UNIVERSITY OF MINNESOTA

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

Varada Kolhatkar

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. Ted Pedersen

Name of Faculty Adviser

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

An Extended Analysis of a Method of
All Words Sense Disambiguation

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Varada Kolhatkar

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

August 2009

Acknowledgements

I express my heartfelt gratitude towards my adviser Dr. Ted Pedersen for his guidance during my time at UMD. He has been a great advisor and I thoroughly enjoyed working with him. I sincerely appreciate his close reading and valuable comments. Among other countless things, he has introduced me to the field of Natural Language processing for which I will be always grateful to him.

I sincerely thank Dr. Joe Gallian and Dr. Hudson Turner for being on my thesis committee and for their detailed reading of my thesis. The insightful questions and comments they raised often gave me new perspectives on my research.

I am very grateful to Dean Riehl and Dr. Carolyn Crouch for providing travel funding to the Annual Meeting of the North American Chapter of the Association for Computational Linguistics.

I would also like to thank all the professors from whom I have had the opportunity to learn. Thank you also to the friendly staff of the Computer Science department.

None of this work would have been possible without the support of my friends and family who cheered me up and motivated me whenever I needed it. Finally, I thank the great lake of Duluth for the company of her natural beauty and soothing spirit on many long walks.

Abstract

One of the central problems in processing a natural language is ambiguity. In every natural language there are many potentially ambiguous words. Humans are fairly adept at solving ambiguity by drawing on context and their knowledge of the world. However, it is not so easy for machines to understand the intended meaning of a word in a given context. *Word Sense Disambiguation* (WSD) is the process of selecting the correct sense of a word in a specific context.

It is often useful to generalize the problem of disambiguating a single word to that of disambiguating all content words in a given text. This generalized problem is referred to as *all-words sense disambiguation*. The long history of WSD research includes many different supervised, unsupervised and knowledge-based approaches. But the reality is that current state-of-the-art accuracy in WSD remains a long way off far from natural human abilities.

This thesis presents our analysis of some of the components that might be contributing to the level of error currently plaguing all-words sense disambiguation. Our analysis makes use of WordNet::SenseRelate::AllWords, an unsupervised knowledge-based system for all-words sense disambiguation, which is freely available on the Web as a perl Module. The system assigns a WordNet sense to each word in a text using measures of semantic similarity and relatedness.

We find that the degree of difficulty in disambiguating a word is proportional to the number of senses of that word (polysemy), which confirms the conclusion of Daelemans [10]. The experimental evidence indicates that a significant percentage of word sense disambiguation error is caused by a relatively small number of highly frequent word types. We also demonstrate that part-of-speech tagged text will be disambiguated more accurately than raw text. We show that expanding the context window helps in terms of coverage but doesn't improve disambiguation. Finally we find that if the answer is not the most frequent sense, disambiguation turns out to be a hard problem even for an unsupervised system which doesn't use any information about sense distribution.

Contents

List of Figures

List of Tables

1 Introduction

Words in a natural language often have multiple senses. For example, *squash* can mean *the vegetable squash* or *the game squash*. Nonetheless humans can for the most part resolve word ambiguities quite easily by looking at the context in which a word is used. For example, given the following sentences, it is not very hard for humans to distinguish between the senses of *blue* used as *gloomy mood* in the former and as *blue color* in the latter.

> The sorrow etched in his face reflected his thoroughly **blue** mood.
> The **blue** mountains were looking beautiful.

However, it is not so easy for machines to understand the intended meaning of a word in a given context. Correctly understanding the meaning of particular instances of a given word requires successfully distinguishing between different senses of that word. It is then naturally very useful if machines used for manipulating language have the ability to differentiate different senses of a word. Thus, for instance, machine-translation of the above sentences into a different language will ideally produce translations that reflect different meanings of *blue*, as these would be picked up by a native English speaker.

In Computational Linguistics terms, this problem is known as Word Sense Disambiguation(WSD). WSD is defined as the task of automatically assigning the correct sense to a given word based on the context in which it occurs. WSD has a long history of research and is considered one of the hardest problems in Artificial Intelligence. In the late 1940s, WSD was first thought of as a part of Machine Translation, the general field investigating the use of computer software to translate from one natural language to another. WSD quickly proved an immensely challenging problem. In the 1970s, several attempts to solve the problem using Artificial Intelligence techniques were made. In the 1980s, the release of large scale lexical resources enabled automatic extraction of knowledge and WSD research reached a turning point (e.g. Wilks et al., 1990 [44]). Later in the 1990s, WSD was mainly dominated by statistical and machine learning approaches.

WSD task is viewed broadly in two different ways. In a lexical sample task (also called ‘target word task’), a sample of words is selected from the lexicon and the selected words are disambiguated in a

short given context. For instance, Figure 1 shows two instances of the target word *line* used in two different contexts. The lexical sample task would be to disambiguate each token of *line* in the given context.

> Culinova fresh entrees, launched in 1986 by Philip Morris Cos.'s General Foods Corp., hit similar distribution problems. Last December, shortly after Philip Morris bought Kraft Inc., the struggling **line** was scrapped.
> For now , we'll concentrate on the system. Some of the major data banks , like the ones held by Defense , may have five hundred or a thousand access **lines**.

