	1356	58.70
	Activated Protein C	479	20.74
	Adenomatous Polyposis Coli	430	18.61
	Aphidicholin	37	1.60
	Atrial Premature Complexes	8	0.0035
ASP (141)	Aspartate	60	42.55
	Antisocial Personality	54	38.29
	Asparaginase	17	12.06
	Aspartic Acid	8	5.67
	Ankylosing Spondylitis	2	1.42
BPD (906)	Bronchopulmonary Dysplasia	465	51.32
	Biparietal Diameter	233	25.72
	Borderline Personality Disorder	208	22.96
BSA (3162)	Bovine Serum Albumin	2,808	88.80
	Body Surface Area	354	11.20
DIP (112)	Distal Interphalangeal	81	72.32
	Desquamative Interstitial Pneumonia	31	27.68
FDP (431)	Fibrinogen Degradation Product	382	88.63
	Flexor Digitorum Profundus	39	9.05
	Fructose Diphosphate	8	1.86
	Formycin Diphosphate	2	0.0046
LAM (183)	Lipoarabinomannan	103	56.28
	Lymphangioleiomyomatosis	56	30.60
	Lymphangiomyomatosis	22	12.02
	Leukocyte Adhesion Molecule	2	1.09
MAS (112)	Meconium Aspiration Syndrome	81	72.32
	Mccune Albright Syndrome	31	27.68

mosaic: two of its senses are very closely related, *M2* (Mosaicism) and *M3* (Embryonic Mosaic). They were therefore assigned the same label *M2* during manual sense tagging. This sense covers 52 instances, which are listed in the column *M2*.

4.1.2 MEDLINE Abbreviation Dataset

The second dataset that we use in our experiments is the one introduced by Liu et al., [35], containing 15 ambiguous abbreviations. This collection contains automatically sense tagged instances

Table 7: **The sense distribution for the ambiguous abbreviations in the MEDLINE Abbreviation dataset (continued from Table 6).**

Abbreviation	**Expansions**	**Count**	**Dominance(%)**
MAC (862)	Mycobacterium Avium Complex	535	62.06
	Membrane Attack Complex	231	26.80
	Macrophage	40	4.64
	Mental Adjustment to Cancer	19	2.20
	Monitored Anesthesia Care	19	2.20
	Macandrew Alcoholism Scale	18	2.09
MCP (461)	Monocyte Chemoattractant Protein	185	40.13
	Metoclopramide	157	34.06
	Membrane Cofactor Protein	102	22.13
	Multicatalytic Protease	9	1.95
	Metacarpophalangeal Joint	8	1.73
PCA (1553)	Patient Controlled Analgesia	507	32.65
	Passive Cutaneous Anaphylaxis	376	24.21
	Principal Component Analysis	343	22.09
	Para Chloroamphetamine	210	13.52
	Posterior Cerebral Artery	112	7.21
	Posterior Communicating Artery	5	0.0032
PCP (2225)	Phencyclidine	1,071	48.13
	Pneumocystis Carinii Pneumonia	812	36.49
	Pentachlorophenol	341	15.33
	P Chlorophenylalanine	1	0.00044
PEG (70)	Polyethylene Glycols	52	74.29
	Percutaneous Endoscopic Gastrostomy	18	25.71
PVC (571)	Polyvinylchloride	473	82.84
	Premature Ventricular Contraction	98	17.16
RSV (1954)	Respiratory Syncytial Virus	1,335	68.32
	Rous Sarcoma Virus	619	31.68

of ambiguous abbreviations from the MEDLINE abstracts. The approach that was used to create a superset of this dataset is described in Liu et al. [33], where sense tagged data for 35 three-letter abbreviations was automatically created using the simple yet effective idea of searching for abbreviations that followed in a parenthetical expression after one of their possible expansions obtained from the UMLS. For example, if a MEDLINE abstract contained the fragment "... *Atrial Premature Complexes (APC)* ...", and *Atrial Premature Complexes* is known to be one of the possible expansions of *APC* from the UMLS knowledge base, then the abstract is selected as a sense tagged instance of *APC* and stored after replacing "Atrial Premature Complexes (APC)" with just "APC." The subset of 15 abbreviations that we and Liu et al., [35] utilize was chosen after eliminating 18 abbreviations that had a most frequent sense greater than 90% and one abbreviation *EMG* that had three very closely related expansions out of four (*exomphalos macroglossia gigantism, electromyography, electromyographs* and *electromyogram*), where the last three senses are closely related.

Tables 6 and 7 show the sense distribution for the 15 abbreviations in the MEDLINE abbreviation dataset. The abbreviation and the total number of its instances are shown in the first column, then its expansions and associated instance counts are shown in the second and third column respectively. We show the dominance of each expansion in the fourth column, which is simply the percentage of the total instances that have the given expansion.

4.1.3 Mayo Clinic Abbreviation Dataset

The third dataset that we have used consists of 7,738 instances of 16 ambiguous abbreviations that have been manually disambiguated. This corpus is derived from the Mayo Clinic database of clinical notes. Nine of the abbreviations were annotated in a previous study by Pakhomov et al., [45] while the remaining seven were newly annotated for our work presented in Joshi et al., [26]. The annotation process was similar to the previously reported one by Pakhomov et al., [45] with the exception that the current process was based on the entire database of 17 million notes spanning years 1994–2005 instead of a 1.7 million subset only from the year 2002.

Tables 8 and 9 summarize the data for the 16 abbreviations. The abbreviation and the total number of instances are shown in the first column, and then the top three expansions and associated instance counts are shown in the second and third column. If there are more than three expansions for an abbreviation, then we combine the counts of the remaining expansions into a single row. We show the dominance of each expansion in the fourth column, which is simply the percentage of the total instances that have the given expansion.

In the next six sub-sections we describe the six different types of experiments we have performed. For each type of experiment we sub-divide the section according to the dataset used for the experiment and for each dataset we elaborate on the particular subset of the data used, the experi-

Table 8: **The sense distribution for the ambiguous abbreviations in the clinical notes data from the Mayo Clinic.**

Abbreviation	Top 3 Expansions	Count	Dominance(%)
AC (464)	Acromioclavicular	146	31.47
	Antitussive with Codeine	139	29.96
	Acid Controller	109	23.49
	10 more expansions	70	15.08
APC (376)	Argon Plasma Coagulation	157	41.76
	Adenomatous Polyposis Coli	94	25.00
	Atrial Premature Contraction	55	14.63
	10 more expansions	70	18.62
LE (615)	Limited Exam	291	47.32
	Lower Extremity	270	43.90
	Initials	44	7.15
	5 more expansions	10	1.62
PE (519)	Pulmonary Embolism	251	48.36
	Pressure Equalizing	160	30.82
	Patient Education	48	9.24
	12 more expansions	60	11.56
CP (578)	Chest Pain	321	55.54
	Cerebral Palsy	110	19.03
	Cerebellopontine	88	15.22
	19 more expansions	59	10.21
HD (254)	Huntington's Disease	142	55.91
	Hemodialysis	75	29.52
	Hospital Day	22	8.66
	9 more expansions	15	5.91
CF (710)	Cystic Fibrosis	530	74.65
	Cold Formula	101	14.22
	Complement Fixation	36	5.07
	6 more expansions	43	6.06
MCI (344)	Mild Cognitive Impairment	269	78.20
	Methylchloroisothiazolinone	34	9.88
	Microwave Communications, Inc.	18	5.23
	5 more expansions	23	6.69

mental methodology, the results and finally a discussion related to the experiment.

Table 9: **The sense distribution for the ambiguous abbreviations in the clinical notes data from the Mayo Clinic (continued from Table 8).**

Abbreviation	Top 3 Expansions	Count	Dominance(%)
ID (574)	Infectious Disease	450	78.40
	Identification	105	18.29
	Idaho	7	1.21
	Identified	7	1.21
	4 more expansions	5	0.87
LA (488)	Long Acting	385	78.89
	Person	53	10.86
	Left Atrium	17	3.48
	5 more expansions	33	6.76
MI (690)	Myocardial Infarction	590	85.51
	Michigan	96	13.91
	Unknown	2	0.29
	2 more expansions	2	0.29
ACA (541)	Adenocarcinoma	473	87.43
	Anterior Cerebral Artery	62	11.46
	Anterior Communication Artery	3	0.006
3	more expansions	3	0.006
GE (591)	Gastroesophageal	521	88.15
	General Exam	40	6.77
	Generose	22	3.72
	General Electric	8	1.35
HA (509)	Headache	470	92.34
	Hearing Aid	30	5.89
	Hydroxyapatite	6	1.18
	2 more expansions	3	0.59
FEN (80)	Fluids, Electrolytes and Nutrition	78	97.5
	Drug Fen Phen	1	1.25
	Unknown	1	1.25
NSR (405)	Normal Sinus Rhythm	401	99.01
	Nasoseptal Reconstruction	4	0.99

4.2 Baseline Experiments

The first type of experiments that we have performed on all of our datasets are those that will serve as the baseline for our experiments using semantic kernel methods. This section describes the baseline experiments for all three of our datasets.

Note that as of now we have not performed semantic kernel related experiments on our third dataset, the abbreviation dataset from the Mayo Clinic. However, we have performed some feature engineering experiments on that dataset, including use of features specific to the clinical notes and we will present those results as a baseline for future semantic kernel based experiments that we plan to perform on the Mayo Clinic abbreviation dataset.

The baseline to compare the results obtained in our baseline experiments is the output of the majority classifier. A majority classifier simply assigns the most frequently occurring sense of a word to all the instances. For example, referring to Table 4, the majority classifier would classify all the 100 instances of the word *adjustment* as belonging to the sense label *M2*, thus achieving an accuracy of 62%.

4.2.1 NLM Dataset

This section presents the results from our previous work [28] on the NLM dataset.

Data The first set of baseline experiments were conducted on the entire collection of 50 ambiguous words from the NLM dataset. However, in order to evaluate performance of different feature representations and different off-the-shelf classifiers, we have considered only 30 words out of 50. The twenty words *association, cold, condition, energy, extraction, failure, fluid, frequency, ganglion, glucose, inhibition, pathology, pressure, reduction, resistance, secretion, single, surgery, transient* and *transport* were excluded from analysis since none of the classifiers we used could achieve an improvement of five percentage points or more over the majority classifier. We believe that in these cases the algorithms could not achieve considerable improvement over the majority classifier since most of these words have a majority sense that exceeds 80 percent and therefore has a very skewed distribution which provides few instances of senses other than the majority sense. Although the word *failure* does not have an overly skewed distribution, it has a very high number of instances belonging to the sense *None*, which might have degraded the performance of classifiers due to the reason discussed earlier in Section 4.1.1.

Methodology For each of the 50 words, 2 formations of the datasets were available, one consisting of the features based on the sentence contexts and the other consisting of the features based on the abstract contexts.

We have used five different classifiers from the WEKA data mining suite, with their default settings, as the baseline machine learning algorithms: (1) the Support Vector Machine implementation in WEKA with the default linear kernel, (2) the naïve Bayes classifier, (3) the J48 decision tree implementation of the C4.5 decision tree algorithm, (4) the PART decision list algorithm and finally (5) a classifier ensemble approach in the form of the AdaBoostM1 algorithm, using a DecisionStump (a single node decision tree classifier) as the base weak learner. We have performed one run of 10-fold cross-validation for each classifier.

The features used in these experiments were unigrams and bigrams in the entire available context of the ambiguous word, that is the entire sentence or the entire abstract. Feature selection for unigrams was done based on simple elimination of standard English stop words and a frequency cutoff ranging from two to five. A frequency cutoff of two rejects any unigram feature that occurs less than two times in the entire set of instances of an ambiguous word. Our bigram features can be non-contiguous. The two component words of a bigram could occur within some number of words of each other (a window). A window of three means zero or one other word can occur in between the component words of the bigram. Similarly a window of four means zero, one or two other words can occur in between the component words of the bigram and a window of two means the component words should be contiguous. We varied this window size from two to five for the bigrams. For bigram feature selection we used the standard English words stop list in a disjunctive mode, so that a bigram feature is discarded when either one or both the component words of the bigram are in the stop list. Additionally a frequency cutoff ranging from two to five was applied. In order to avoid selecting randomly co-occurring bigrams, we also applied the Log Likelihood test to eliminate insignificant bigrams. The Log Likelihood cutoff values that we used are 3.841 and 6.635, which correspond to 95% and 99% confidence in the significance of the bigrams respectively.

Thus, in all we performed $18,000$ experiments (50 words $\times$ 2 contexts $\times$ 5 classifiers) $\times$ (4 unigram type features + 4 $\times$ 4 $\times$ 2 bigram type features).

Note again however that we provide overall evaluation in the Results section below for only 30 significant words, corresponding to $10,800$ experiments.

Results We have presented all of our results in the form of graphs comparing the accuracy of the methods that we have used for various settings. Accuracy is simply the percentage of correctly classified instances. For evaluating statistical significance of our results, we have used a paired t–test with a two-tailed confidence interval of 0.05. While reporting our results, we show the 95% confidence intervals on our graphs and report the corresponding $p\text{-}values$.

Here we present results pertaining to the overall analysis of the 30 significant words. A best case analysis and detailed results for each significant word can be found in our previous paper [28].

Figures 4 and 5 show the average accuracy of the five classifiers from WEKA across the 30 chosen words and 36 feature representations, in the abstract and the sentence context respectively.

Figures 6 and 7 show the average accuracy obtained using 36 different feature representations across 30 significant words and five WEKA classifiers, in the abstract and the sentence context respectively.

The best feature representation for the abstract context was unigrams with a frequency cutoff of five and for the sentence context was unigrams with a frequency cutoff of four. The improvement in performance using unigrams over the best type of bigram features is statistically significant ($p < 0.01$ for the abstract context and for the sentence context). However, the difference between the top two unigram performances is not statistically significant ($p = 0.522$ for the abstract context and $p = 0.848$ for the sentence context). Figures 8 and 9 show the classifier performance when the best feature representation mentioned above are used for each of the contexts.

Discussion The general observations that can be made from these baseline experiments are as follows. First, from Figures 8 and 9, the performance of all classifiers in abstract context is better than that in the sentence context. Second, as mentioned before unigram features give better performance across all classifiers.

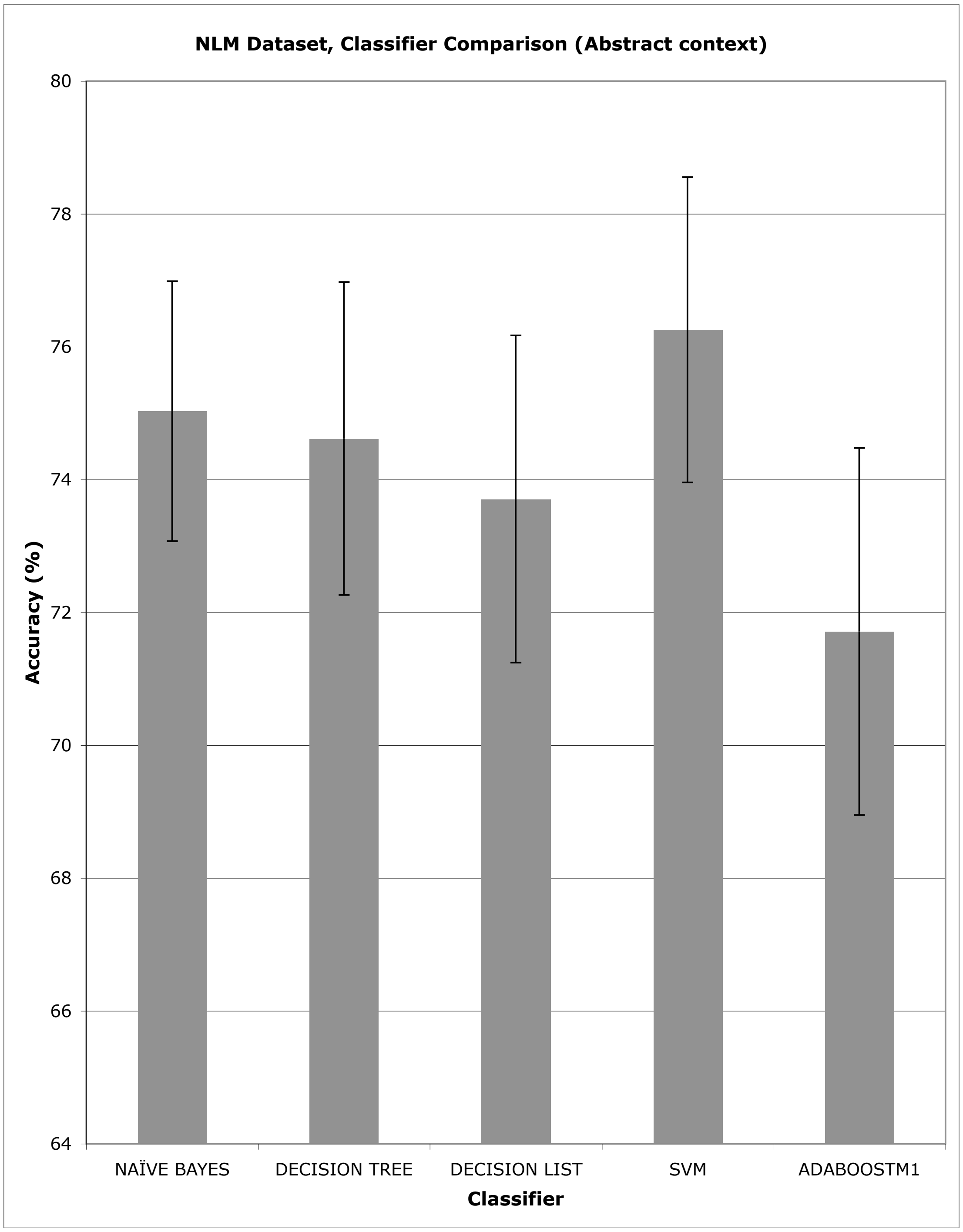

Figure 4: **A comparison of the average performance of five classifiers across 30 words from the NLM dataset and 36 feature representations, with the abstract context. The graph shows the average accuracy value for each classifier along with the 95% confidence interval using the error bars.**

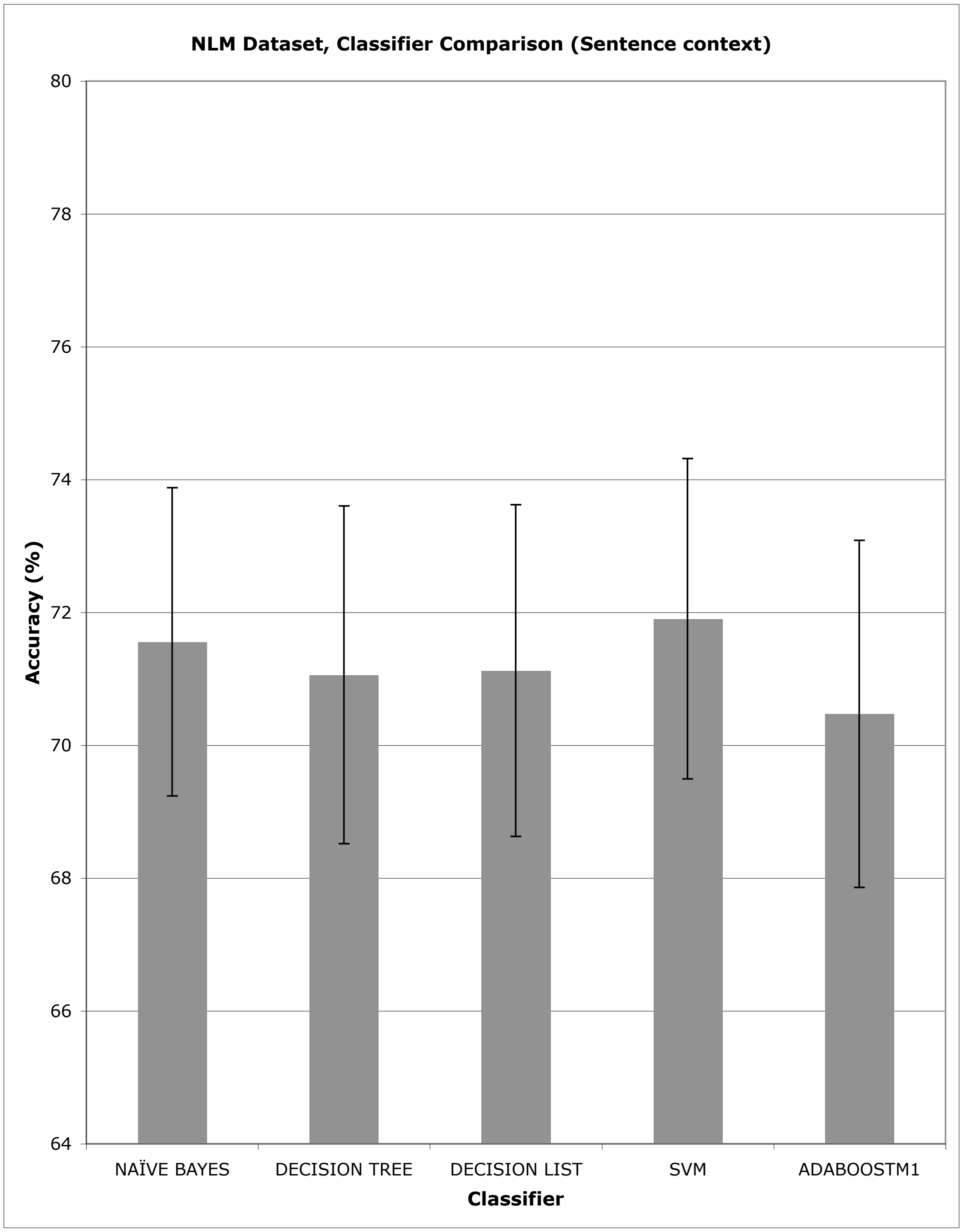

Figure 5: **A comparison of the average performance of five classifiers across 30 words from the NLM dataset and 36 feature representations, with the sentence context. The graph shows the average accuracy value for each classifier along with the 95% confidence interval using the error bars.**