Figure 1: Lexical sample example

It is assumed that the word to be disambiguated has a fixed set of senses in the sense inventory, where the sense inventory contains the mapping of words and their different senses. Given this assumption, WSD can be thought of as a classification problem where, given a word in a context and its possible senses in a lexicon, the task is to classify the occurrence of the word in one or more sense classes. For instance, the classification problem may look like: Given two sense classes *line* and *phone*, classify the examples in Figure 1 in one of the two classes based on whether *line* is used with the *phone* sense or the *product* sense. For this task, it is possible to apply Machine Learning classification techniques that build highly accurate models.

However, lexical sample disambiguation is limited, in that it is only able to disambiguate a few words at a time. To be useful in a practical setting, disambiguation should be considered in more general terms. The all-words task serves that purpose.

Given a piece of text and a lexicon with a sense inventory, the all-words task is to disambiguate every content word in the text based on the context in which it occurs. More formally, let T be a tuple made up of k content words $T = (c_1, c_2, ..., c_k)$ and S be the sense inventory such that $\forall c_i \epsilon T$, c_i is defined in S with finite number of senses associated with each word c_i. Suppose each c_i has m_i possible senses in S, denoted as $\{s_{i1}, s_{i2}, ..., s_{im_i}\}$. The all-words task is to select for every c_i, one of the senses from the set $\{s_{i1}, s_{i2}, ..., s_{im_i}\}$ as the most appropriate sense.

The complexity of the problem lies in the fact that the mapping between words and senses is many to many. A word can have multiple meanings and a meaning or a sense can have multiple words associated with it (synonyms). Even so, being able to disambiguate all words in a text makes the

disambiguation more useful, because it helps understand the overall meaning of a sentence and thus can be used more generally in the translation, searching or summarization of a text.

It is important to note that all-words is not just an extension of the lexical sample problem. When it comes to disambiguating all content words in a text, the classification problem we discussed before becomes very complicated, introducing an enormous number of classes, which creates difficulty in using machine learning techniques for classification with their usual settings.

Several approaches have been proposed for tackling WSD. *Supervised* approaches use a large sense-annotated corpus for training and use supervised learning techniques. A sense-annotated corpus is created by humans manually annotating each occurrence of the target word (in case of lexical sample task) or all content words (in case of all-words task) in a text. A good deal of progress has been made in supervised WSD for the lexical-sample task, achieving good accuracies. Different supervised learning techniques such as decision lists, decision trees, naive bayes classifier, neural networks, instance-based learning, and support vector machines have been tried. But these approaches need manually sense-tagged corpora for training the classifiers.

Unfortunately, creation of such a sense-tagged corpus is a time consuming and expensive process. It might be relatively easy for the lexical sample task since the annotator disambiguates a single target word occurring many times making her familiar with the various senses of the word. However, in the all-words setting, a human annotator will encounter many words only one time, and will have to study and learn the sense inventory for a word simply to tag one occurrence, and then must move on to another word which might well be new and need to be learned. This also makes all-words annotation less accurate than lexical sample annotation. As a result of these difficulties, a very small number of such manually sense-tagged corpora are available. The availability becomes more of an issue when it comes to disambiguating a text in a certain domain, say a text from Biology. The scarcity of the sense-tagged data prevent the use of supervised methods in such cases.

This leads to using *semi-supervised* and *unsupervised* approaches. The former uses very little annotated training data and the latter uses no training data. Yarowsky's bootstrapping algorithm [46] is a semi-supervised algorithm that exploits a decision list and uses a self-training approach. Unsupervised methods include clustering. This type of WSD uses the idea that the same sense of a word will have similar surrounding words and thus it creates clusters of words based on surrounding

words. A review of all of the above approaches can be found in Manning and Schütze [17]. It has been observed that given sufficient training data, supervised WSD approaches outperforms unsupervised ones. But the unavailability of sufficient sense-annotated data leads to using unsupervised approaches for broad coverage WSD.

Because of the availability of a variety of lexical databases, ontologies and thesauri, approaches that use knowledge in such resources have become popular. Particularly with the release of the lexical database WordNet in the late 1980s, a variety of methods exploiting the structure of such resources have been proposed. Methods which use external knowledge for disambiguation are called *knowledge-based methods*. These methods can again be supervised or unsupervised. Supervised knowledge-based methods use knowledge and manually-annotated training data for disambiguation, while unsupervised knowledge-based methods use knowledge extensively, but have the advantage that they don't need any manually annotated data.

The method used in this thesis is an unsupervised knowledge-based method for all-words sense disambiguation. It doesn't require any manually-annotated training corpus but makes extensive use of the lexical database WordNet for disambiguation. The method was originally developed by Michelizzi in 2003 [19] and is now enhanced significantly.

Before turning to the results presented in this thesis, it is important to mention why we are doing this research. Apart from being an interesting problem in an academic setting, all-words sense disambiguation also has significant utility in real world applications. For example, being able to translate the sentence '*The sorrow etched in his face reflected his thoroughly* ***blue*** *mood.*' sensibly into other languages (without translating the word *blue* as *blue color*) requires the words to be disambiguated correctly. As an another example, in information retrieval, if the query includes long phrases or sentences, all-words sense disambiguation has a potential application. Beyond these applications WSD would also be useful in lexicography and question answering. Kilgariff [12] describes some of these applications.

That said, the state-of-the-art accuracy of all-words sense disambiguation is not very encouraging, a fact which prevents its wide use in the applications mentioned above. Improving the ability to assign senses to all the words that appear in a written document will significantly advance the state of the art in Natural Language Processing, in particular methods that seek to understand documents so as

to better summarize, categorize, or translate them. In this thesis, we present our analysis of some of the components that might be contributing to the level of error currently plaguing all-words sense disambiguation. We present an extensive set of experimental results along with the observations we made.

The thesis starts by formalizing the algorithm of Michelizzi [19] in mathematical and graph theoretic notations. It also discusses the time complexity of the algorithm.

Inspired by the question raised by George Miller [22], regarding how much context is required for WSD, we evaluate our algorithm with different context sizes in order to determine the effects of expanding the context. To see where exactly the error lies, we score the results in different ways and this thesis presents those results along with the observations we made.

Polysemy, in general, is the the property of having multiple senses. This poses a problem in WSD. For example, *bass* can mean a *a kind of a fish* or *the lowest part of the musical range*. When translating from English to some other language, if the word is not disambiguated correctly, it may be translated by a wildly inaccurate term, resulting in an awkward - if sometimes humorous - translation. There are mixed conclusions about polysemy and the difficulty of disambiguation. Preiss [33] argues that polysemy is not an ideal measure of difficulty. On the other hand Daelemans [10] concludes that the fluctuations in accuracy of disambiguation largely depend on the polysemy and entropy of the ambiguous words. In this thesis we further present results on the relation of polysemy to the difficulty of disambiguation.

In general, the words in a text follow a Zipfian distribution [47]. This means that that a few words occur very frequently, while many words only occur a few times. Words like *have* and *be* occur more frequently than words like *metamorphosis* and there are many words like *metamorphosis* which only occur a few times. In order to improve overall disambiguation we need in particular to see if the most frequently occurring words are disambiguated correctly. This thesis provides the disambiguation results of frequently occurring words and presents our analysis of the difficulty in disambiguating some of these frequently occurring words.

This thesis also examines to what degree the errors in identifying parts-of-speech contribute to the overall disambiguation error. The WSD problem can be thought of as the task of assigning both

the correct part-of-speech to a word and then the correct sense from among the possible senses of the word as that part-of-speech. For example, in the news line *Grant helps **train** students in Computational Mathematics research*, it is first necessary to identify the part-of-speech of *train* and then assign the appropriate sense to it. This gives an indication of the importance of the part-of-speech tag or syntactic information for disambiguation.

To summarize briefly, this research presents an extensive set of experimental results on issues central to the future direction of WSD, together with our related observations.

Other contributions include

- We refined the method developed by Michelizzi [19] in significant ways and carried out an extensive evaluation of the method on almost all available sense-tagged corpora by introducing different scoring mechanisms.
- We also developed a web interface to the refined system[1].
- We have released (via the CPAN archive) a freely available software package[2] that includes the above enhancements.

[1] http://talisker.d.umn.edu/allwords/allwords.html

[2] http://search.cpan.org/dist/WordNet-SenseRelate-AllWords/

2 Background

2.1 WordNet

WordNet [24] is an electronic lexical database for English which is widely used in the Natural Language Processing (NLP) community for applications in Information Retrieval, Machine Translation and Word Sense Disambiguation.

WordNet grew out of research at Princeton University in the 1980s by psychologist George Miller's group about how children acquire vocabulary. In the course of this research, they started to record the relations between and among words. WordNet forms a large network out of these relations which can be visualized as a directed acyclic graph where each node represents *a concept* and an edge or a link between nodes represents the *relation* between the concepts. Each node consists of one or more words that are *cognitively synonymous*, meaning that the words are approximately equivalent and can be substituted in many contexts. The groups of such synonymous words are called synonymous sets or *synsets*. Each synset represents a concept, and a unique gloss or definition, possibly with usage examples, is associated with it. For example, {squash, squash racquets, squash rackets} is the synset for the gloss *a game played in an enclosed court by two or four players who strike the ball with long-handled rackets*. Henceforth, we use the terms "synset," "concept" and "node" interchangeably.

Figure 2 shows a portion of the WordNet[3] entry for the word *squash*. In this thesis, combination of a word, part-of-speech and sense is denoted by word#pos#sense. For example, the second noun sense of *squash* will be denoted as *squash#n#2*.

WordNet considers four syntactic categories, *noun (n), verb (v), adjective (a)* and *adverb (r)* which are called open class categories and doesn't include closed class categories of English (propositions, pronouns, and delimiters). The synsets were added separately for each category, forming four different networks. The networks of nouns and verbs may be viewed as hierarchies where the nodes at the higher level represents general concepts such as *entity* or *object* and the leaf nodes represent more specific concepts such as *apple pie* or *butternut squash*. Within each category, the network de-

[3] http://wordnet.princeton.edu/wordnet/download/

scribes lexical and semantic relationships between synsets. The relations within nouns in WordNet are considered as one of the distinguishing characteristic of WordNet. The hierarchies for nouns are deeper, making it useful for Natural Language Processing applications. The hierarchies for verbs are many and shallow. Relations in WordNet generally do not cross part-of-speech. Although there exist some derivational links between nouns and verbs and relations such as an *attribute* relation that expresses the relationship between a noun and an adjective. For example, an attribute of the adjective *beautiful* is the noun *beauty*. However, the links that cross part-of-speech are very sparse.