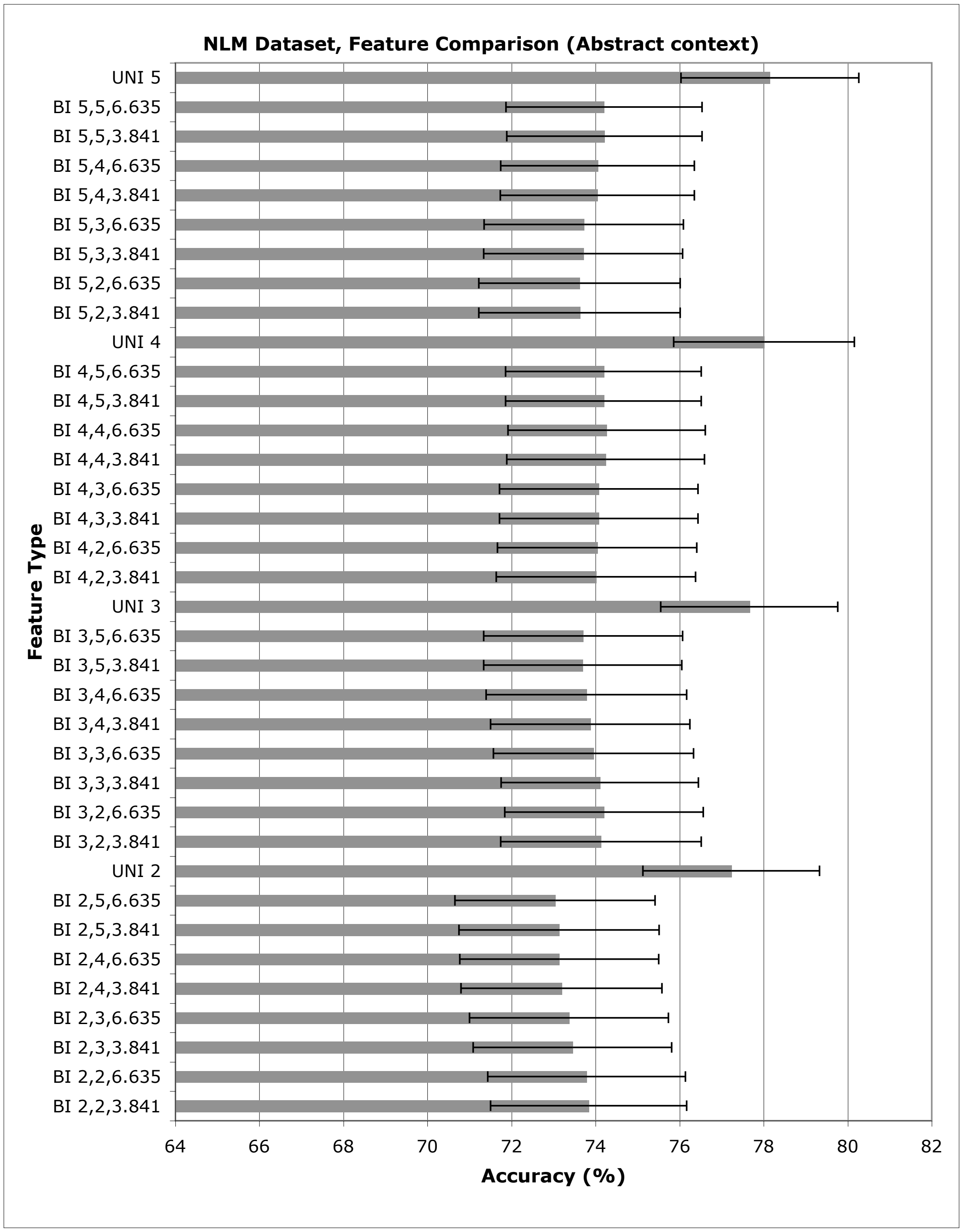

Figure 6: **A comparison of the average performance of 36 feature representations across 30 words from the NLM dataset and five classifiers, with the abstract context. The graph shows the average accuracy value for each feature representation along with the 95% confidence interval using the error bars.**

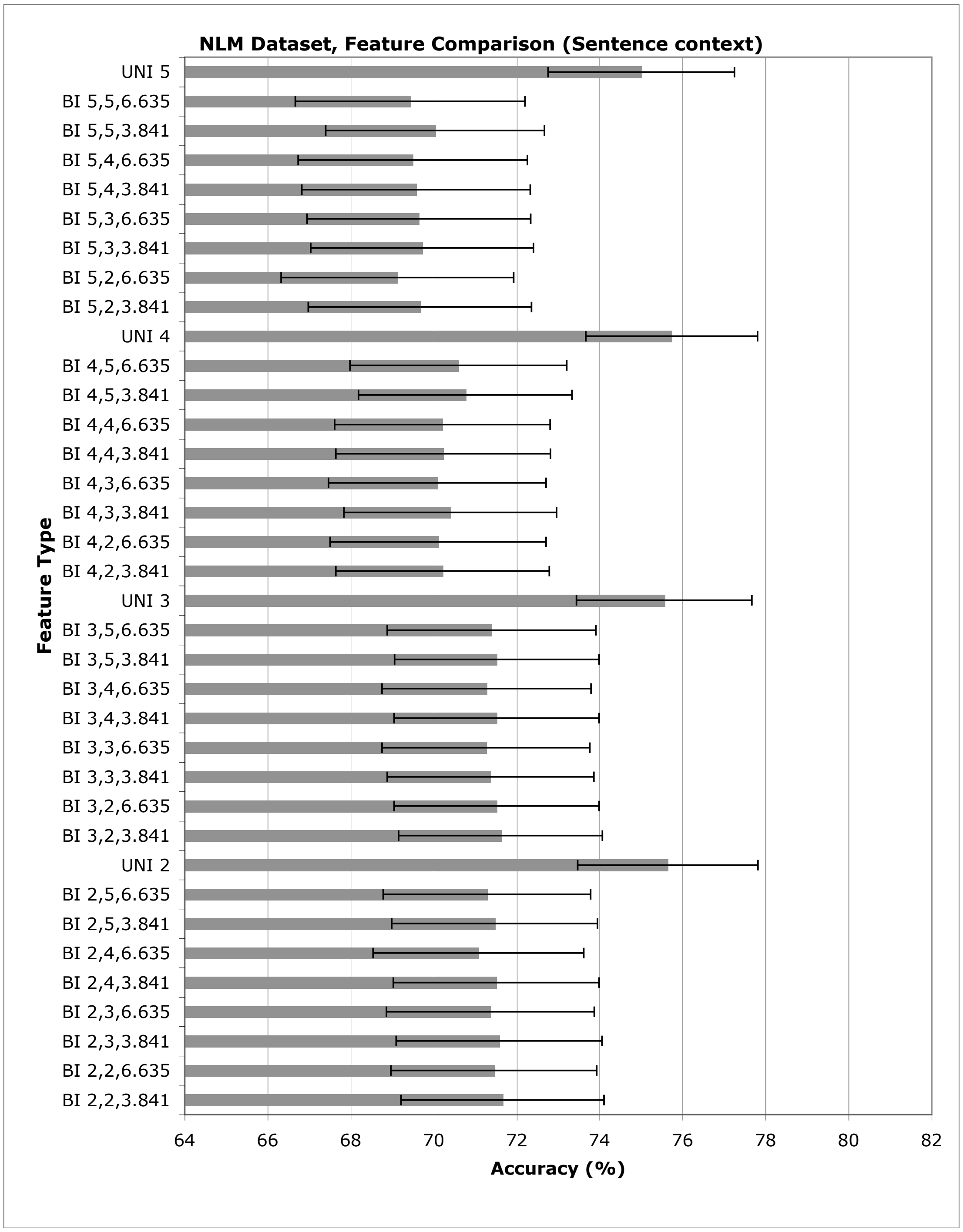

Figure 7: **A comparison of the average performance of 36 feature representations across 30 words from the NLM dataset and five classifiers, with the sentence context. The graph shows the average accuracy value for each feature representation along with the 95% confidence interval using the error bars.**

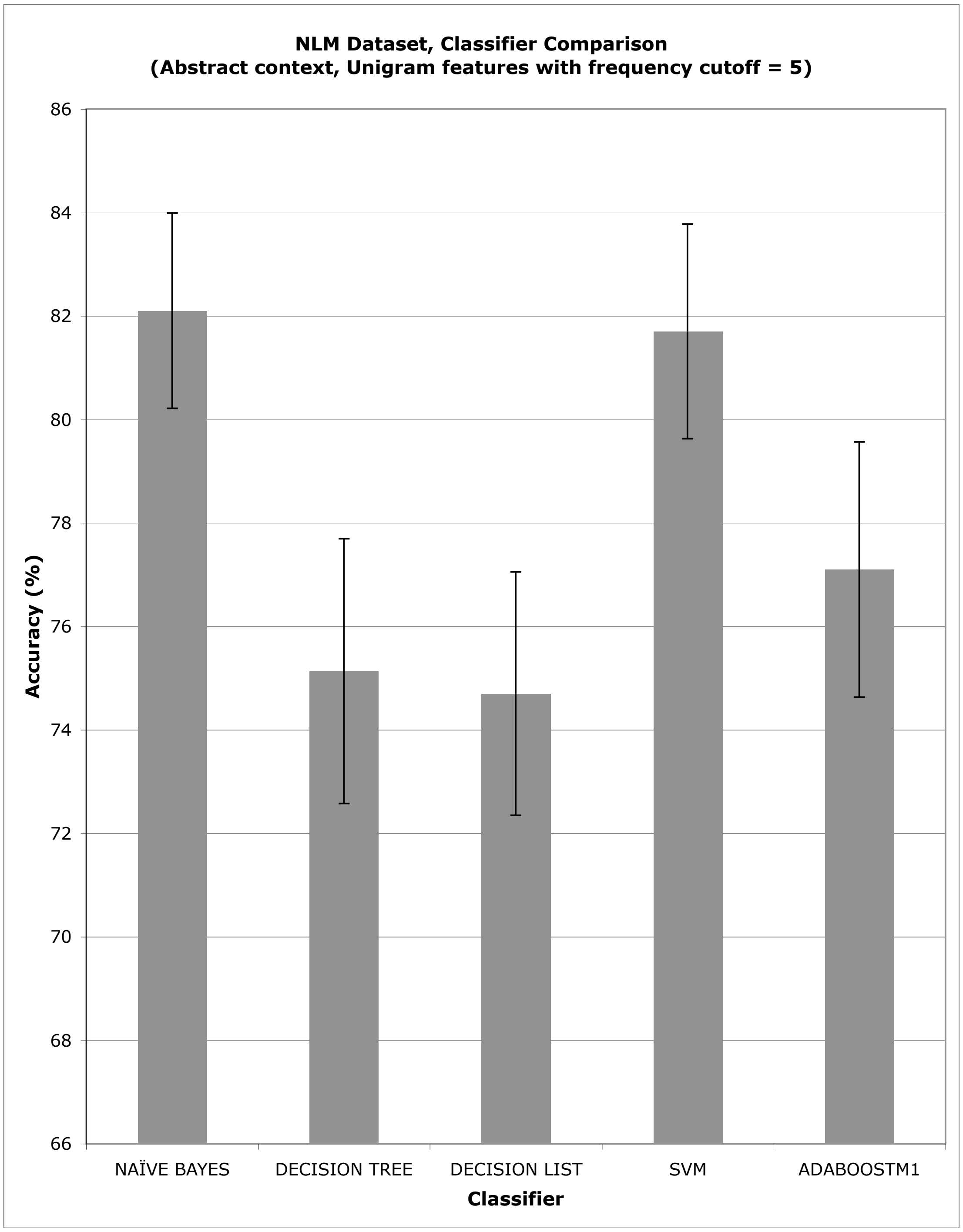

Figure 8: **A comparison of the average performance of five classifiers across 30 words from the NLM dataset, using unigram features with a frequency cutoff of five, with the abstract context. The graph shows the average accuracy value for each classifier along with the 95% confidence interval using the error bars.**

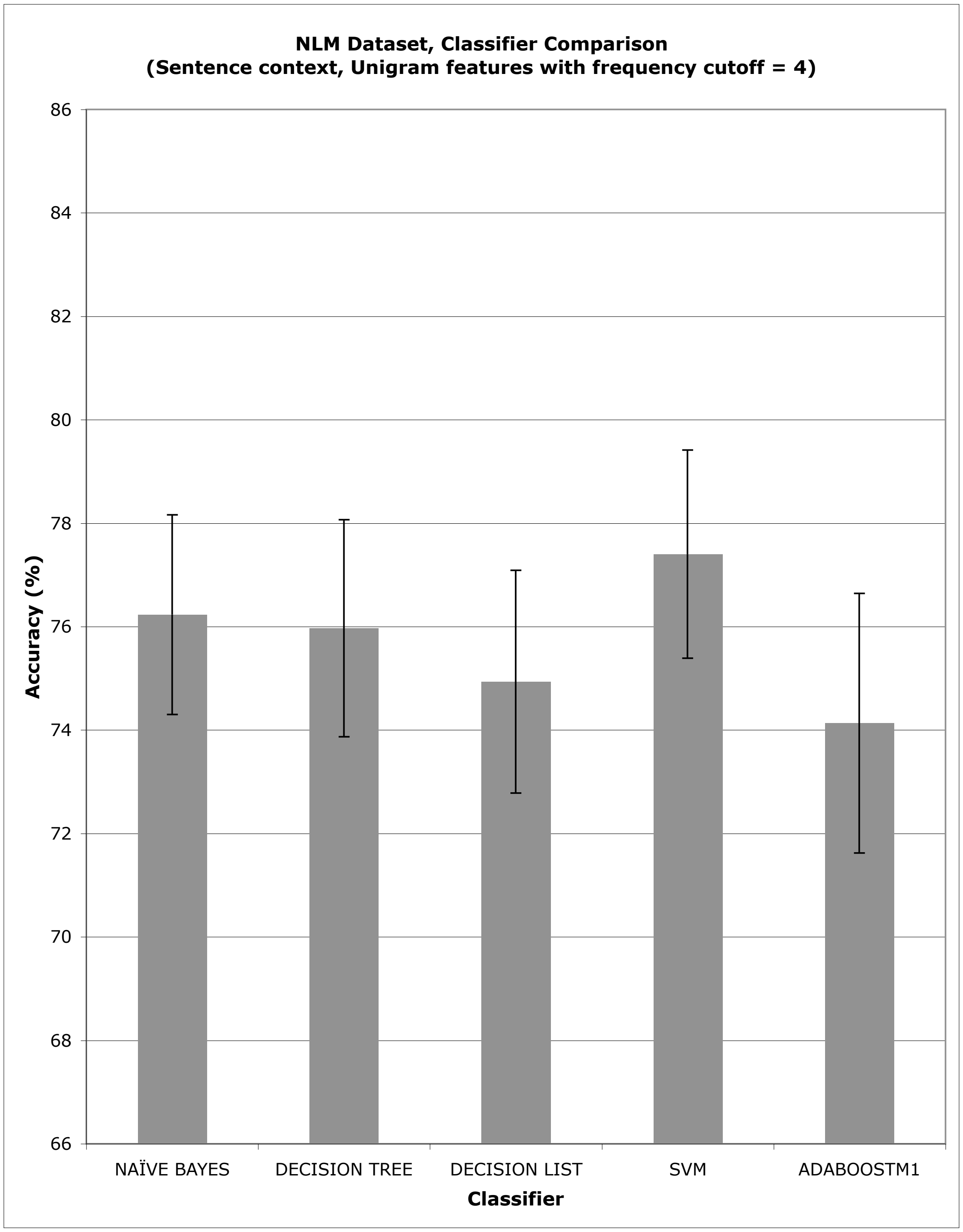

Figure 9: **A comparison of the average performance of five classifiers across 30 words from the NLM dataset, using unigram features with a frequency cutoff of four, with the sentence context. The graph shows the average accuracy value for each classifier along with the 95% confidence interval using the error bars.**

4.2.2 MEDLINE Abbreviation Dataset

This section presents the results of using the best feature representation found in the NLM dataset experiments with the same five machine learning algorithms, on the MEDLINE abbreviation dataset.

Data The MEDLINE Abbreviation dataset described in Section 4.1.2 does not have an accompanying unlabeled dataset as in the case of the NLM WSD collection. We have therefore randomly split the instances for each abbreviation into two parts – one containing 75% of the instances and the other containing the remaining 25%. We treat the portion containing 75% of the instances as unlabeled data, ignoring the sense annotations present in those instances and the remaining 25% as labeled data for our experiments. In order to have at least 100 labeled instances of each abbreviation, we only use a subset of ten abbreviations that have 400 instances or more. The abbreviations that we use are *APC, BPD, BSA, FDP, MAC, MCP, PCA, PCP, PVC* and *RSV*.

Methodology We have one dataset for each of the ten abbreviations, containing the entire abstract in which the abbreviation appears as the context.

We use the same five WEKA classifiers as for the NLM dataset, with the default settings. However, we have performed ten runs of 10-fold cross-validation for each of the classifiers. For these experiments we only use the best type of feature representation found in our NLM dataset experiments for the abstract context, unigrams with a frequency cutoff value of five.

Thus, in all we have performed 500 experiments (10 abbreviations $\times$ 1 context $\times$ 5 classifiers $\times$ 1 unigram type features $\times$ 10 runs of 10-fold cross-validation)

Results Figure 10 shows the accuracy of the five classifiers that we have evaluated, averaged over the ten abbreviations and ten runs of 10-fold cross-validation.