Overview of noun squash

The noun squash has 3 senses (no senses from tagged texts)

1. squash, squash vine – (any of numerous annual trailing plants of the genus Cucurbita grown for their fleshy edible fruits)
2. squash – (edible fruit of a squash plant; eaten as a vegetable)
3. squash, squash racquets, squash rackets – (a game played in an enclosed court by two or four players who strike the ball with long-handled rackets)

Overview of verb squash

The verb squash has 1 sense (first 1 from tagged texts)

1. (1) squash, crush, squelch, mash, squeeze – (to compress with violence, out of natural shape or condition; ”crush an aluminum can”; ”squeeze a lemon”)

Figure 2: A portion of the WordNet 3.0 entry for the word *squash*

The most common and useful relation for nouns is *is-a*. Figure 3 illustrates the is-a hierarchy of WordNet. An is-a relation defines the *is-a-kind-of* relationship between synsets. In WordNet terminology, it is described with a *hyponym* and *hypernym* pair. For example, in Figure 2, {*plant material, plant substance*} is a hypernym of *wood* and *wood* is a hyponym of {*plant material, plant substance*}. A synset can have more than one hypernyms. For example, in Figure 3, *cheese* has two hypernyms, *dairy product* and {*food, solid food*}. Verbs are related with each other by the is-a-way-of-doing relationship. For example, *bake* is a way of *cook*. *Meronymy* and *holonymy* describe has-a relationship for noun synsets. If B is a part of A then *B* is a *meronym* of *A* and *A* is a *holonym* of *B*. For example, *accelerator* is a meronym of *car* and *car* is a holonym of *accelerator*. Adjective and adverbs are not linked with is-a or has-a relations. Some relations like antonymy, similarity and

see also are applicable for these part-of-speech. *A* is called an antonym *B*, if they express opposite concepts. For example, *poor* is an antonym of *rich*. This relation holds for all parts-of-speech. Table 1 shows some useful WordNet relations along with examples.

WordNet has now become a large lexical database for the English language which comprises of about 155,287 words organized in over 115,000 synsets for a total of 207,000 word-sense pairs. It contains about 117,700 nouns, 11,500 verbs, 21,400 adjectives and 4400 adverbs.

The structure of WordNet is well suited for tasks where interpretation of a word based on its lexical semantics is required, and thus has become a very useful resource for research in WSD.

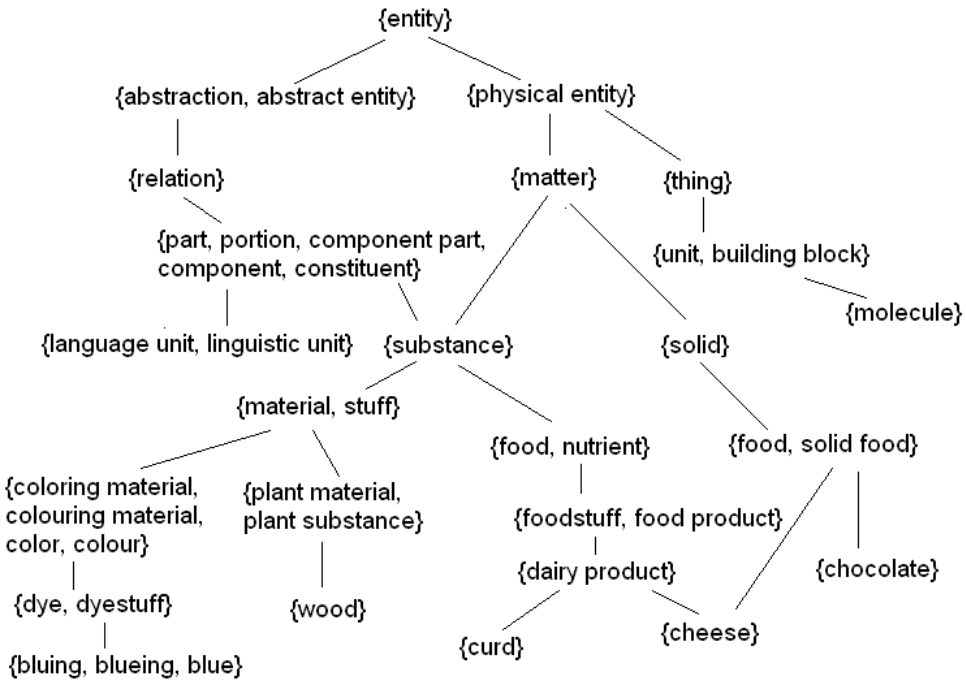

Figure 3: Illustration of WordNet *is-a* relations

Relation	Description	Example
ants(n,v,a,r)	Antonyms	*rich* is an antonym of *poor*
hype(n,v)	Hypernyms	*squash* is a hypernym of *winter sqash*
hypo(n,v)	Hyponyms	*baking* is a hyponym of *cooking*
entav	Verb Entailment	Entailment of *arrive at* is *travel*
syns(n,v,a,r)	Synonyms	*pretty* is a synonym of *beautiful*
meron	All Meronyms	*accelerator* is a meronym of *car*
holon	All Holonyms	*car* is a holonym of *accelerator*
pert(a,r)	Pertainyms	Adj. *neural* pertains to noun *neuron*
attr(n,a)	Attributes	Attributes of adj. *beautiful* is noun *beauty*
deri(n,v)	Derived Forms	The noun *locomotion* is derived from the verb *locomote*

Table 1: Some of the WordNet Relations. The parenthesis denote the possible parts-of-speech.

2.2 Measures of Similarity and Relatedness

With the release of lexical database WordNet, many WSD approaches that exploit the structure of WordNet were proposed. Most of the approaches are based on measuring the semantic similarity or relatedness of concepts. Semantic similarity or relatedness is the degree to which two concepts are related. In most of the cases, humans are able to tell the degree to which two concepts are related. For example, they can tell that *computer* and *keyboard* are more related than *squash* and *keyboard*. Similarity and relatedness measures try to quantify the degree of similarity or relatedness between two concepts. This in turn can be used to find the meaning of a word based on its linguistic contexts.

This thesis makes a distinction between the notions of similarity and relatedness. Relatedness is considered more general than similarity, in that two concepts can be related although they are not similar. For example, *rich#a#1* and *poor#a#2* are related with the antonymy relation in WordNet but they are not similar. In other words, similarity measures are limited to the is-a hierarchies in WordNet whereas relatedness measures can be applied to all kinds of relations. Since is-a hierarchies are applicable only in case of noun-noun and verb-verb pairs, similarity measures can't go beyond these part-of-speech. On the other hand, relatedness measures can be applied to all open class part-of-speech.

A large number of similarity and relatedness measures have been proposed to date. This thesis uses the implementation of similarity and relatedness measures provided by the freely available Perl software WordNet::Similarity[4] developed by Pedersen et al. [32]. We'll now briefly discuss the measures implemented in WordNet::Similarity.

These measures are grouped into three categories.

2.2.1 Path Based Measures

Given the *is-a* hierarchies in WordNet, the most intuitive way of finding similarity between synsets is by using the *path-length*, that is by counting the number of edges between two synsets and deriving a formula based on how many edges or nodes lie on the path between the synsets. The greater

[4] http://wn-similarity.sourceforge.net/

the path-length, the less similar the synsets are. For example, in Figure 3, the path length between *cheese* and *chocolate* is 2 and between *cheese* and *wood* is 7 indicating that *cheese* and *chocolate* are more similar than *cheese* and *wood*.

The most basic formula for path based similarity measures is given as

$$Sim_{path}(s1, s2) = \frac{1}{length(s1, s2)} \tag{1}$$

where, $length(s1, s2)$ is the shortest path-length between synsets $s1$ and $s2$.

Unfortunately, path length measure is not very well suited for the hierarchies in which the individual nodes have different interpretations. Sussna [43] observed that, for WordNet, the nearby synsets that are more specific (deep) in the hierarchy tend to be closely related to each other than the synsets which are same path length apart but are more general (higher) in the hierarchy. For example, in Figure 3, *curd* and *chocolate* are more related to each other than *relation* and *matter* though both have path-length four. This introduces the notion of the *depth* of a synset. The depth of a synset is defined as the path-length between the synset and the root node of the taxonomy. The depth of the taxonomy is defined as the longest path between the leaf node and the root of the hierarchy.

In order to address the problem of simple path-length for measuring similarity, a variety of measures were suggested which manipulate the measure by using the depth of the synsets and the depth of the taxonomy.

Rada et al. [35] proposed a measure based on path-lengths for measuring semantic relatedness of medical terms, using a medical taxonomy called MeSH. Leacock and Chodorow [13] suggested a similar kind of measure for WordNet. The similarity between two synsets is measured using path-length between the synsets and then scaling it by the depth D of the taxonomy.

Thus they define the similarity between two synsets s_1 and s_2 as

$$Sim_{lch}(s_1, s_2) = -\log(\frac{length(s_1, s_2)}{2 \times D}) \tag{2}$$

Not all hierarchies in WordNet are of same depth. Some are deeper and others are very shallow. The introduction of a unique root node in WordNet 3.0, creates 2 different taxonomies, one for nouns

and one for verbs[5]. This associates a constant number with the depth of a taxonomy.

As we noted, a synset can have multiple hypernyms. The common ancestors of two synsets are called subsumers and the most specific ancestor among those is called as the lowest common subsumer (lcs). For example, in Figure 3, the subsumers of *cheese* and *chocolate* are {*food, solid food*}, *matter, physical entity* and *entity* and the lowest common subsumer (lcs) is {*food, solid food*}.

In 1994, Wu and Palmer [45] suggested to use depth of the least common subsumer and the path lengths between the synsets and the least common subsumer. Resnik [38] suggested a slight modification of using depths of the synsets instead of using the path lengths between the synsets and the least common subsumer. Thus the measure is defined in terms of finding the depth of the lcs and then scaling it by summation of the depths of the two synsets.

$$Sim_{wup}(s_1, s_2) = \frac{2 \times (depth(lcs(s_1, s_2)))}{depth(s_1) + depth(s_2)} \tag{3}$$

Wu and Palmer describe this measure relative to a verb taxonomy, but in fact the measure can be used equally well for other parts-of-speech as long as the concepts are arranged in a hierarchy.