Discussion The naïve Bayes classifier and the Support Vector Machine are clearly the best performers for the MEDLINE abbreviation dataset. SVMs show higher accuracy on average, with a very low standard error. However, the difference between the naïve Bayes classifier and SVMs is

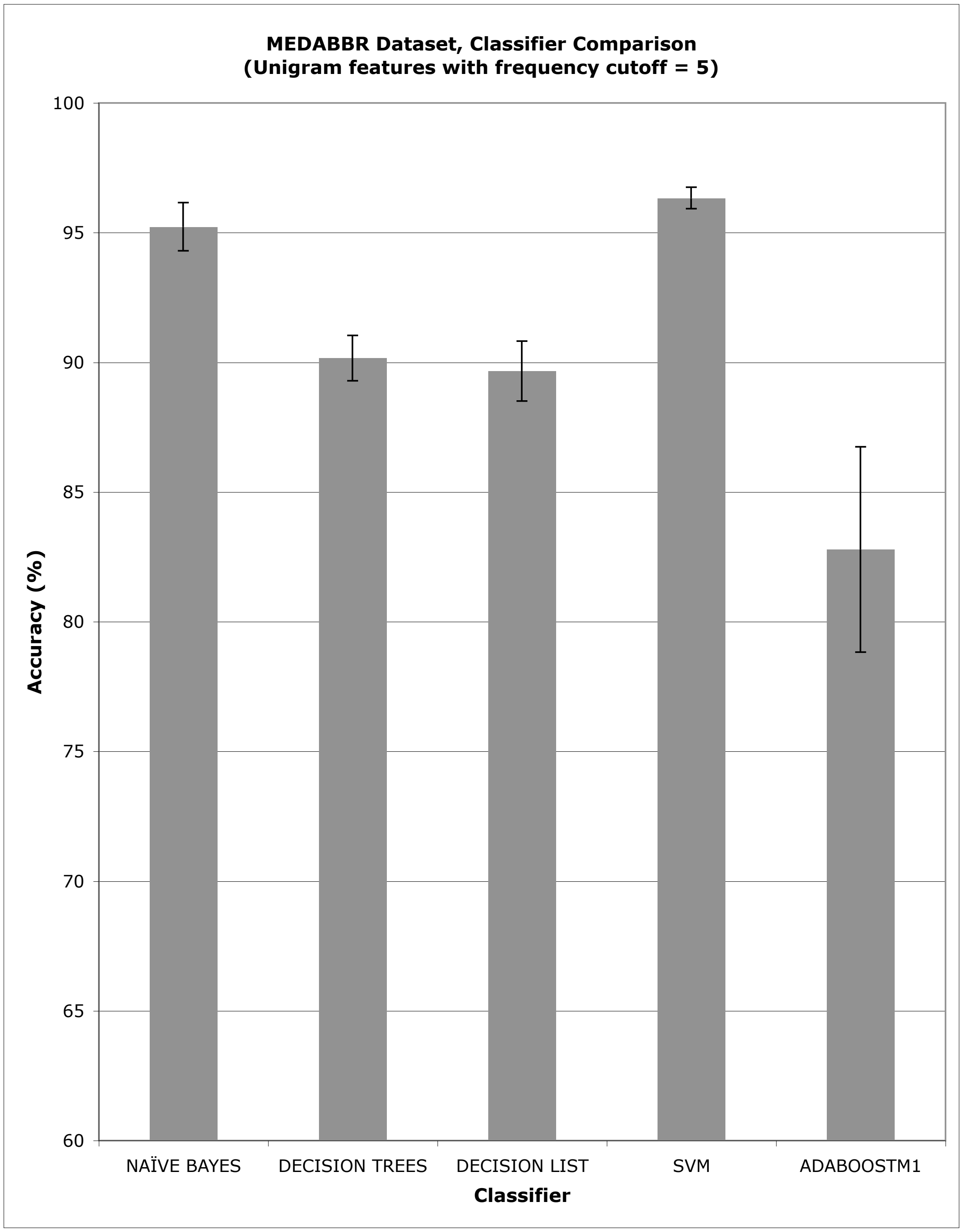

Figure 10: **A comparison of the average performance of five classifiers across 10 abbreviations from the MEDLINE dataset, using unigram features with a frequency cutoff of five. The graph shows the average accuracy value for each classifier, along with the 95% confidence interval using the error bars.**

not statistically significant on these set of observations ($p = 0.092$). Both the naïve Bayes classifier and SVMs show statistically significant improvement over the next best algorithm, which is the decision tree learner ($p < 0.001$ for the naïve Bayes classifier and for SVMs).

4.2.3 Mayo Clinic Abbreviation Dataset

In this section, we present results on the dataset of ambiguous abbreviations from the Mayo Clinic. The first set of experimental results are those presented in Joshi et al., [26]. The second set of experiments include the use of features specific to the domain of clinical notes.

Data The data used for the first set of experiments is the entire collection of 16 abbreviations from the Mayo Clinic database of clinical notes. The second set of experiments are conducted on a subset of nine acronyms – *AC, ACA, APC, CF, HA, HD, LA, NSR* and *PE*.

Methodology In both the set of experiments we have used three WEKA classifiers – the decision tree classifier J48, the naïve Bayes classifier and the linear kernel SVMs with their default settings, performing one run of 10-fold cross-validation. For both sets of experiments, we have focused on the feature engineering aspect for this dataset.

In all the experiments on the Mayo Clinic abbreviation dataset, the bigram features are contiguous (i.e., no other words can be present in between the component words of the bigram).

For the first set of experiments, we have used the following type of features, individually by themselves and then combining them all together. First, we use unigrams occurring within some window size of words around the ambiguous abbreviation. The window size we have used is five words to the left and five words to the right of the abbreviation, not crossing the boundary of the context for an abbreviation, where the context is one clinical note record in the database (i.e., we do not use words from previous or following notes). Our windowing approach is a *flexible window* approach as we do not physically fix the window to five words, but allow it to be flexible and expand beyond five positions when the intermediate words are rejected by the unigram selection criteria. The unigram selection criteria uses a frequency cutoff value of five. Second, we use bigrams within a flexible window of five along with a frequency cutoff value of five. Third, we use Parts-of-Speech tags of two words to the left and to the right of the ambiguous abbreviation as well as the Part-of-Speech tag of the abbreviation itself as features. Finally, we also use a combination of all of the above three feature representations.

For the second set of experiments that was conducted on a subset of nine abbreviations out of 16, our aim was first to evaluate if using the flexible window improved performance and second to make use of features specific to the clinical notes domain and verify if they improved accuracy.

For comparing flexible and fixed window approaches, we make use of unigram features with a frequency cutoff of five and fixed and flexible window sizes ranging from one through ten. Similarly for bigrams we use a frequency cutoff of five, a log-likelihood score cutoff of 3.842 (corresponding to the 95% confidence in significance of bigrams) and fixed and flexible window sizes ranging from one through ten. Finally, we have combined the unigram and bigram features in the corresponding window sizes, that is unigrams and bigrams with frequency cutoff of five, and a Log Likelihood score cutoff of 3.841 for bigrams and fixed and flexible window sizes ranging from one through ten.

For evaluating the impact of features specific to clinical notes, we chose three feature configurations from the above set of flexible window experiments, one corresponding to the best performance of each of the classifiers. The three configurations were unigrams and bigrams in a flexible window of two, nine and ten. We have made use of three clinical-notes related features. First is the gender code of the patient (M/F) associated with the clinical note in which the abbreviation occurs. Second is the department code from where the clinical note originated. Third is the section identifier inside the clinical note in which the abbreviation has occurred.

Apart from the clinical note features, we also added Part-of-Speech tags of two words to the left and right of the abbreviation and that of the abbreviation itself as features in the second set of experiments.

Results The baseline that we have used for these experiments is the majority classifier accuracy, which is shown as a dotted line in each of our result graphs for the first set of experiments. Figures 11 through 13 show the accuracy values for the three classifiers for three groups of abbreviations (based on their average majority sense), for the four feature representations we have used.

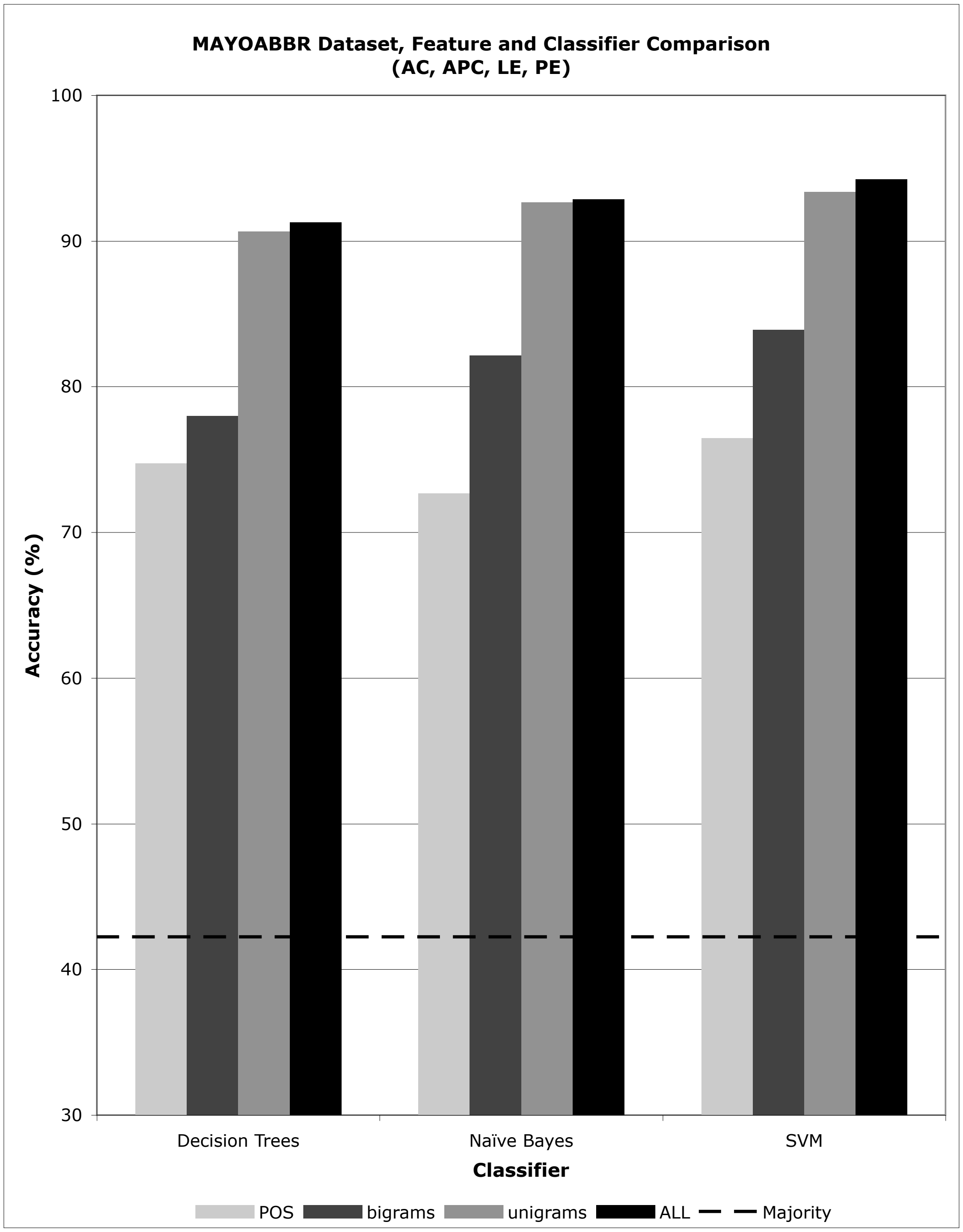

Figure 11: **A comparison of the average performance of three classifiers and four feature representations across four abbreviations from the Mayo Clinic dataset, having majority sense less than 50%. The graph shows the average accuracy for each "classifier-feature representation" combination.**

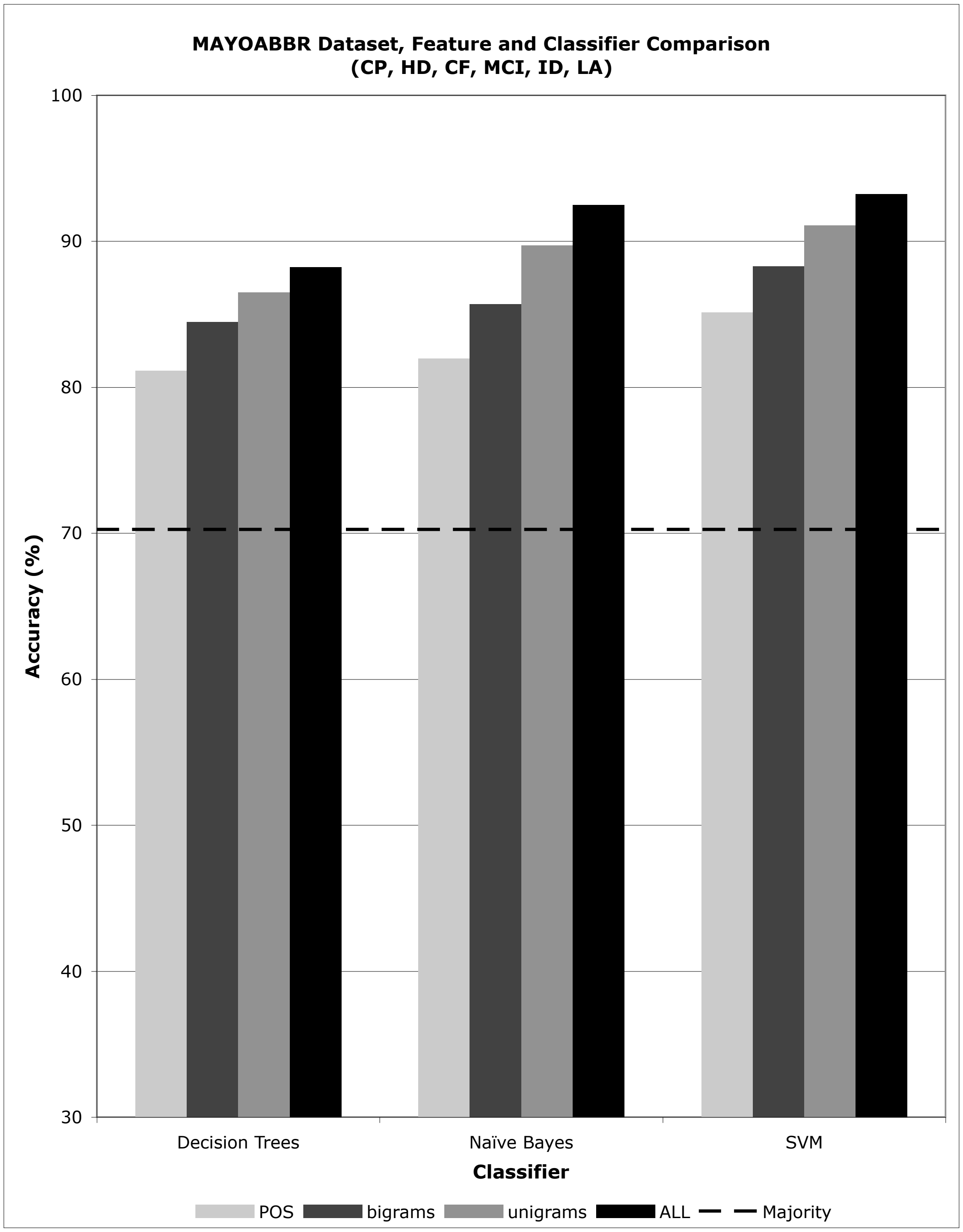

Figure 12: **A comparison of the average performance of three classifiers and four feature representations across six abbreviations from the Mayo Clinic dataset, having majority sense between 50% and 80%. The graph shows the average accuracy for each "classifier-feature representation" combination.**

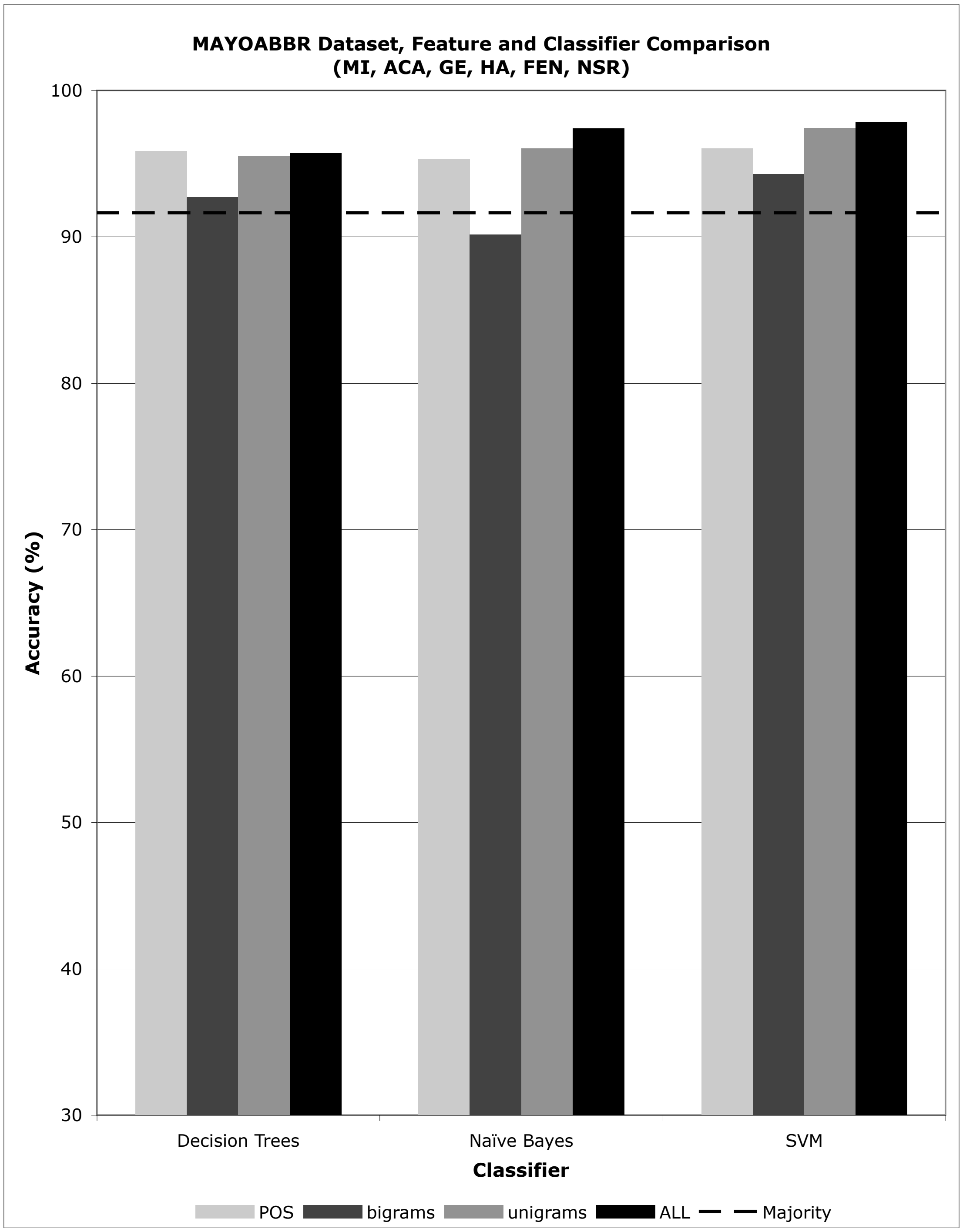

Figure 13: **A comparison of the average performance of three classifiers and four feature representations across six abbreviations from the Mayo Clinic dataset, having majority sense greater than 80%. The graph shows the average accuracy for each "classifier-feature representation" combination.**