The measures we discussed so far were limited to *is-a* relations. Hirst and St-Onge [9] suggested a path based measure of relatedness that goes beyond *is-a* relations. Because of this, the measure is able to find relatedness between two synsets across different parts-of-speech. For example, it can find relatedness between *money#n* and *rich#a*. The measure is based on finding lexical chains between two synsets. The intuition behind the measure is that the closely related synsets will lie in the lexical chain that is neither too long nor has many changes in direction. The degree of relatedness is described with the nominal values *extra-strong, strong, medium-strong* and *weak*. Two synsets have extra-strong relatedness if they are identical. The relatedness is strong in three scenarios. First, when synsets are synonyms. Second, when there exists a single link between the synsets (e.g. in case of antonym). Third, when one synset is a compound word of another word and when a WordNet relation exists between them (e.g. *winter squash* is a compound word of *squash* and *squash* is a hypernym of *winter squash*).

[5]In WordNet 3.0, there was a structural change that linked all the nouns together by introducing a hypothetical root node.

The medium-strong relatedness is decided based on the weights calculated as below.

$$weight = C - path\text{-}length - (k \times \#changes\text{-}in\text{-}direction) \tag{4}$$

where C and k are constants. The constant k decides how much weight should be given to the number of changes in direction. In WordNet::Similarity the values for C and k are 8 and 1 respectively. If there are not many changes in the direction and the synsets not far away in the lexical chain, then the weight is higher. A weak relation has a weight of 0.

2.2.2 Information Content Based Measures

Information content measures the specificity of a concept. Specificity can be thought of as the reciprocal of the occurrence frequency of a concept. A frequently occurring concept is less specific and has lower information content while a rarely occurring concept is more specific and has higher information content. For example, the more general concept *object#n#1* would have low information content, while the more specific concept *apple_pie#n#1* will have a high information content.

Mathematically, information content for a concept s is defined as the negative log of the probability of concept s in a given corpus containing N distinct concepts.

$$IC(s) = -\log P(s) \tag{5}$$

And the probability $P(s)$ is defined as

$$P(s) = \frac{frequency(s)}{N} \tag{6}$$

An interesting aspect of information content is that the counts associated with a synset are propagated up the hierarchy, in that when the child synset gets a count, the counts associated with its ancestors are incremented as well. Therefore the higher level concepts have higher counts associated with them. This leads to lower information content for general concepts and relatively higher information content for specific concepts.

To calculate the probability, the measures need a sense-tagged corpus to compute concept frequencies. If sensetagged text is not available the measures must adopt an alternative counting scheme. Resnik [36] suggests counting the number of occurrences of a word in a corpus and then dividing the number of different senses associated with that word type equally within the count. For example, as shown in Figure 2, *squash* has 4 senses. If *squash* occurs 10 times in the corpus, each sense of *squash* would get a frequency of $10/4$.

To handle the zero valued frequencies, WordNet::Similarity uses add-1 smoothing.

Resnik [37] proposed a similarity measure based on information content that computes the information content of the least common subsumer of the synsets s_1 and s_2

$$Sim_{res}(s_1, s_2) = IC(LCS(s_1, s_2)) \tag{7}$$

The similarity between synsets is based on how much information they share with each other. If they are sharing more specific information then the synsets are more related. The limitation of this similarity measure is that there are many concepts in WordNet that share the same least common subsumer. This results in assigning the same similarity score to all the concepts. This is more common for verbs, as verb hierarchies in WordNet are very shallow. Because of this, the measure is unable to make fine distinctions between two concepts and is considered as a coarse grained measure.

Jiang and Conrath [11] use information content to find semantic distance between concepts s_1 and s_2 in the noun hierarchy. The intuition behind the measure is that, for concepts that share a lot of information, the information content of the lowest common subsumer will be high. This will result in a smaller semantic distance between the concepts and the lowest common subsumer. The semantic distance defined by Jiang and Conrath is:

$$Dist_{jcn}(s_1, s_2) = IC(s_1) + IC(s_2) - (2 \times IC(lcs(s_1, s_2))) \tag{8}$$

The concepts with smaller semantic distance are more similar to each other than the concepts with a larger distance.

Thus the similarity can be described as

$$Sim_{jcn}(s_1, s_2) = \frac{1}{Dist_{jcn}(s_1, s_2)} \tag{9}$$

In 1998, Lin [15] proposed a measure based on information content which is similar to Jiang and Conrath's measure. Lin calculates the semantic similarity between the synsets s_1 and s_2 using the equation

$$Sim_{lin}(s_1, s_2) = \frac{2 \times IC(LCS(s_1, s_2))}{IC(s_1) + IC(s_2)} \tag{10}$$

The idea is that if two concepts share a lot of specific information, then the similarity score would be greater. If the concepts do not share much information, then the score will be lower. Note that the term *depth* used by Wu and Palmer in path based measures can also be thought of as a measure of specificity, in that the more deep the concepts are, the more specific they are. The measure developed by Wu and Palmer (wup) is considered as a special case of the Lin measure.

2.2.3 Gloss Based Measures

The similarity measures we have seen by now are limited to *is-a* relations in WordNet. Gloss based measures are relatedness measures and go beyond is-a relations. In 1986, Lesk proposed a solution to word sense disambiguation based on the overlaps between semantic relatedness between words [14]. He suggests that given a specific word from a text, the sense of that word could be identified by counting the number of overlaps in the definitions of that word and the definitions of the word preceding or following that word and the sense with maximum matches (overlaps) would potentially be the intended sense.