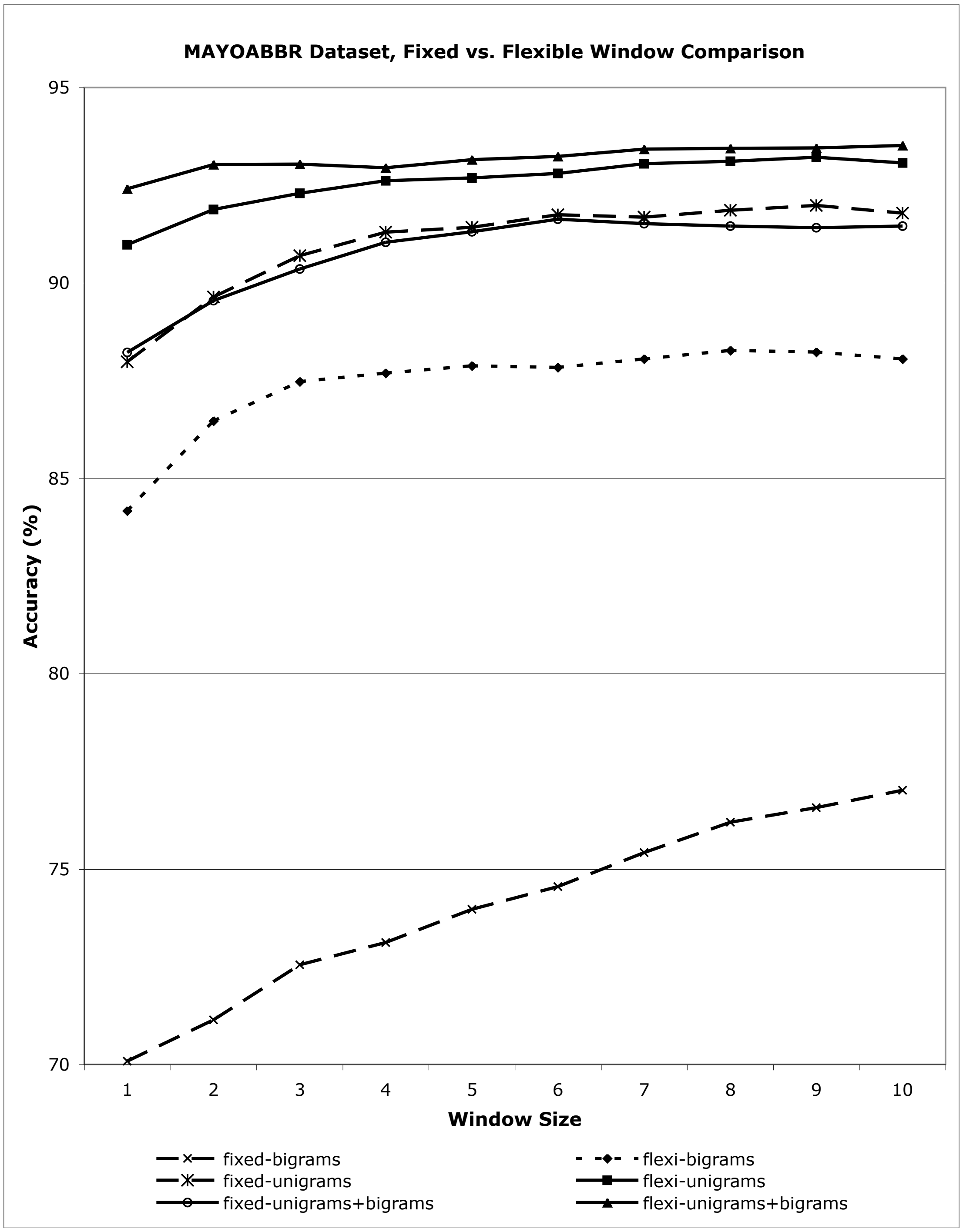

Figure 14: **A comparison of the fixed and flexible window approach for window sizes ranging from one through ten. The graph shows the average accuracy value across a subset of nine abbreviations from the Mayo Clinic dataset and three classifiers, for each fixed and flexible window size used.**

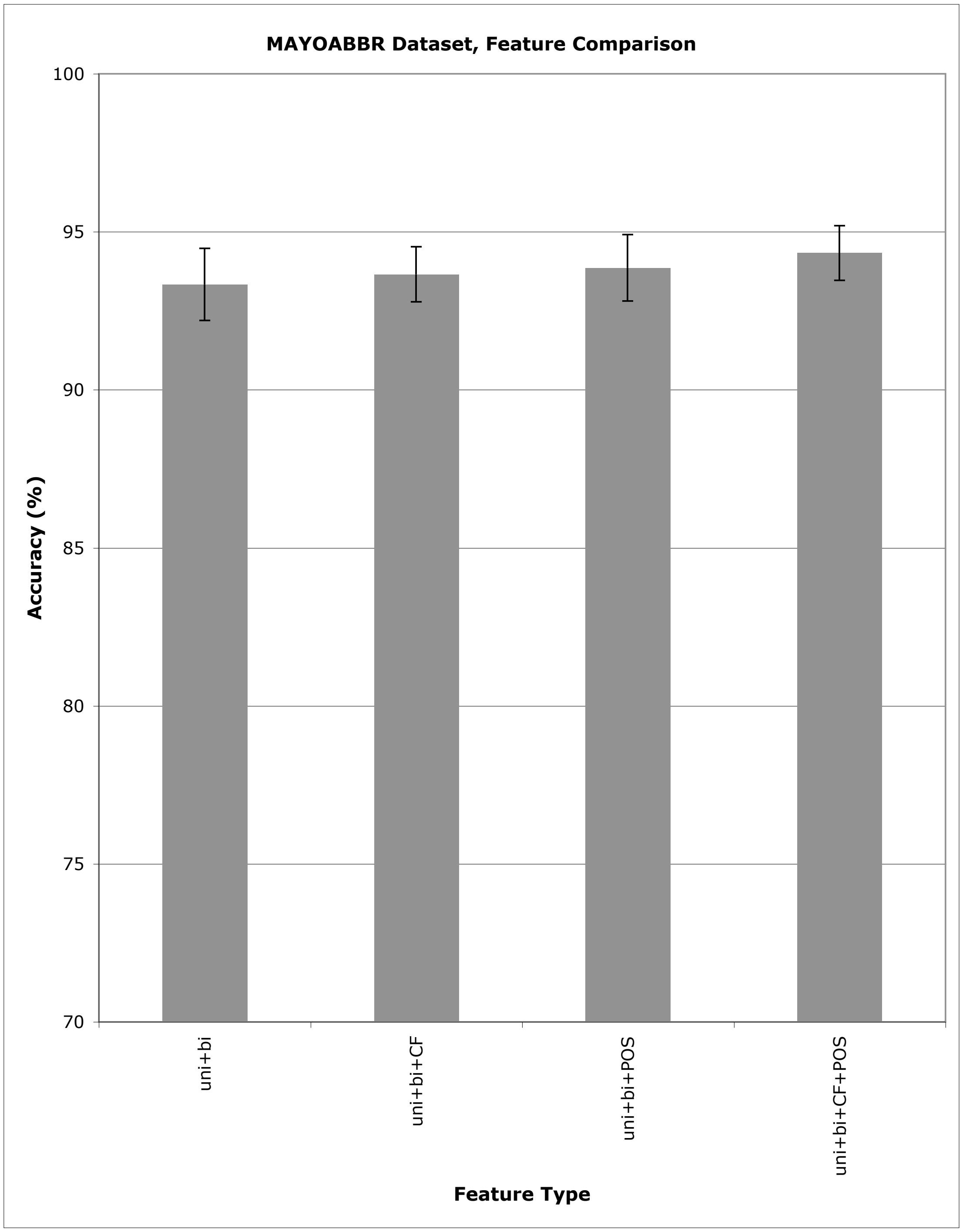

Figure 15: **A comparison of features specific to clinical notes and Part-of- Speech tag features with unigram and bigram features. The graph shows average accuracy value for each feature representation, along with the 95% confidence interval using the error bars.**

For the second set of experiments, Figure 14 shows the comparative performance of the fixed and flexible windows for sizes ranging from one through ten. The accuracy values are averaged across all the nine abbreviations and the three classifiers.

Figure 15 shows the comparative performance of clinical notes specific features (indicated by CF in the graph) and the Part-of-Speech tag features (indicated by POS) in the graph, when they are introduced in addition to the unigram and bigram features. The accuracy values are averaged across all the nine acronyms, three classifiers and the three window sizes of two, nine and ten.

Discussion For the first set of experiments on all of the 16 abbreviations, in general all of the learning algorithms and feature representations improve upon the majority classifier to a significant degree, unless the Majority classifier already attains an average accuracy of approximately 92% (Figure 13). In this case, it means that the data for a particular abbreviation is heavily skewed to one dominant sense, and there are relatively few training examples for the other expansions, making it difficult to outperform the majority classifier. Among the three classifiers, SVMs show the best performance with statistically significant improvement over the naïve Bayes classifier ($p < 0.005$).

Among the most striking results is the relatively high accuracy obtained by simply using Part-of-Speech tag features. While this does not rival the highest accuracy, this does consistently result in accuracy that is significantly greater than that of the majority classifier, which is the standard baseline for supervised learning performance. This suggests that using relatively simple syntactic information in addition to the lexical features can be an important factor in improving supervised abbreviation expansion results. Mohammad and Pedersen [40] have employed this idea for the task of WSD in general English domain.

In the result graphs we also observe that unigrams perform at nearly peak accuracy for all abbreviations. However the combination of unigrams, bigrams and part of speech features results in a slightly higher accuracy that is statistically significant for this dataset with $p < 0.005$. The combined feature set is dominated by unigram performance, with relatively small improvements due to the inclusion of the part of speech and bigram features.

When used on their own, bigrams underperformed unigrams and combined features to a significant degree. We believe that this might have been due to our reliance on a frequency cutoff to identify significant bigrams, which can also be identified using statistical tests of association such as the Log Likelihood ratio, which might make these features more informative and less noisy.

For the second set of experiments on the subset of nine abbreviations, Figure 14 shows a clearly significant improvement in the accuracy of bigram features with a flexible window ($p < 0.001$). The improvement is also significant for unigram features and the combined unigram and bigram feature set, with $p < 0.001$ for both. This demonstrates the general utility of using a flexible window over a fixed window size.

Finally, the evaluation of features specific to clinical notes does not show significant improvement over the basic unigram and bigram features ($p = 0.495$) and neither does adding Part-of-Speech tag features show any significant improvement by itself ($p = 0.174$). However, combining these two features in addition to the unigrams and bigrams does show a significant improvement ($p < 0.05$).

4.2.4 Baseline Experiments Summary

Overall, we found that using the wider abstract context for the NLM dataset yields better accuracy than using the smaller sentence context. Unigram features with a frequency cutoff of five was the best feature representation and SVMs and the naïve Bayes classifier were the best classifiers. We employed the best feature representation from our NLM dataset experiments on the MEDLINE abbreviation dataset and have obtained a competitive baseline for our kernel experiments on this dataset. For our feature engineering experiments on the Mayo Clinic abbreviation dataset, we found that a combination of unigram and bigram features in a flexible window around the abbreviation, with Part-of-Speech features and features specific to clinical notes such as gender code, department code and section identifier obtains the best accuracy.

4.3 Kernel Experiments

In this section we describe the experiments that we have performed on the NLM WSD dataset and the MEDLINE abbreviation dataset using our semantic kernels. The goal of these experiments is to compare the accuracy obtained by our semantic kernels with the accuracy of the five classifiers that we have used in our baseline experiments.

4.3.1 NLM Dataset

Data For our kernel experiments, we have chosen a subset of the NLM dataset consisting of the eleven words *adjustment, blood_pressure, evaluation, growth, immunosuppression, japanese, man, mosaic, nutrition, radiation* and *white*. These words were selected based on the criteria that the majority sense of the selected words should not exceed 75% and the *None* sense of the selected words should not exceed 25%.

Methodology For the labeled examples in our kernel methods, we have chosen a feature set of 20 feature representations from the 36 feature representations described in the baseline experiments on the NLM dataset. The 16 feature configurations that were rejected were those of bigram features with Log Likelihood score cutoff of 6.635 along with all of its window size and frequency cutoff combinations. This was done because the Log Likelihood cutoff of 6.635 was too strict and rejected many useful bigram features, thus degrading the performance.

For every supervised feature configuration, we experimented with four types of kernels – LSA kernels with unigram features, LSA kernels with bigram features, LSA kernels with co-occurrence features and finally Association kernels with bigram features.

For each of the kernel types, we experimented with the size of the context around the ambiguous word to use for learning from unlabeled data (called as the *training context*) and the size of the context to use from the labeled data to evaluate context similarity (called as the *test context*). We used training context sizes of 100, 50 and 25 words around the ambiguous word and test context sizes of 10, 5 and 2 words around the ambiguous word, resulting in nine combinations. Additionally we also trained and tested on the entire contexts, adding one more configuration to the nine combinations.

The unigram features for LSA kernels were selected using frequency cutoff values of 2, 5, 10 and 15. The bigram and co-occurrence features for LSA kernels were selected using frequency cutoff values of two and five. Finally, the bigram features for the Association kernels were selected using frequency cutoff values of 2, 5, 10 and 15. Thus, overall we show results for 12 kernels (four unigram LSA kernels, two bigram LSA kernels, two co-occurrence LSA kernels and four bigram Association kernels).

Results Each of the 12 kernels above has been used with all the 20 feature representations of the labeled data. However, ere we present the result graphs created using only the best supervised feature representation for the abstract context baseline results, the unigram features with frequency cutoff of five. The intention is to evaluate how much better the kernel methods can perform beyond what can be achieved using the best feature configuration for off-the-shelf algorithms.

Figures 16 through 27 present the average accuracy obtained by our 12 kernels across all the eleven words, when using unigrams with frequency cutoff equals five as the feature set from the labeled data.

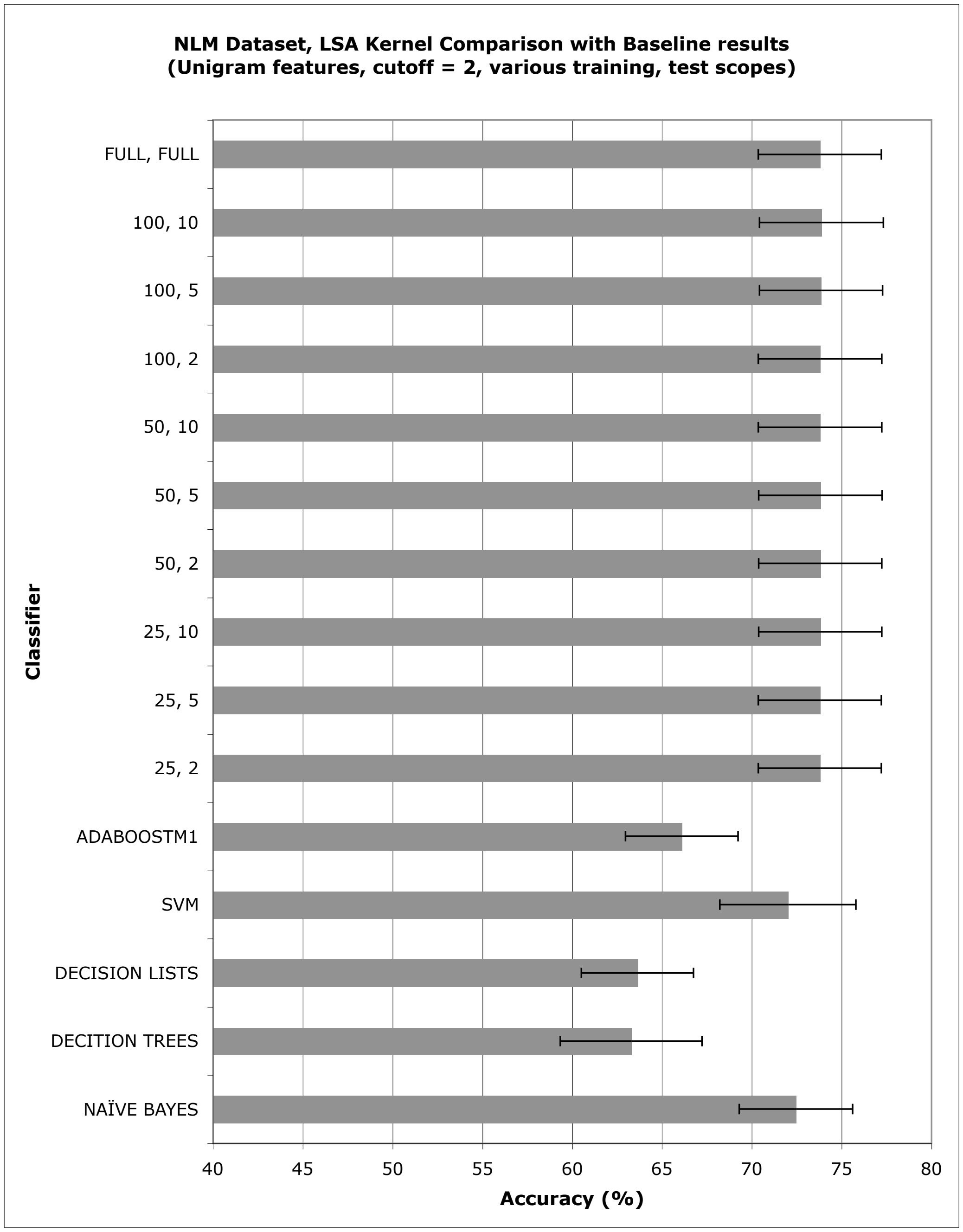

Figure 16: **A comparison of the unigram based LSA kernels (frequency cutoff of two) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each unigram LSA kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

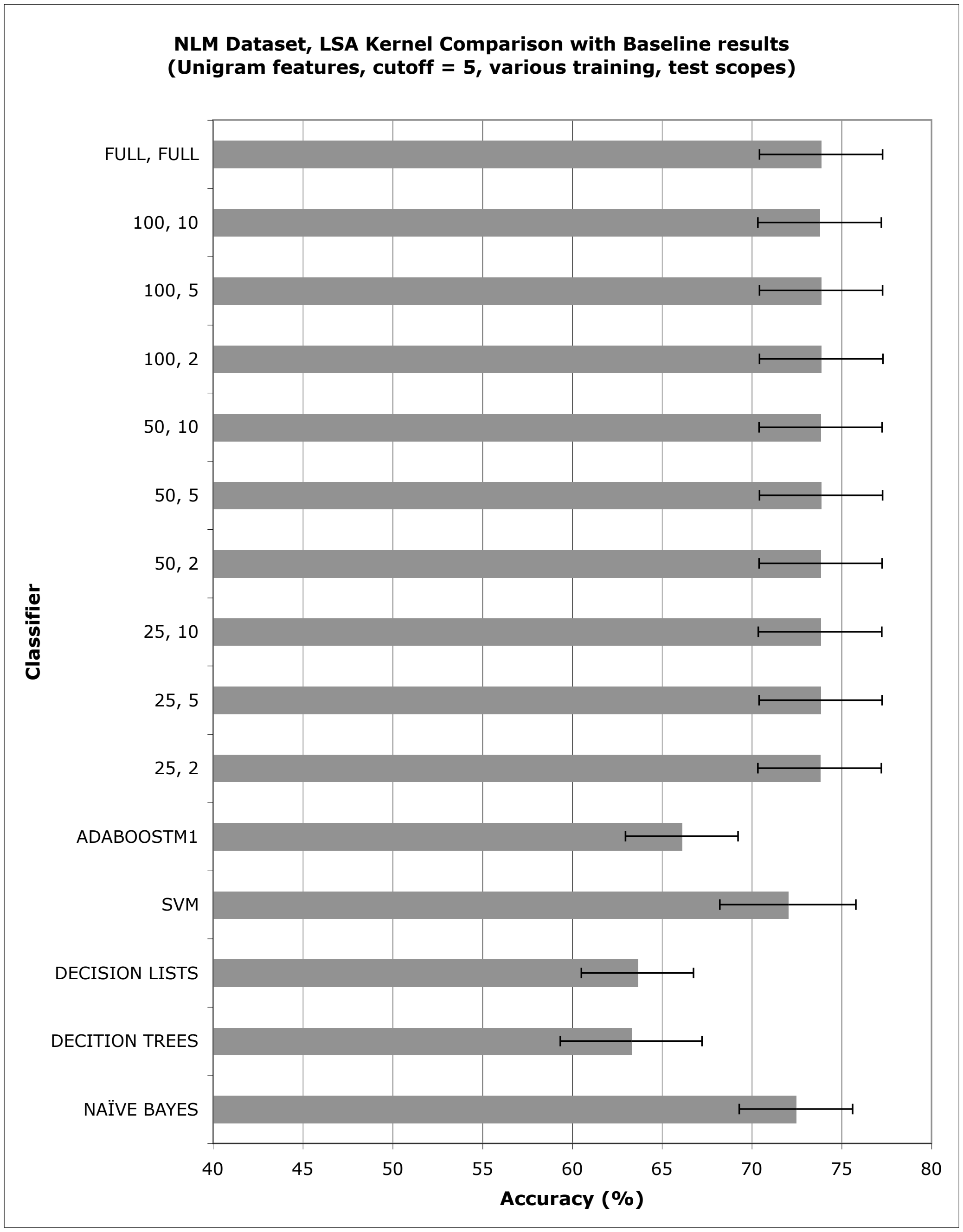

Figure 17: **A comparison of the unigram based LSA kernels (frequency cutoff of five) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each unigram LSA kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

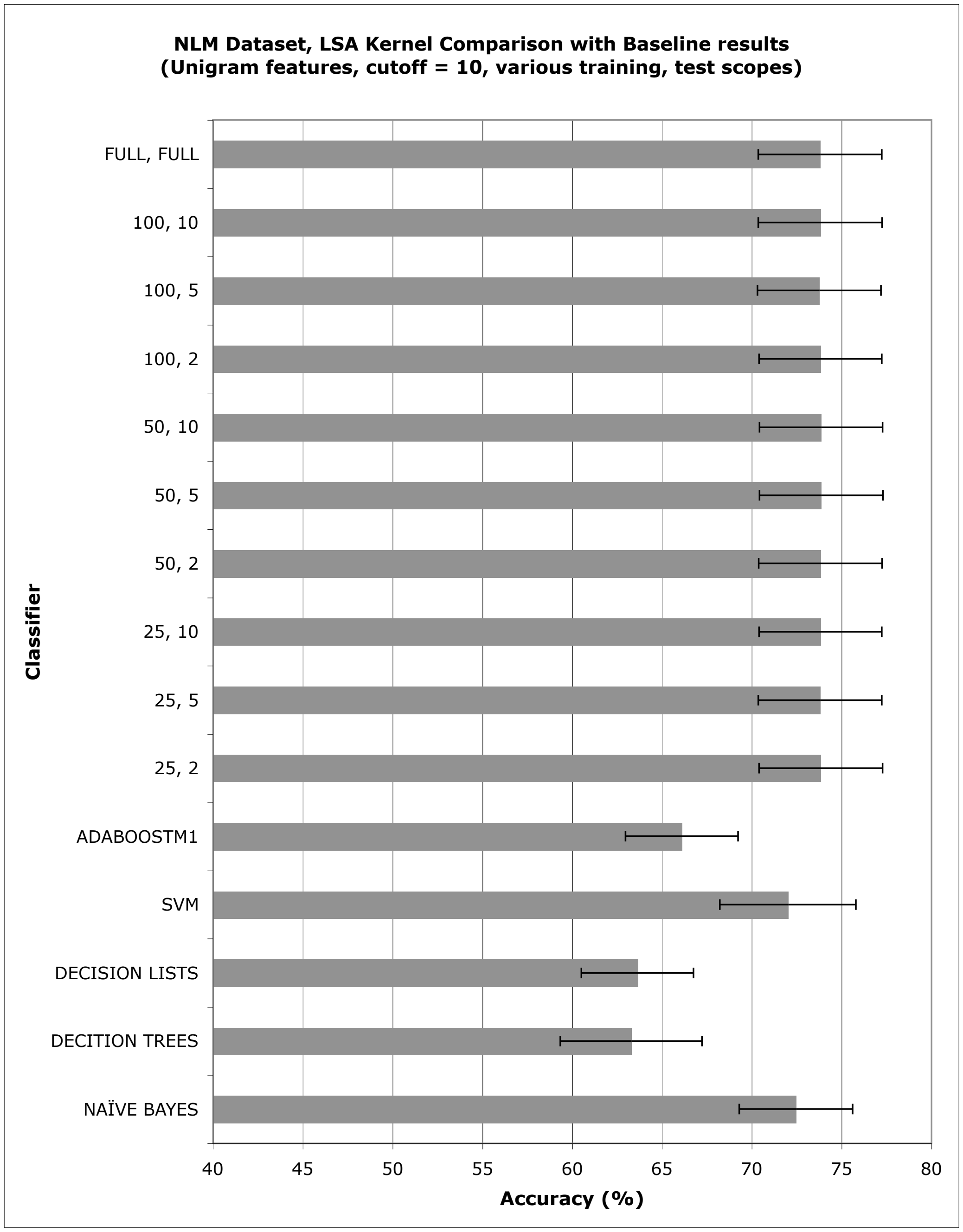

Figure 18: **A comparison of the unigram based LSA kernels (frequency cutoff of ten) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each unigram LSA kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

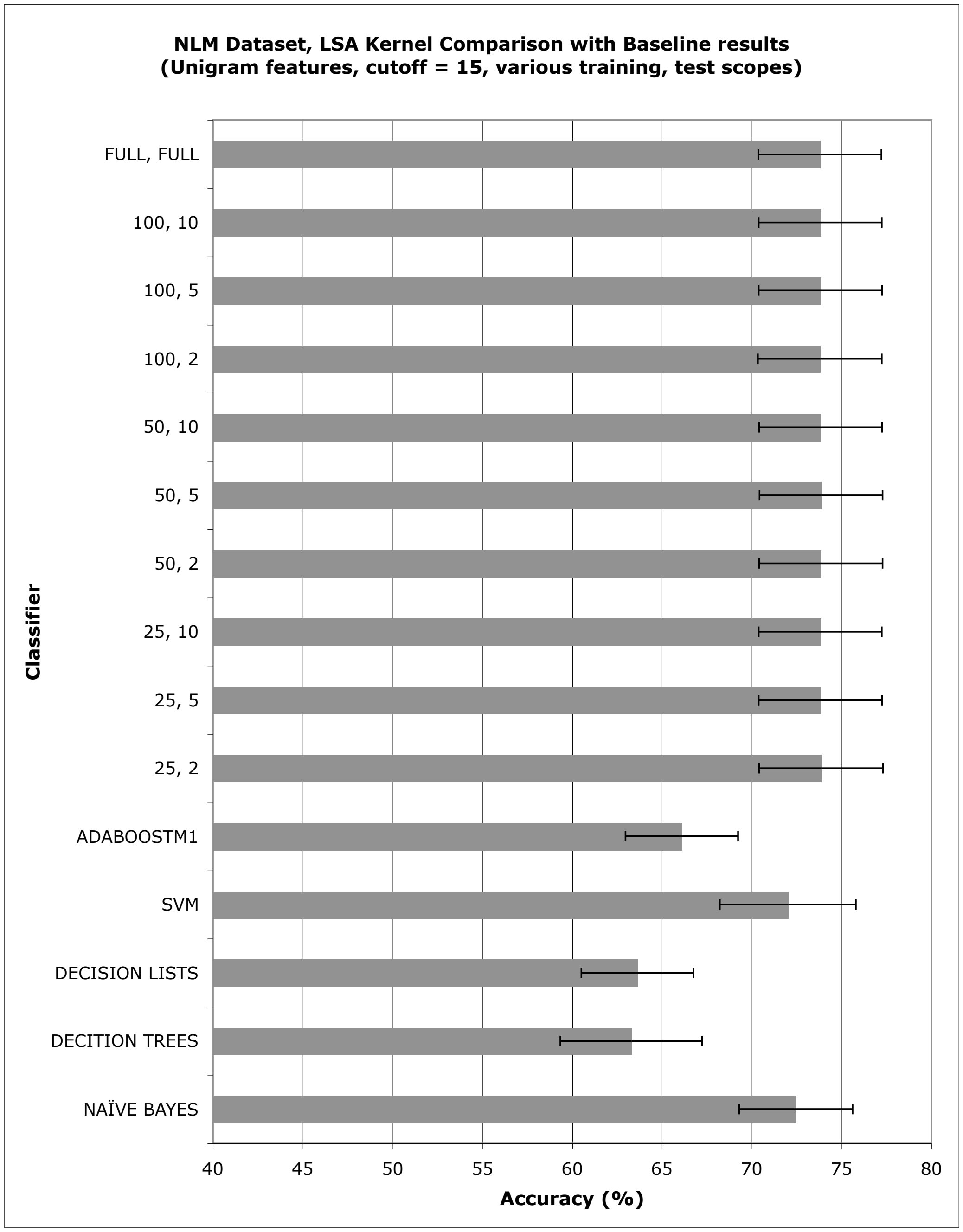

Figure 19: **A comparison of the unigram based LSA kernels (frequency cutoff of 15) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each unigram LSA kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

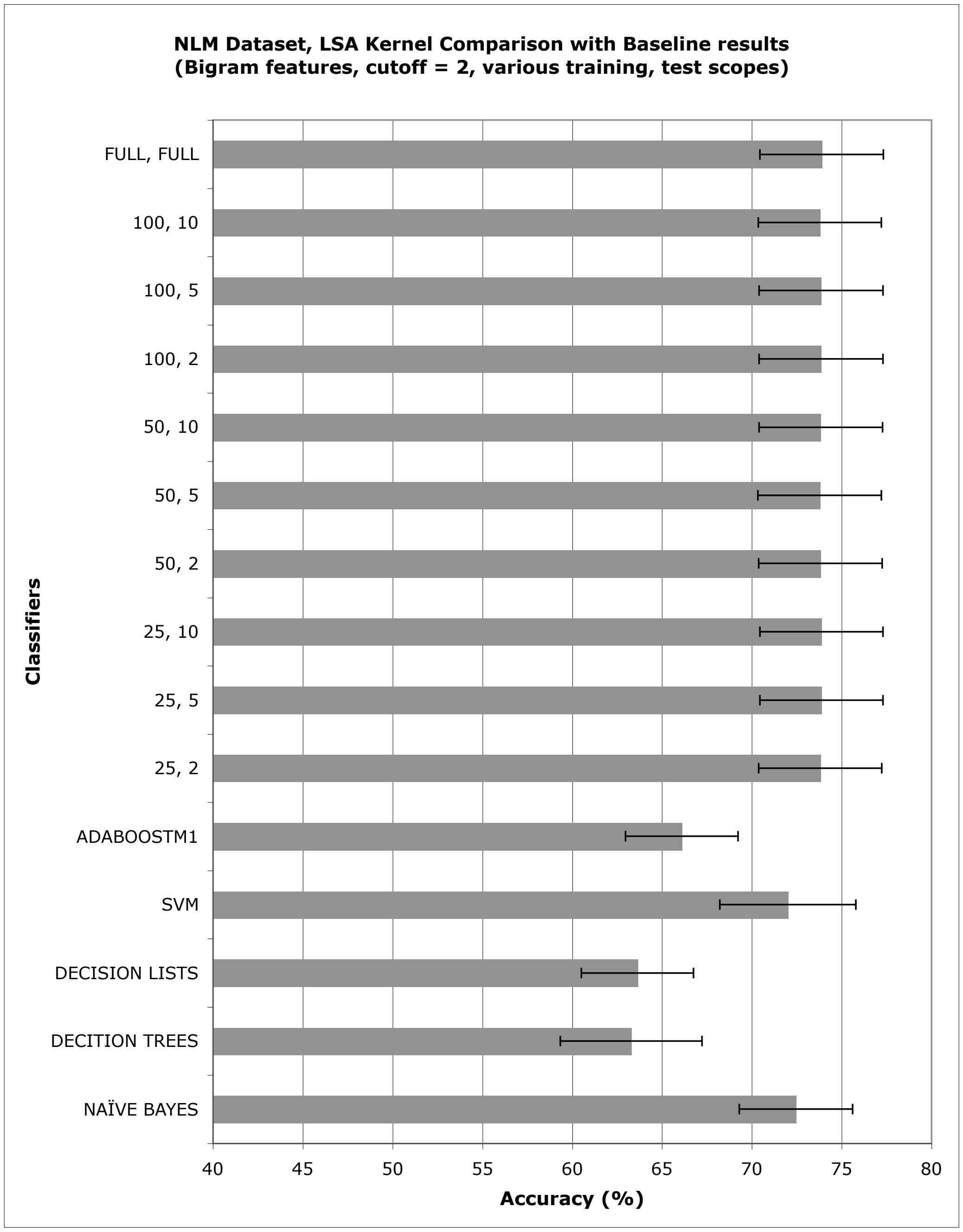

Figure 20: **A comparison of the bigram based LSA kernels (frequency cutoff of two) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each bigram LSA kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

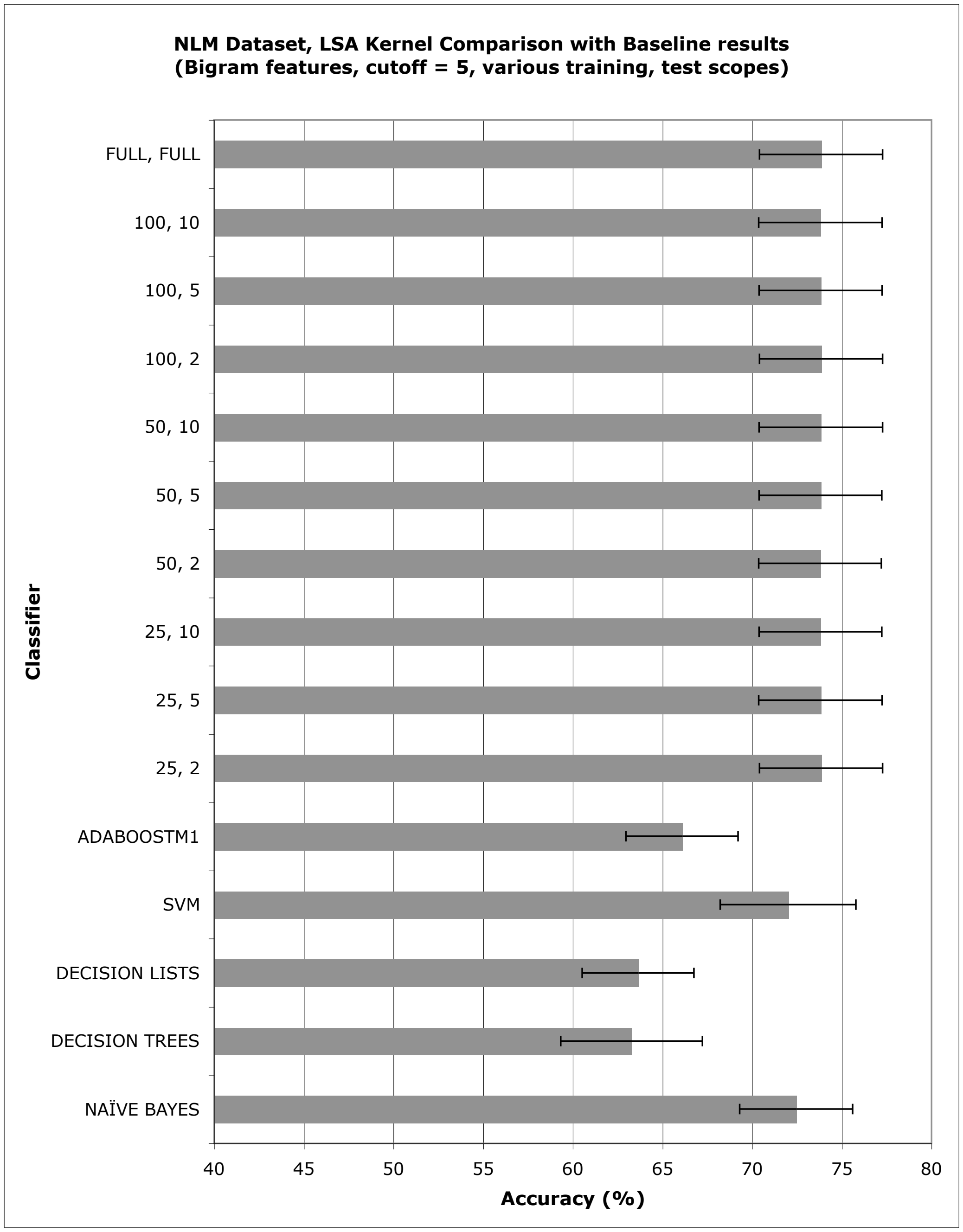

Figure 21: **A comparison of the bigram based LSA kernels (frequency cutoff of five) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each bigram LSA kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

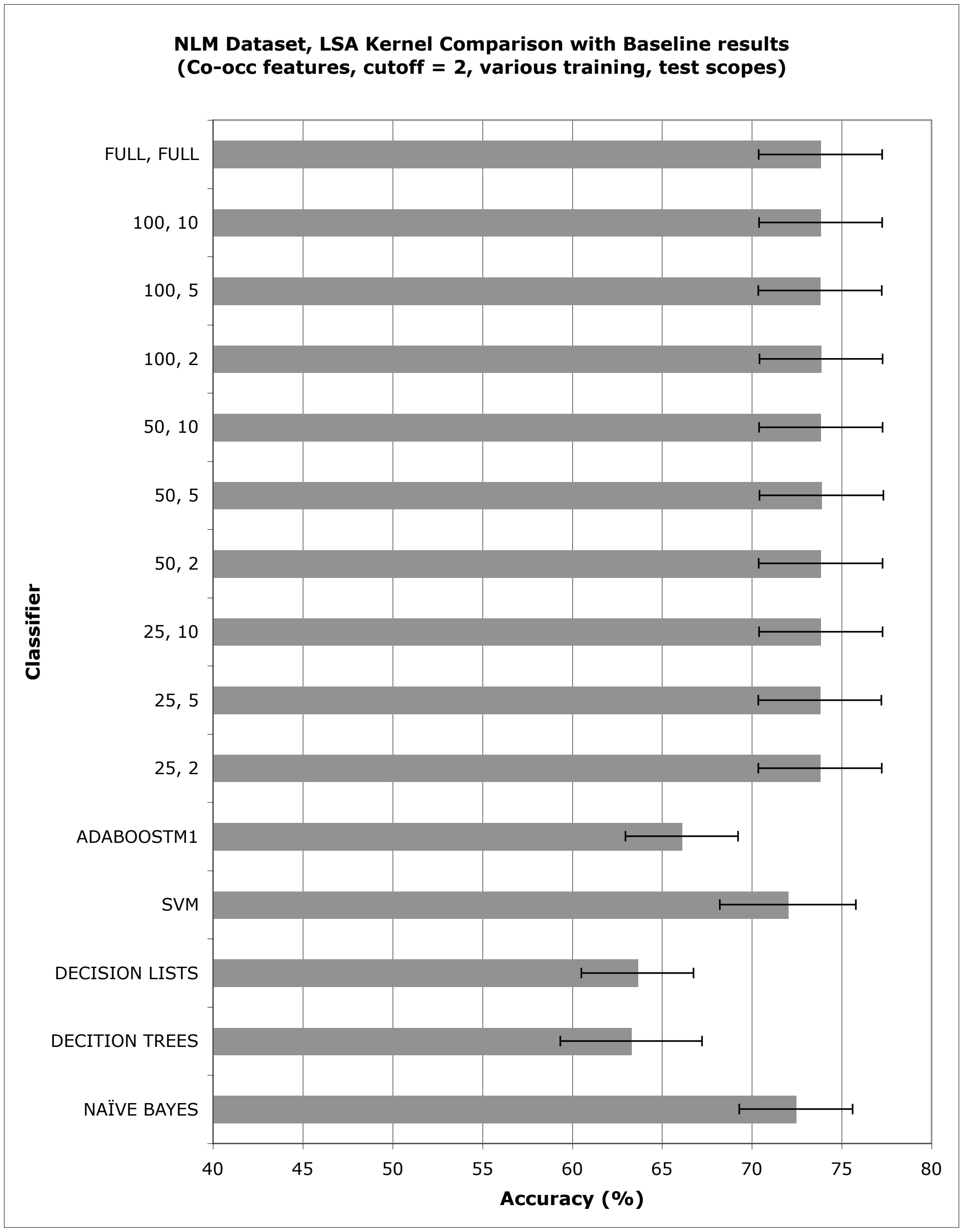

Figure 22: **A comparison of the co-occurrence based LSA kernels (frequency cutoff of two) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each co-occurrence LSA kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