Based on this idea, a new measure of semantic relatedness based on extended gloss overlap (lesk) was then introduced by Banerjee and Pedersen [1]. The measure combines the advantages of Lesk's gloss overlap with the structure of a concept hierarchy to create an extended view of relatedness. The scoring mechanism is different than Lesk's scoring, since this measure doesn't differentiate between a single word and a phrasal overlaps. Given two glosses, the longest overlap between them is detected. Then the overlap is removed and a unique marker is placed in both strings. The process is continued in a recursive fashion until there is no overlap in the two strings. A phrasal overlap of

n words is assigned a score n^2. The summation of the squares of the lengths of individual overlaps is the score for the pair of glosses.

$$pairscore = \sum_{i=1}^{\#_{overlaps}} length^2(overlap_i) \tag{11}$$

where, $length^2(overlap_i)$ is the number of overlapping words in the i^{th} overlap.

For example, given synsets, $s_1 = cat\#n\#7$ and $dog\#n\#1$ and the overlaps $\{$*claws, fissiped mammals*$\}$ using hypernym-hypernym relation, the pairscore between s_1 and s_2 will be $1 + 2^2 = 5$.

A pairscore is computed for each relation pair. The relatedness score between synsets is then given by

$$Relatedness(s_1, s_2) = \sum_{j=1}^{\#_{relations}} pairscore_j \tag{12}$$

For the above example, if there exist overlaps for 2 more relation pairs, a pairscore of 20 for hyponym-hyponym relation pair, and a pairscore of 1 for hyponym-holonym pair, then the relatedness score between s_1, ands_2 would be $5 + 20 + 1 = 26$.

Another gloss based measure of relatedness is the context vectors measure (vector) proposed by Patwardhan [29]. In this measure, each synset is represented by a gloss vector where a gloss vector is a context vector formed by considering a WordNet gloss as the context. The semantic relatedness of two synsets is then computed by measuring the cosine of the angle between the corresponding normalized gloss vectors. A context vector is a sparse vector, which contains components denoting the co-occurrence frequencies of the word and the words in its glosses. For example, suppose we are finding the relatedness between s_1 = *cent#n#2* and s_2 = *dollar#n#3*. The gloss of *cent#n#2* contains the content words (words that are included in WordNet) *coin, worth, one-hundredth, value, basic, unit*. The context vector of the word *coin* which contains the content words *flat, metal, piece, used, usually, money* in its gloss may look like[6]

$$\overrightarrow{coin} = (0\ 2\ 0\ 3\ 0\ 1\ 0\ 1\ 0\ 5\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)$$

[6]In reality this vector is a very high dimensional sparse vector.

The higher frequencies mean that the words co-occur more number of times. A gloss vector is then created by adding all context vectors. For example, the gloss vector for $\overrightarrow{cent\#n\#2}$ will be

$$\overrightarrow{cent\#n\#2} = \overrightarrow{coin} + \overrightarrow{worth} + \overrightarrow{one\text{-}hundredth} + \overrightarrow{value} + \overrightarrow{basic} + \overrightarrow{unit}$$

Similarly a gloss vector for *dollar#n#3* is created and the relatedness between synsets s_1 and s_2 is computed as the cosine angle between the gloss vectors $\overrightarrow{s_1}$ and $\overrightarrow{s_2}$

$$Relatedness(s_1, s_2) = \frac{\overrightarrow{s_1}.\overrightarrow{s_2}}{|\overrightarrow{s_1}||\overrightarrow{s_2}|} \tag{13}$$

Vector measure is more general than the extended lesk measure in that it goes beyond finding exact string overlaps between glosses.

This thesis uses the measures of relatedness and similarity discussed above to solve the problem of all-words sense disambiguation. It is important to note that, for a WSD problem, the usefulness of WordNet based measures is limited by sparsity of its arcs. Therefore, it is important to understand the variations and limitations of these measures in order to understand the results presented in this thesis.

In the subsequent chapters, we'll discuss the algorithm and experimental results. We refer to our system as WN-SRAW (**W**ord**N**et **S**ense**R**elate **A**ll**W**ords). The similarity and relatedness measures will be referred to with the abbreviations in the parenthesis: Path Length (path), Wu and Palmer [45] (wup), Leacock and Chodorow [13] (lch), Hirst and St-Onge [9] (hso), Resnik [37] (res), Lin [15] (lin), Jiang and Conrath [11] (jcn), Extended Gloss Overlap [1] (lesk) and Gloss Vector [29] (vector).

2.3 Definitions

Monosemy

The problem of WSD exists because words are ambiguous. That said, in an arbitrary text, not all words are ambiguous. A significant number of words simply have one meaning. A monoseme is

a word or a phrase with a single meaning. For example, *friday* is a monoseme which has a single sense in WordNet – *the sixth day of the week; the fifth working day*.

Polysemy

Though most words in a dictionary are monosemous, it is the ambiguous words that occur in a text more often [23]. The term *polysemous* is defined as having or being characterized by multiple meanings. For example, *walk* is polysemous, in that it can mean *went walking, went for a walk, walk the dog, graduation walk*. Though there are multiple meanings associated with the word *walk*, they are more or less related. But in some cases, the words have same form but completely different meanings. For example, *squash* in Figure 2, has completely different meanings of *vegetable squash* and *game squash*. This is called homonymy, where the words take same form but have completely different senses.