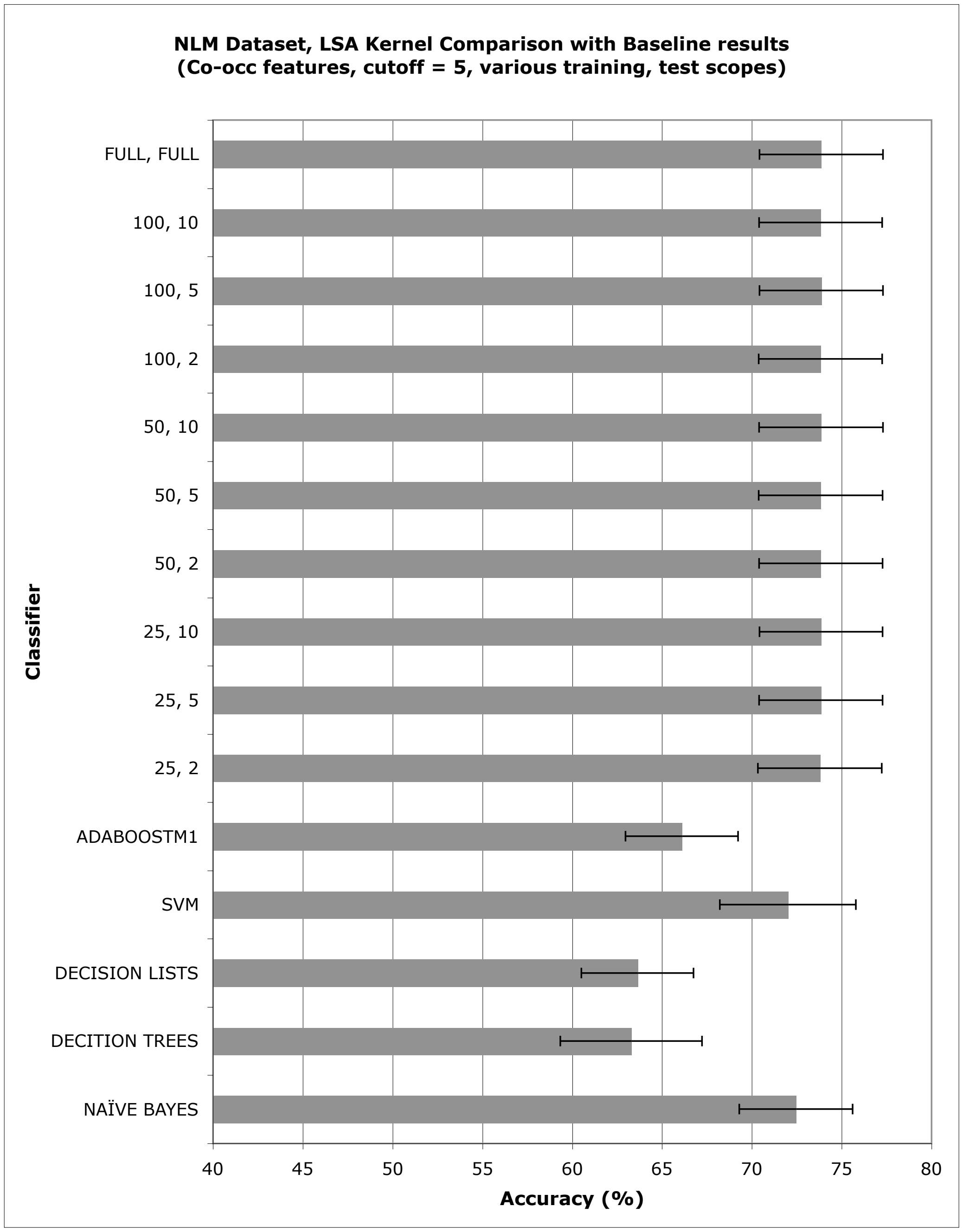

Figure 23: **A comparison of the co-occurrence based LSA kernels (frequency cutoff of five) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each co-occurrence LSA kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

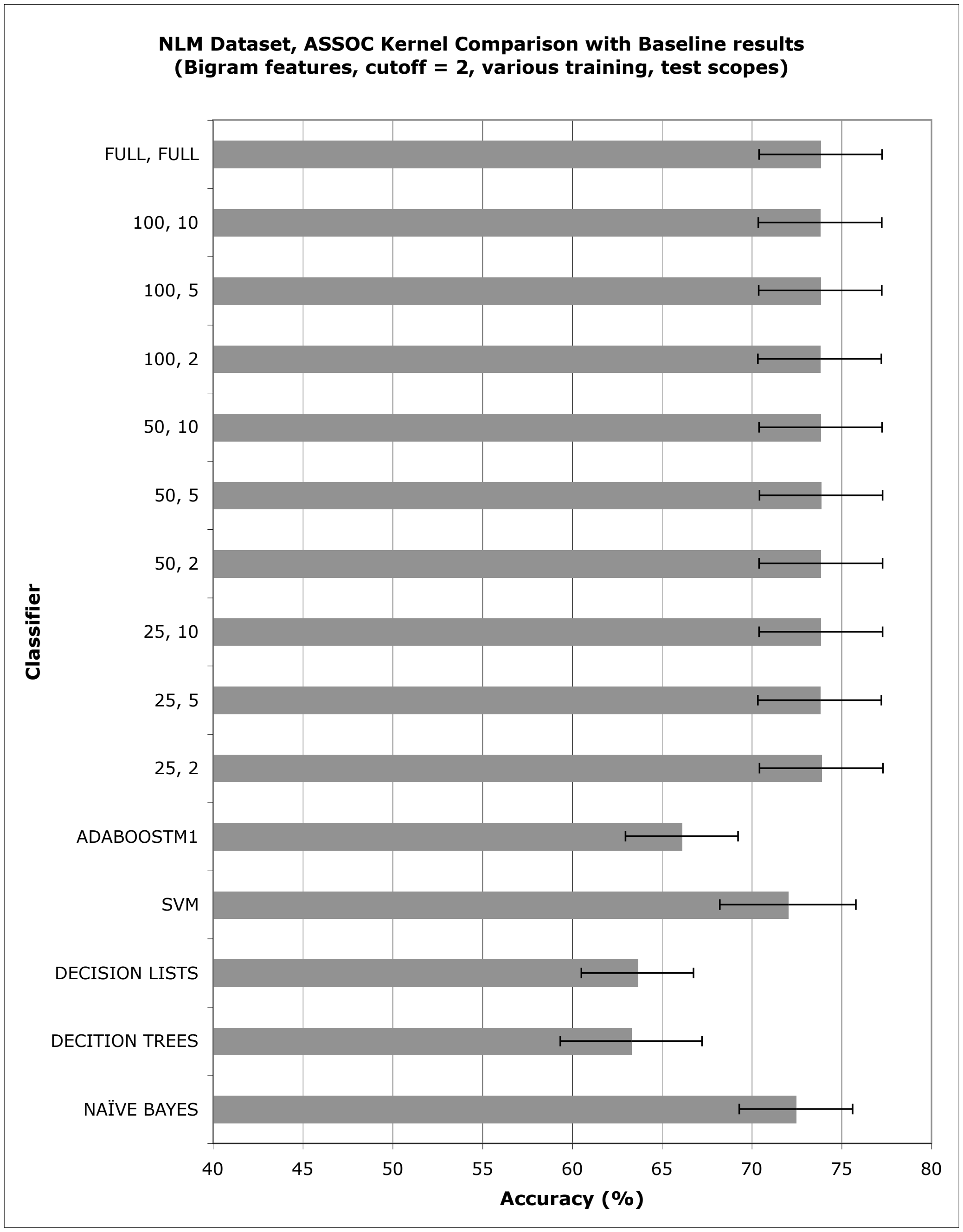

Figure 24: **A comparison of the bigram based association kernels (frequency cutoff of two) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each bigram association kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

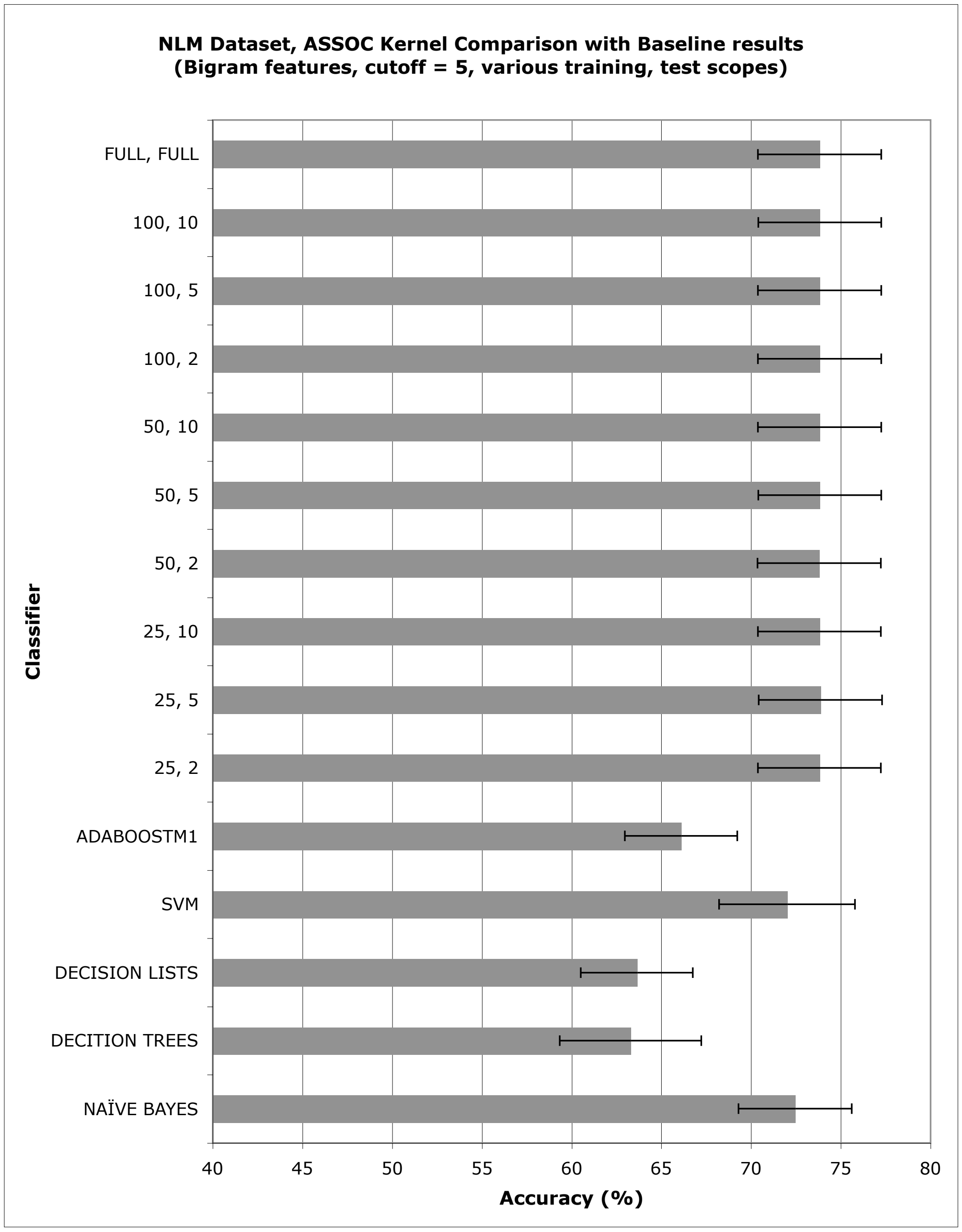

Figure 25: **A comparison of the bigram based association kernels (frequency cutoff of five) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each bigram association kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

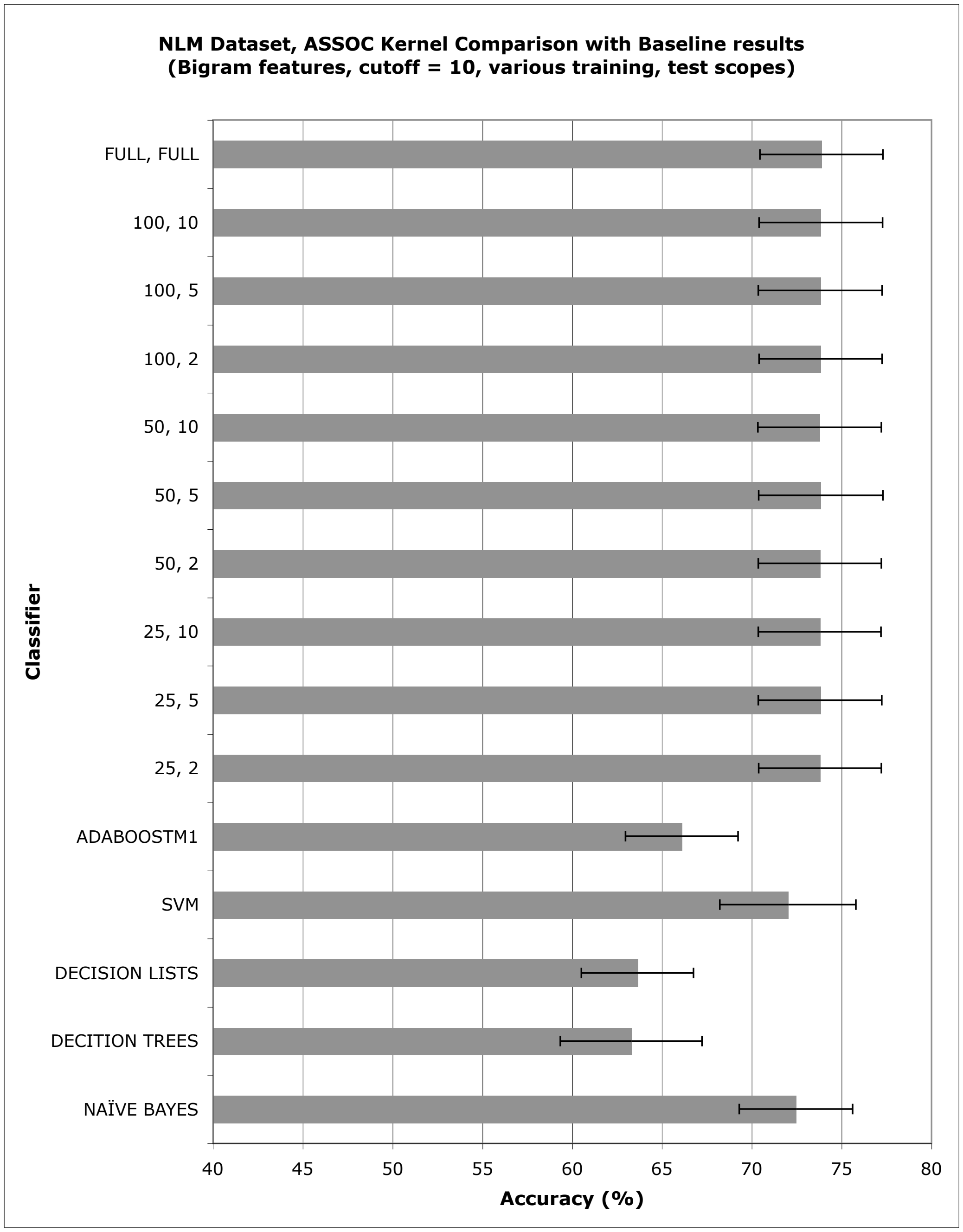

Figure 26: **A comparison of the bigram based association kernels (frequency cutoff of ten) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each bigram association kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

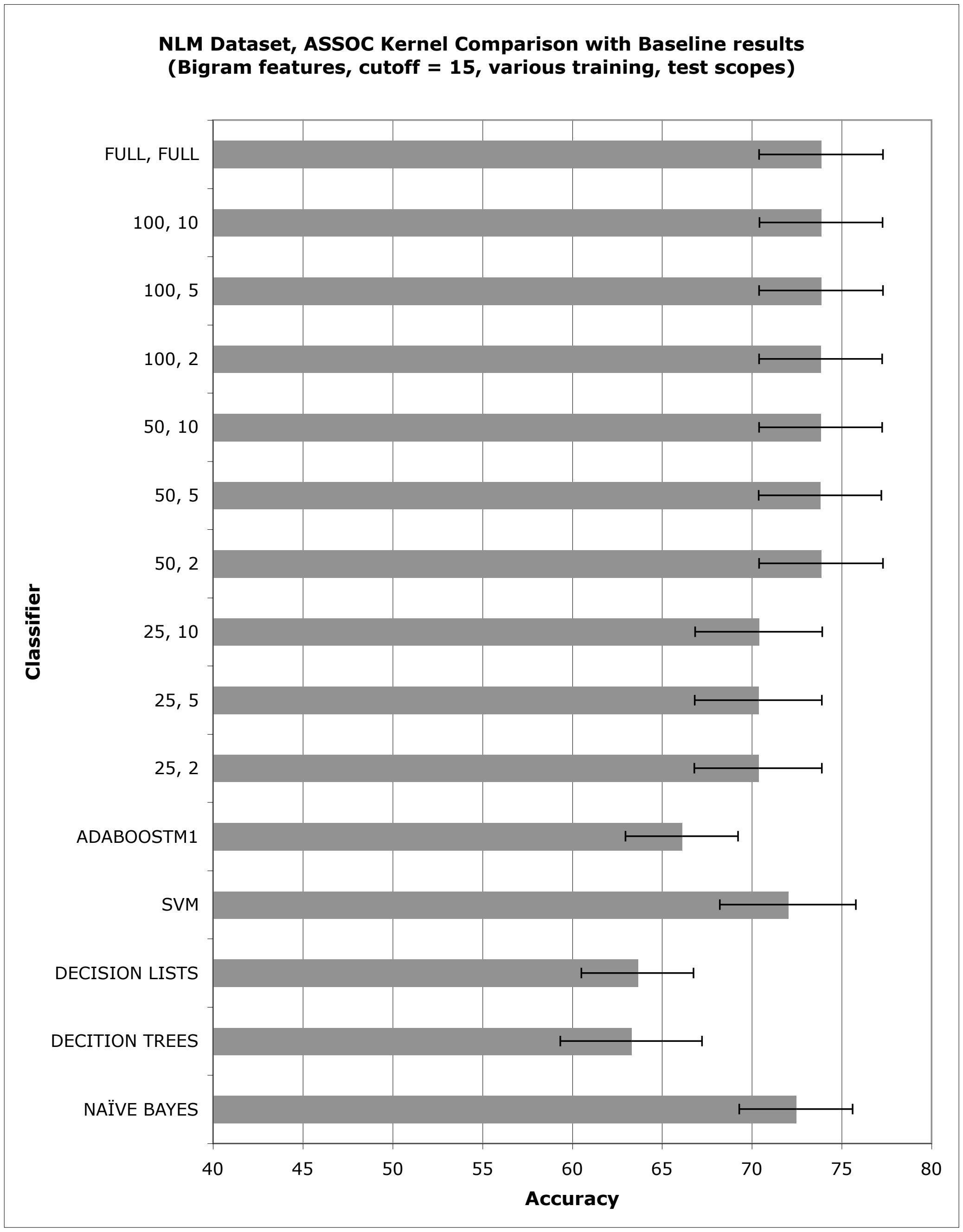

Figure 27: **A comparison of the bigram based association kernels (frequency cutoff of 15) with the baseline WEKA classifiers. The graph shows average accuracy value across 11 words in the NLM dataset, for each bigram association kernel and WEKA classifier, along with the 95% confidence interval using the error bars. Comma separated numbers on the classifier axis indicate the size of the training and test contexts used.**

Figure 28 shows the comparison of accuracy of the best unigram LSA kernel (which uses unigram features from unlabeled data with a frequency cutoff of two and training scope of 100 and test scope of ten, shown in the graph as "*LSA, UNI, cutoff = 2, 100, 10*") and the best bigram Association kernel (which uses bigram features from the unlabeled data with a frequency cutoff of ten and uses the entire contexts for the training as well as test scope, shown in the graph as "*ASSOC, BI, cutoff = 10"*), for each of the 11 words that we have used for our kernel experiments. We compare them with the performance of the baseline linear kernel SVM learner (shown in the graph as SVM).

Discussion The best baseline performers for the subset of eleven words are again the SVMs and the naïve Bayes classifier. In all the results graphs, if we compare the accuracies of the best kernel combination with each of the naïve Bayes classifier and the baseline SVM, then we see a slight improvement. However these improvements in general are not statistically significant. The improvements come close to being statistically significant in almost all cases (p values close to 0.1), but do not meet the criterion. One thing to note however is that our baseline estimates have been derived only from one run of 10-fold cross-validation, however our kernel results are derived from averaging 10 runs of 10-fold cross-validation. So there is a chance that the baseline estimates are in fact lower, making the improvement significant. However, this is speculative until established through experiments.

We can see from Figure 28 that both of our best semantic kernels show significant improvement in accuracy for the words *blood_pressure evaluation, immunosuppression, mosaic* and nutrition ($p < 0.05$ for both the unigram LSA kernel and for the bigram Association kernel). This does not correlate to the number of unlabeled instances that we have for each of the words above (*blood_pressure* has 4,767, *evaluation* has 11,523, *immunosuppression* has 954, *mosaic* has *402* and *nutrition* has 2637). We also note that performance has in fact decreased due to semantic kernels for the word *man* which has as many as 3,852 unlabeled instances (significantly greater than those for *immunosuppression* or *mosaic*). We therefore conclude that the accuracy of kernel methods based on unlabeled text depends highly on the quality of the unlabeled data rather than quantity. This is an intuitive conclusion since the noise in the unlabeled data will determine the quality of semantic relationships learned from it, which will directly affect performance. However, what we think is crucial is that even a small amount of high quality unlabeled data can improve performance of SVM

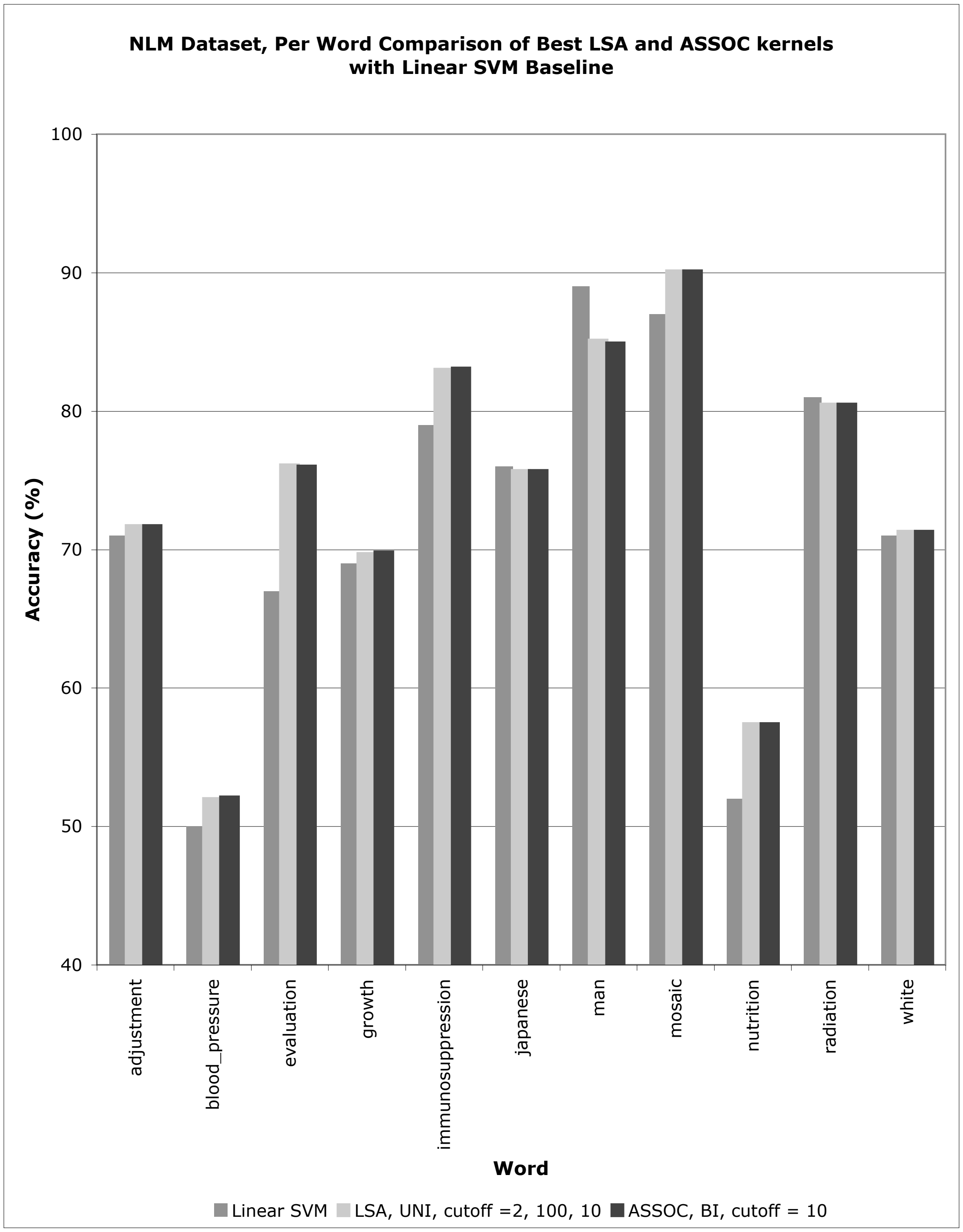

Figure 28: **A comparison of the best unigram based LSA kernel, the best bigram based Association kernel and the baseline WEKA SVM learner, for each word in the NLM dataset. The graph shows the accuracy of each kernel and the WEKA SVM learner for each of the 11 words.**

learners significantly, as shown in case of the words *immunosuppression* and *mosaic*.

Another trend that we note is that among the five words for which our semantic kernels have significant performance improvements, the sense distribution is balanced for four words, except for the word *nutrition*. The other four words have two predominant senses in almost equal proportion. We believe this is might indicate that the unlabeled instances for these words also had a similar sense distribution as the labeled instances, which helped our kernel methods to learn the distinction between the predominant senses better.

4.3.2 MEDLINE Abbreviation Dataset

We now present semantic kernel results on the MEDLINE abbreviation dataset. The goal in these experiments is to evaluate using the MEDLINE abbreviation dataset, the generality of the best feature configuration and the best kernel methods that we have found from the NLM dataset. Note that both the NLM dataset and the MEDLINE abbreviation dataset are derived from the MEDLINE collection of abstracts and therefore we hypothesize that the best configuration from our NLM dataset experiments will translate well on this dataset as well.

Data We use the same set of ten abbreviations as in the baseline experiments. As mentioned before, we do not have an unlabeled set of instances for the MEDLINE abbreviation dataset. We therefore randomly split the instances of an abbreviation into two sets, one containing 75% of the instances and the other containing the remaining 25% of the instances. These sets are identical across our baseline and kernel experiments. We utilize the set containing 75% of instances of each abbreviation as unlabeled instances and the set containing the remaining 25% of the instances as labeled data.

Methodology For the kernel experiments on the MEDLINE abbreviation dataset, we have only chosen the best feature configurations for labeled data, that is unigrams with frequency cutoff of five, and the best unigram LSA kernel and the best bigram Association kernel, based on the average accuracy value. The kernels chosen were the unigram LSA kernel with frequency cutoff of two, training scope of 100 and test scope of 10 and the bigram Association kernel with frequency cutoff value of ten.

Results Figure 29 shows the comparison of the accuracy of our unigram LSA kernel and the bigram Association kernel with the baseline classifier accuracies shown earlier in Figure 10.

Figure 30 shows the comparison of the two kernels with the default linear kernel SVM, for each of the abbreviations in our dataset.

Discussion Both the unigram LSA kernel and the bigram Association kernel show significant improvement over the baseline results and improve upon the best baseline SVM learner ($p < 0.005$). Both the kernels have the same average accuracy over the abbreviation dataset. This improvement in accuracy verifies our hypothesis that the best configuration on the NLM dataset should also show good performance on the MEDLINE abbreviation dataset which is similar to the NLM dataset in some respects, since both the dataset originate from the MEDLINE abstracts.

4.4 Summary

The goal of our experiments is to initially establish a competitive baseline for our semantic kernels, using various off-the-shelf classifiers and the best possible feature representation, and then compare the accuracy obtained using our semantic kernels with this baseline. We also focus on the feature engineering aspect of the abbreviation disambiguation task in the domain of clinical notes.

In our baseline experiments we have found that in general a wider context helps in improving accuracy on the task of WSD. The best feature representation was unigram features with a frequency cutoff of five. The best classifiers among the five classifiers that we evaluated for the baseline were the SVMs and the na”ive Bayes classifier.

Our semantic kernel experiments were performed by using the best feature representation found in our baseline experiments. We found that among the two classes of kernels that we evaluated, unigram LSA kernels and bigram Association kernels both improve the accuracy of WSD and abbreviation expansion. The improvement on NLM dataset is significant for words that have a balanced sense distribution. The improvement is not correlated to the number of unlabeled instances

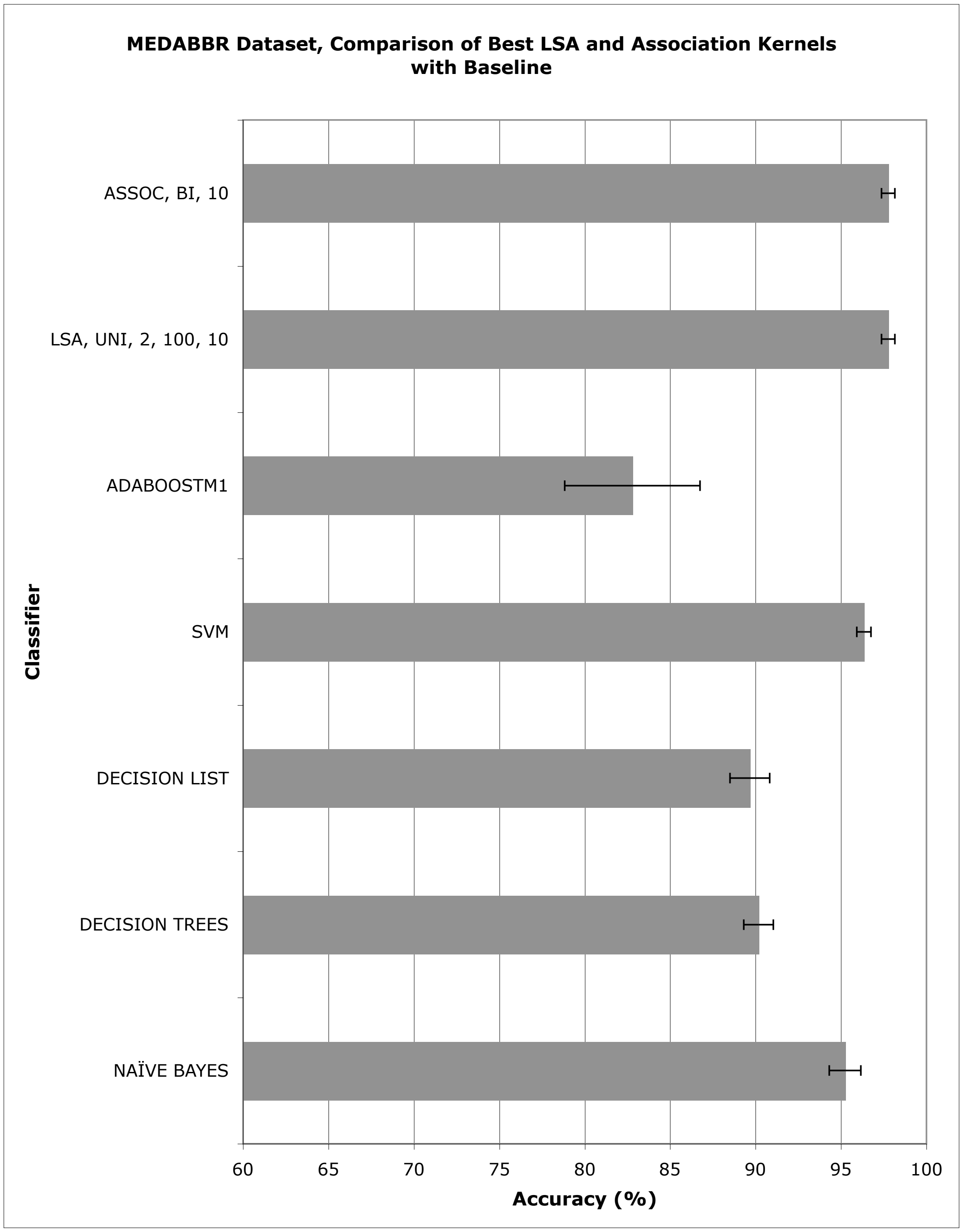

Figure 29: **A comparison of the best unigram LSA kernel and the best bigram association kernel with baseline WEKA classifiers on the MEDLINE abbreviation dataset. The graph shows the average accuracy of each kernel and the WEKA classifiers across ten abbreviations, along with the confidence interval using the error bars.**

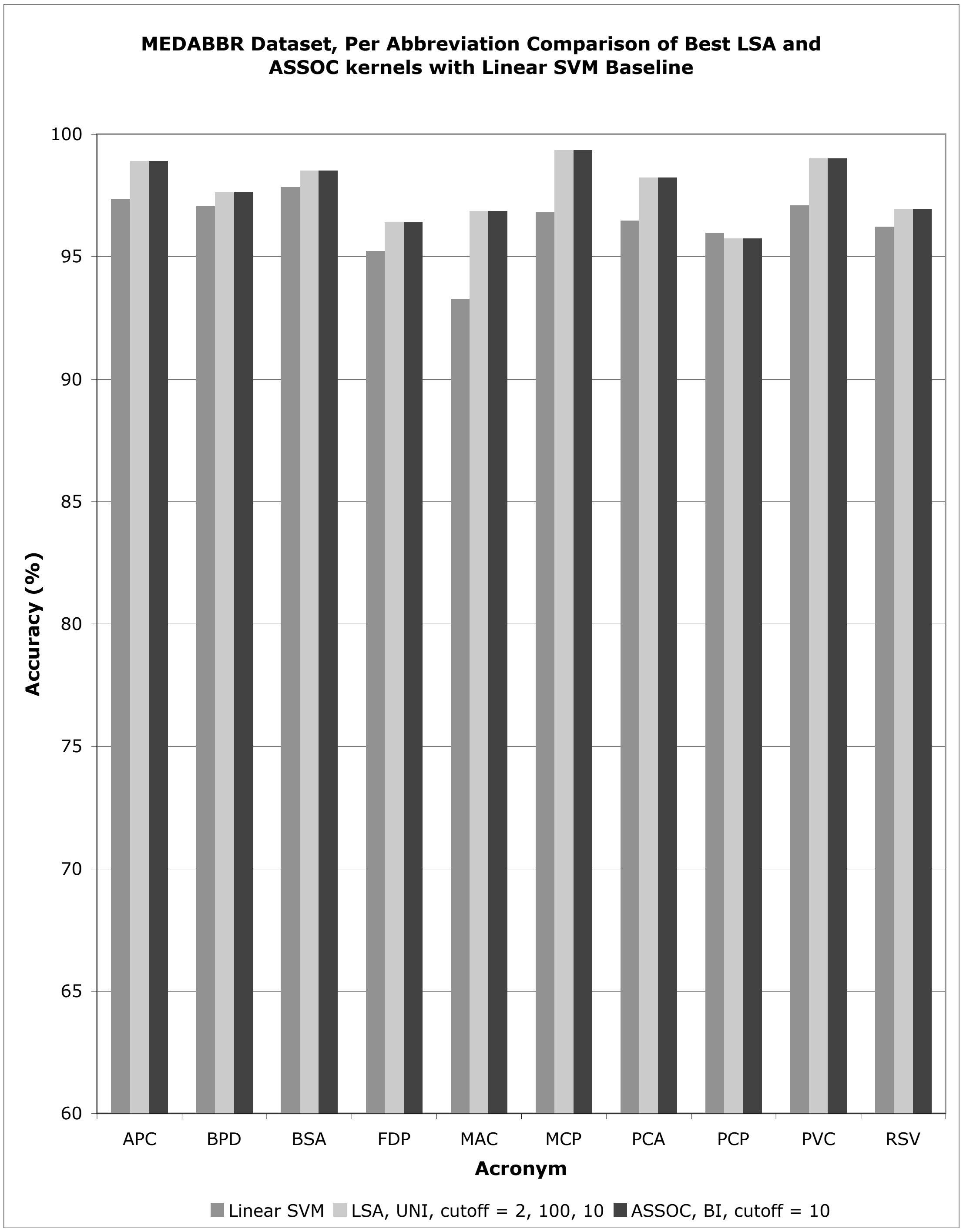

Figure 30: **A comparison of the best unigram LSA kernel, the best bigram association kernel and the baseline WEKA SVM learner, for each abbreviation in the MEDLINE abbreviation dataset. The graph shows the accuracy of each kernel and the WEKA SVM learner for each of the ten abbreviations.**

available for a word.

In our feature engineering experiments on the Mayo Clinic abbreviation dataset we found that unigram and bigram features in a flexible window, along with Part-of-Speech tag features and features specific to clinical notes such as gender code, department code and section identifier gave the best accuracy. In particular, the flexible window approach showed significant improvement over the commonly used fixed window approach, for all feature representations.

5 Related Work

In this chapter we give an overview of previous work in addition to what has been detailed in the Background section.

We divide this chapter into three sub-sections. First we note related work on semantic kernels. Then we describe previous work that uses the NLM WSD collection that we have used for most of our experiments. Finally we include related work with respect to WSD in the general English domain that uses Support Vector Machines.

5.1 Previous Work on Semantic Kernels for WSD

Gliozzo et al., [19] have presented domain kernels for word sense disambiguation. Their key notion is to make use of domain knowledge while performing word sense disambiguation. An example they discuss is the ambiguity of the word *virus*. A *virus* can mean *"a malicious computer program"* in the domain of computers or *"an infectious agent which spreads diseases"* if we switch to the domain of medicine. Gliozzo et al. propose a domain matrix (with words along the rows and domains along the columns) that consists of soft clusters of words in different domains. A word can belong to multiple domains with different probabilities – thus representing word ambiguity, whereas a domain can contain multiple words – thus representing its variability. They make use of the fully unsupervised approach of Latent Semantic Analysis (LSA) to automatically induce a domain matrix from raw text corpus. This domain matrix is used in transforming the conventional *term by document* vector space model into a *term by domain* vector space model, where the domains are the ones induced by LSA. This is called the domain vector space model. They define a domain kernel function which evaluates distances among two words by operating upon the corresponding word vectors obtained from this domain vector space model. Traditionally these vectors are created using Bag Of Words (BOW) or Part-of-Speech (POS) features of words in surrounding context. The kernels using these traditional vectors are referred as the BOW kernel and the POS kernel respectively. Using the domain kernels, Gliozzo et al. have demonstrated significant improvement over BOW and POS kernels. By augmenting the traditional approach with domain kernels, their results show that only 50 percent of the training data is required in order to attain the accuracy offered by purely traditional approaches, thus reducing the knowledge acquisition bottleneck to a great extent.

Our work differs from Gliozzo et al., [19] in that we have proposed a generalization of their domain kernels based on different types of features used to create the domain matrix. Additionally we have used a word co-occurrence data representation proposed by Purandare and Pedersen [49] to develop Association kernels.

5.2 Previous Work on NLM WSD Collection

The National Library of Medicine (NLM) WSD collection is a set of 50 ambiguous medical terms collected from medical journal abstracts. It is a fairly new dataset and has not been extensively explored. Following is related work connected to this collection.

Liu et al., [35] evaluate the performance of various classifiers on two medical domain datasets and one general English dataset. The classifiers that they have considered include traditional decision lists, their adaptation of the decision lists, the naïve Bayes classifier and a mixed learning approach that they have developed. Their features included combinations of: (1) unigrams in various window sizes around the ambiguous word with their orientation and distance information and (2) two-word collocations (word co-occurrences) in a window size of two on either side of the ambiguous word, and not including the ambiguous word. The general biomedical term dataset that they used is a sub-set of the NLM WSD data collection that we have used for our experiments. They achieved the best results for the medical abbreviation dataset using their mixed learning approach and the naïve Bayes classifier. No particular combination of features, window size and classifiers provided stable performance for all the ambiguous terms. They therefore concluded that the various approaches and feature representations were complimentary in nature and as a result their mixed learning approach was relatively stable and obtained better results in most of the cases.

Our work differs from the work of Liu et al., [35] in two respects. First we have made use of a different subset of the NLM dataset and the MEDLINE abbreviation dataset. Second, we focus on the semi-supervised learning aspect of the WSD problem to make use of unlabeled data instances in improving accuracy on WSD, unlike the purely supervised approach take by Liu et al., [35].

Leroy and Rindflesch [31] explore the use of symbolic knowledge from the UMLS ontology for disambiguation of a subset of the NLM WSD collection. The basic features of the ambiguous word that they use are: the status of the ambiguous word in the phrase, whether it is the main word or not, and its part of speech. Unlike many BOW approaches which use the actual words in context as features, they make use of just the semantic types of words in the context as features. Additionally they also use semantic relations among the semantic types of non-ambiguous words. Finally, they also make use of the semantic relations of the ambiguous type with its surrounding types. The semantic types and their relations are derived from the UMLS ontology. Using the naïve Bayes classifier from the Weka data mining suite [55], their experiments were performed with incremental feature sets, thus evaluating the contribution of new features over the previous ones. They achieved significant improvements over the majority sense baseline in some cases, but observed degradation of performance in others. In general it was not the case that a maximum set of features yielded the best results. However, semantic word features in context and their relationship with the various senses of the ambiguous word were useful along with the information about whether the ambiguous word was the main word or not. Therefore this approach can possibly be used in combination with the conventional BOW approaches to improve the results.

Our approach differs from that of Leroy and Rindflesch [31] with respect to the source of external knowledge that has been used and also in the approach that uses this external knowledge. Leroy and Rindflesch [31] use a medical ontology to augment the feature set for the naïve Bayes classifier. We make use of unsupervised methods that derive semantic relationships from purely unlabeled corpora, and use these relationships to define semantic kernels for SVMs. Also, creation of a medical ontology involves significant manual effort, which is not required for raw text corpora.

5.3 Previous Work on SVMs and WSD in the General English Domain

In the last several years, a number of researchers have explored the use of Support Vector Machines in general English word sense disambiguation.

Cabezas et al., [7] present a supervised word sense tagger using Support Vector Machines. Their system was designed for performing word sense disambiguation independent of the language of lexical samples provided for the SENSEVAL-2 task. A lexical sample for an ambiguous word is a corpus containing several instances of that word, having multiple senses. Their system identified two types of features: unigrams in a wider context of the ambiguous word, and up to three words on either side of the ambiguous word with their orientation and distance with respect to the ambiguous word. The second feature captures the *collocations* containing the ambiguous word, in a narrow context around the word. Cabezas et al. use the term collocations to mean *word co-occurrences* unlike the more conventional linguistic sense which defines collocations as *two or more words that occur together more often than by chance*. These features were weighed according to their relevance for each ambiguous word, using the concept of Inverse Category Frequency (ICF) where the ICF score of a feature is higher when it is more representative of any particular sense. For multi-class classification of words having more than two senses, they employed the technique of building a "one against all" classifier for each of the senses. In this method, the classifier for a given sense categorizes all the instances into two classes:- one that represents the given sense and the other that represents anything that does not belong to the given sense. For any ambiguous word, the sense that is assigned is the one whose classifier voted for that sense with highest confidence. Their results show a convincing improvement over baseline performance.

Lee et al., [30] use Support Vector Machines to perform Word Sense Disambiguation for general English and for translating an ambiguous English word into its Hindi equivalent. They have made use of all the features available from the following knowledge sources: (1) Parts Of Speech (POS) of up to three words around the ambiguous word and POS of the ambiguous word itself, (2) morphological root forms of unigrams in the entire context, with function words, numbers and punctuations removed, (3) collocations, that is word co-occurrences consisting of up to three words around the ambiguity and (4) various syntactic relations depending upon the POS of the ambiguous word. They make use of all the extracted features and do not perform any kind of feature selection, that is they do not use any statistical or information gain measures to refine their feature set. Additionally, they have also used (5) the English sense of ambiguous words as a feature for the translation task, which improved their system's performance. They have made use of the SVM implementation available in the WEKA data mining suite [55], with the linear kernel and default parameter values. This is the exact configuration that we have used for our experiments. The results that they obtained for the general English corpus were better than those obtained for the translation task.

Ngai et al., [43] propose a supervised approach to semantic role labeling. The FrameNet corpus [4] is an ontologically structured lexical database that consists of semantic frames, lexical units that activate these frames, and a large corpus of annotated sentences belonging to the various semantic frames. A *semantic frame* is an abstract structure relating to some event or concept and includes the participant objects of the event or concept. These participant objects are known as *frame elements*. Frame elements are assigned semantic types wherever appropriate. A *lexical unit* is any word in a sentence (often the verb, but not necessarily so) that determines the semantic frame the sentence belongs to. For example, FrameNet has a semantic frame titled *Education_teaching*, two of its frame elements being *Teacher* and *Student* which have the semantic type *Sentient*. Some of the lexical units which activate this frame are *coach, educate, education, teach* and *instruct*. Ngai et al. propose to solve the problem of semantic role labeling of sentence parse constituents by posing it as a classification task of assigning the parse constituents to the appropriate frame element from the FrameNet corpus. This is in principle similar to our task where we aim at classifying words into different concepts as defined in the Unified Medical Language System (UMLS) repository, which is to some extent more "coarse" than word sense disambiguation in the conventional sense. They make use of the following types of features: lexical and syntactic features available from the FrameNet ontology, such as the lexical identity of the target word, its POS tag, syntactic category and extracted features such as the transitivity and voice of verbs, and head word of the parse constituent. They have tested different machine learning methods including SVMs, Maximum Entropy classifier, Sparse Network of Winnows (SNOW) and decision lists, individually as well as their ensembles (i.e., additive learning methods). Their best results from SVMs were obtained with polynomial kernel with degree four. For multi-class classification, they too have used the "one against all" approach. Although SVMs were not the best individually due to their comparatively lower recall scores, they obtained very high precision values and were part of the classifier ensemble that gave the best results.

The primary difference between all of the three works above and our work is that we make use of unlabeled data for generating semi-supervised semantic kernels, whereas the three works above are examples of purely supervised machine learning approaches.

6 Future Work

In this chapter we outline some possible future avenues of research based on the work done so far.

We categorize the future directions into two groups. The first group is related to extension of the kernel methods that we have implemented. The second group is related to the feature engineering aspect of Word Sense Disambiguation and the abbreviation expansion problem and also general ideas from Machine Learning.

6.1 Kernel Methods

For our Latent Semantic Analysis kernels, we currently utilize a domain matrix consisting of feature vectors along the rows and context vectors along the columns. For evaluating the similarity of any two contexts, we average together those feature vectors from our domain matrix. Another intuitive way of evaluating similarity of the two contexts would be to compute a vector for each of the two contexts that consists of the context's similarity values with each of the documents in the domain matrix representation. For example, if we generate a domain matrix of n features along the rows and k "contexts" along the columns (these are not the original contexts, but abstract concepts after the process of Singular Value Decomposition), then we have $n-$dimensional feature vectors for each of the abstract contexts. Given two new contexts to analyze, we should represent them as $n-$dimensional feature vectors using the same features along the rows of the domain matrix. Each of the two new context vectors should then be analyzed for similarity with the k abstract context vectors. This will yield k similarity values for each of the two new contexts, which can be expressed as $k-$dimensional similarity vectors. Once we have these similarity vectors for each of the contexts, the similarity between those vectors can be evaluated to yield a similarity value for the two contexts. This similarity value will be the semantic kernel function value for the two contexts.

As of now we simply add the semantic kernel value to the default linear kernel outcome. One possible extension is to use a different off-the- shelf kernel in the addition, such as adding the semantic kernel to the polynomial kernel. Another possible extension would be to transform the semantic kernel using polynomial or Gaussian functions as done by Cristianini et al., [9].

We would like to develop semantic kernels based on shallow syntactic features obtained from a shallow parser, in addition to the unlabeled corpora. The idea is to analyze the shallow parses of the contexts of an ambiguous word for similarity, using the concept of "edit distance", which measures the number of changes that one would have to perform in order to convert one shallow parse representing a context into the other shallow parse representing another context. This is an attractive option given that shallow parsing can be done with fairly high accuracy and the fact that Part-of-Speech features by themselves performed very well on the abbreviation disambiguation dataset from the Mayo Clinic.

6.2 Feature Engineering

As in the case of kernel method improvements, we would like to make use of shallow syntactic features from the context of an ambiguous word or abbreviation.

For the problem of abbreviation expansion, we believe that there is a good chance that even if the entire abbreviation expansion does not appear in context, one or more words from the expansion do. For example, for the abbreviation *PVC*, we have observed that when the correct expansion is *Polyvinyl Chloride*, then the word *chloride* tends to occur in the context of the abbreviation frequently. We would first like to verify the generality of this belief using elementary statistical analysis and if found valid, would like to make better use of such special features for abbreviation expansion.

From a general Machine Learning perspective, we would like to apply our methods to a bigger set of data and test their generality. We would also like to compare our approach with a wider variety of machine learning methods such as Bayes Net learning and Boosting approaches other than AdaBoostM1.

7 Conclusions

Word Sense Disambiguation (WSD) is the problem of automatically deciding the correct sense of an ambiguous word based on its surrounding context. Automatic abbreviation expansion for abbreviations that have multiple possible expansions can be treated as a WSD problem with the multiple expansions acting as “senses” for the abbreviation.

The most popular approaches to WSD rely on supervised machine learning methods, where a machine learning classifier is required to be trained on manually labeled training instances, to generate a classifier model that can be used to classify future instances. These methods however face the problem of knowledge acquisition bottleneck, where the amount of labeled data provided to the classifiers is limited.

The goal of this thesis is to explore kernels for Support Vector Machines developed using unsupervised methods that can learn semantic relationships from unlabeled data. We have applied our kernel methods to the problems of word sense disambiguation and automatic abbreviation expansion in texts relating to the medical domain. We have developed two classes of semantic kernels based on existing unsupervised learning methods and their variations. The first class of Latent Semantic Analysis (LSA) kernels is a generalization proposed on existing LSA based kernels [9, 19] . The second class of Word Association kernels is based on existing unsupervised methods [49] that learn word correlation-ships from unlabeled data.

Additionally we also focus on the feature engineering aspect of the WSD problem for medical domain, especially for the domain of clinical notes at the Mayo Clinic. We have made use of feature specific to the domain of clinical notes and introduced a flexible window approach to capture word features in context of the ambiguous word. The features we have used are the gender code of the patient corresponding to the clinical note, the department code of the department from where the clinical note originated and the section identifier of the section of the clinical note in which the abbreviation occurs.

Based on our semantic kernel experiments on the medical domain datasets, we conclude that kernel methods based on unlabeled text to incorporate external knowledge into SVM learners are able to achieve higher accuracy on the task of WSD in the medical domain, provided that the quality of the unlabeled data is good. We have found that some words with a small number of unlabeled instances showed significant improvements using kernel methods, whereas words with a large number of unlabeled instances sometime showed a degradation in performance. This is currently a problem with our kernel methods and we would like to adapt our methods to prevent degradation in the baseline performance achieved using standard SVM learners.

We also find that the sense distribution of an ambiguous word or abbreviation is a crucial factor in improving the accuracy using kernel methods. The best improvement using our kernel methods is seen for words that have a balanced sense distribution.

We find that our flexible window approach significantly improves accuracy for unigram features and also for the bigram features. The large boost in bigram performance can be explained based on the fact that our feature selection criteria for bigrams are stringent and a small fixed window around the ambiguous word has a smaller chance of containing a significant bigram, and hence allowing the window to be flexible to include to the significant bigram helps considerably.

The use of features specific to the clinical notes domain does improve performance, but not significantly. This is also the case with Part-of-Speech tag features used in addition to unigrams and bigrams. However combining unigrams, bigrams, Part-of-Speech features and features specific to the clinical notes yields a significant improvement in accuracy.

Overall, we believe that a semi-supervised approach using the unigram LSA kernels and the bigram Association kernels based on unlabeled data helps in improving the accuracy of WSD and abbreviation expansion. The use of a flexible window for unigrams and bigrams and the clinical-note features combined with Part-of-Speech features improve accuracy for abbreviation expansion.

References

[1] Steven Abney. Parsing by Chunks. In *Principle-Based Parsing: Computation and Psycholinguistics*. Kluwer, 1991.

[2] Julian Ambrus. Acronyms and abbreviations. *The Journal of Medicine*, 18(3–4), 1987.

[3] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk A. van der Vorst, editors. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 2000.

[4] Collin Baker, Charles Fillmore, and John Lowe. The Berkeley Framenet project. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, 1998.

[5] Satanjeev Banerjee and Ted Pedersen. The Design, Implementation and Use of the Ngram Statistics Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, 2003.

[6] Ezra Black. An Experiment in Computational Discrimination of English Word Senses. *IBM Journal of Research and Development*, 32(2), 1988.

[7] Clara Cabezas, Philip Resnik, and Jessica Stevens. Supervised Sense Tagging using Support Vector Machines. In *Proceedings of* SENSEVAL-2*, Second International Workshop on Evaluating Word Sense Disambiguation Systems. SIGLEX, Association for Computational Linguistics*, 2001.

[8] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, 2000.

[9] Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. In *Proceedings of 18th International Conference on Machine Learning (ICML'01)*, 2001.

[10] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applica-

tions. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[11] Walter Daelemans, Sabine Buchholz, and Jorn Veenstra. Memory-based Shallow Parsing. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL '99) workshop on Computational Natural Language Learning (CoNLL-99)*, 1999.

[12] Scott Deerwester, Susan Dumais, Thomas Landauer, George Furnas, and Richard Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6), 1990.

[13] Ted Dunning. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1), 1994.

[14] Christiane Fellbaum, editor. *WordNet, an Electronic Lexical Database*. MIT Press, 1998.

[15] John Firth. A Synopsis of Linguistic Theory, 1930-1955. In *Studies in Linguistic Analysis*. Oxford University Press, 1957.

[16] Yoav Freund and Robert Schapire. A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. In *Proceedings of the Second European Conference on Computational Learning Theory (EuroCOLT '95)*, 1995.

[17] Yoav Freund and Robert Schapire. Experiments with a New Boosting Algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML '96)*, 1996.

[18] William Gale, Kenneth Church, and David Yarowsky. A Method for Disambiguating Word Senses in a Large Corpus. *Computers and the Humanities*, 26, 1992.

[19] Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. Domain Kernels for Word Sense Disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, 2005.

[20] Alfio Gliozzo and Carlo Strapparava. Domain Kernels for Text Categorization. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, 2005.

[21] Zellig Harris. *Mathematical Structures of Language*. Interscience publishers, 1968.

[22] Mark Hepple. Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 2000.

[23] Nancy Ide and Jean Véronis. Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, 24(1), 1998.

[24] Thorsten Joachims. Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. In *Proceedings of the Tenth European Conference on Machine Learning*, 1998.

[25] Thorston Joachims. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[26] Mahesh Joshi, Serguei Pakhomov, Ted Pedersen, and Christopher Chute. A Comparative Study of Supervised Learning as Applied to Acronym Expansion in Clinical Reports, 2006.

[27] Mahesh Joshi, Serguei Pakhomov, Ted Pedersen, Richard Maclin, and Christopher Chute. An End-to-End Supervised Target-Word Sense Disambiguation System. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06), Intelligent Systems Demonstrations*, 2006.

[28] Mahesh Joshi, Ted Pedersen, and Richard Maclin. A Comparative Study of Support Vector Machines Applied to the Supervised Word Sense Disambiguation Problem in the Medical Domain. In *Proceedings of the 2nd Indian International Conference on Artificial Intelligence (IICAI'05)*, 2005.

[29] Thomas Landauer, Peter Foltz, and Darrell Laham. An Introduction to Latent Semantic Analysis. *Discourse Processes*, 25, 1998.

[30] Yoong Keok Lee, Hwee Tou Ng, and Tee Kiah Chia. Supervised Word Sense Disambiguation with Support Vector Machines and Multiple Knowledge Sources. In *Proceedings of* SENSEVAL-3*: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.

[31] Gondy Leroy and Thomas Rindflesch. Using Symbolic Knowledge in the UMLS to Disambiguate Words in Small Datasets with a Naïve Bayes Classifier. In *Proceedings of the Eleventh World Congress on Medical Informatics (MEDINFO 2004)*, 2004.

[32] Hongfang Liu, Alan Aronson, and Carol Friedman. A Study of Abbreviations in MEDLINE Abstracts. In *Proceedings of the American Medical Informatics Association Symposium (AMIA-02)*, 2002.

[33] Hongfang Liu, Stephen Johnson, and Carol Friedman. Automatic Resolution of Ambiguous Terms Based on Machine Learning and Conceptual Relations in the UMLS. *Journal of American Medical Informatics Association*, 9(6), 2002.

[34] Hongfang Liu, Yves Lussier, and Carol Friedman. A Study of Abbreviations in the UMLS. In *Proceedings of the American Medical Informatics Association Annual Symposium (AMIA'01)*, 2001.

[35] Hongfang Liu, Virginia Teller, and Carol Friedman. A Multi-aspect Comparison Study of Supervised Word Sense Disambiguation. *Journal of American Medical Informatics Association*, 11(4), 2004.

[36] MEDLINE. Medical Literature Analysis and Retrieval System Online.

[37] Rada Mihalcea and Dan Moldovan. An Automatic Method for Generating Sense Tagged Corpora. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, 1999.

[38] George Miller and William Charles. Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes*, 6(1), 1991.

[39] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[40] Saif Mohammad and Ted Pedersen. Combining Lexical and Syntactic Features for Supervised Word Sense Disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL '04)*, 2004.

[41] Raymond Mooney. Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1996.

[42] Hwee Tou Ng. Getting Serious About Word Sense Disambiguation. In *Proceedings of the Workshop on Tagging Text with Lexical Semantics: Why, What, and How? (ANLP-97 Workshop) at the Fifth Conference on Applied Natural Language Processing*, 1997.

[43] Grace Ngai, Dekai Wu, Marine Carpuat, Chi-Shing Wang, and Chi-Yung Wang. Semantic Role Labeling with Boosting, SVMs, Maximum Entropy, SNOW, and Decision Lists. In *Proceedings of* SENSEVAL-3*: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.

[44] Serguei Pakhomov. Semi-Supervised Maximum Entropy Based Approach to Acronym and Abbreviation Normalization in Medical Texts. In *Proceeding of the 40th Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[45] Serguei Pakhomov, Ted Pedersen, and Christopher Chute. Abbreviation and Acronym Disambiguation in Clinical Discourse. In *Proceedings of the American Medical Informatics Association Annual Symposium (AMIA'05)*, 2005.

[46] Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. Using Measures of Semantic Relatedness for Word Sense Disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, 2003.

[47] Ted Pedersen. A Decision Tree of Bigrams is an Accurate Predictor of Word Sense. In *Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-01)*, 2001.

[48] Amruta Purandare. *Word Sense Discrimination by Clustering Similarity Contexts*. Department of Computer Science, University of Minnesota, Duluth, 2004.

[49] Amruta Purandare and Ted Pedersen. Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL '04)*, 2004.

[50] Gerard Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the Association for Computing Machinery (ACM)*, 18(11), 1975.

[51] Martijn Schuemie, Jan Kors, and Barend Mons. Word Sense Disambiguation in the Biomedical Domain: An Overview. *Journal of Computational Biology*, 12(5), 2005.

[52] UMLS. *UMLS Knowledge Sources*. National Library of Medicine, 12th edition, 2001.

[53] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.

[54] Marc Weeber, James Mork, and Alan Aronson. Developing a Test Collection for Biomedical Word Sense Disambiguation. In *Proceedings of the American Medical Informatics Association Annual Symposium (AMIA 2001)*, 2001.

[55] Ian Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan-Kaufmann, 2005.

[56] David Yarowsky. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, 1995.