Unfortunately, WordNet doesn't make any distinction between polysemy and homonymy and hence this thesis tackles homonyms and polysems the same way. As we use the term, polysemy simply means the property of having multiple senses, irrespective of whether they are related or completely different. If a word has a large number of senses in the sense inventory, we say that the word is highly polysemous. For example, the verb *make*, which has 49 senses in WordNet is highly polysemous.

Types and Tokens

How many unique words are there in the text below? We can see that the words *things, learn, the, you* are repeated several times. The unique words in a text are referred to as *types* or *word types* and the distinct words are referred to as *tokens* or *word tokens*. In the following text there are 19 word types and 25 word tokens.

> "The only things you learn are *the things you* tame," said *the* fox. "People haven't time to *learn* anything. They buy *things* ready-made in stores."

3 WN-SRAW Algorithm

Now that we know the problem of all-words sense disambiguation and have some background about WordNet and the measures of similarity and relatedness, it is time to discuss WN-SRAW approach for all-words sense disambiguation.

In 2003, Patwardhan et al. [30] proposed a method for lexical sample disambiguation based on the measures of semantic similarity and relatedness that are implemented in the WordNet::Similarity Perl package. The method relies on the available context and strictly uses information that can be automatically obtained from the lexical database WordNet, thus requiring no manually annotated data. This makes it an unsupervised knowledge based approach. In 2005, a generalization of this method which can be applied for all-words sense disambiguation was suggested by Michelizzi [19]. This thesis is based on the method described by Michelizzi [19].

Before looking at the algorithm, let us first define the basic units of language such as word, term and sentence as they are used in this chapter. **A word** is a unit of text in which only alphabetic characters, numerals and two special characters namely **'** and **-** are allowed. **A content word** is a word that is found in WordNet. For example, *friend* and *ice-cream* are content words. **A term** is two or more words that represent a single meaning. For example, *'Alan Turing'* and *'White House'* are terms. **A sentence** consists of words and terms and is delimited by a new line character.

WN-SRAW takes a sentence as input and outputs a sense tagged sentence. The input format can be either raw, wntagged or tagged. The raw format is the most generic format and is used for disambiguation of a plain text. WordNet tagged (wntagged) input, where a WordNet tag[7] is assigned to each content word is also supported. The tagged format refers to the Penn Treebank tagged [8] text. Though Penn Treebank tagged text is allowed, prior to disambiguation the tags are mapped to the four WordNet tags[9]. So in turn tagged text is converted to wntagged text before disambiguation and is treated as wntagged text thereafter.

Here are examples of each format.

[7] WordNet tags are n,v,a,r.

[8] http://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html

[9] Refer Appendix A.1 for the mapping.

1. (raw): Lake Superior is the largest of the five Great Lakes of North America.

2. (tagged) : Lake_Superior/NNP is/VBZ the/DT largest/JJS of/IN the/DT five/CD Great_Lakes/NNPS of/IN North_America/NNP.

3. (wntagged) : Lake_Superior#n is#v the largest#a of the five#n Great_Lakes#n of North_America#n[10]

Given a sentence in any of the above formats, WN-SRAW first converts the input text to lower case and follows the steps below sequentially.

3.1 Compoundify

If the format is raw, first the punctuation marks are removed and then WN-SRAW will identify WordNet **compounds**, which are the terms found in WordNet. WordNet represents compounds using the '_' character. *Lake_Superior* and *North_America* represent WordNet compounds in the above example. Compounds have non-compositional meaning. For example, *'White_House'* is a compound which consists of two distinct words, however, the whole term is recognized as a single noun sense. If all compounds in a sentence are identified then it is called **a compound identified sentence**.

If the format is raw, identifying compounds is essential. If words in a compound are considered separately, then it often becomes impossible to disambiguate. For example, if *red tape*, which means *needlessly time-consuming procedure* is considered as two separate words, disambiguation becomes impossible because no combination of senses of *red* and *tape* can represent the sense *needlessly time-consuming procedure*. Moreover, the algorithm will search for all various senses of *red* (7 senses in WordNet) and all various senses of *tape* (8 senses in WordNet) although it is guaranteed that the senses chosen would be wrong. In contrast, most compounds are monosemous and hence disambiguation consists of simply assigning the only available sense to the compound.

In WordNet 3.0, out of the 155,287 unique strings, more than 40% (64,331) are compounds. The longest compound in WordNet 3.0 is *prayer_of_azariah_and_song_of_the_three_children*, which consists of 9 words. Since there is no longer compound in WordNet, the compoundify algorithm only

[10]Since WordNet contains only open class words, words like *the, of* are not tagged.

looks for compounds of $1 < length < 10$, which helps improve the efficiency of the algorithm.

Note that for tagged and wntagged format, it is assumed that the text has no punctuation marks and compounds are already identified as shown in the *Lake_Superior* example above.

WN-SRAW uses WordNet::Tools[11] Perl module for compound identification. Algorithm 1 describes WordNet::Tools's algorithm to identify compounds in a sentence. Given a raw formatted sentence, compoundify identifies all the compounds and returns a compound identified sentence. For example, given sentence *evariste galois founded modern group theory*, the algorithm returns *evariste_galois founded modern group_theory*. The compounds are found using a greedy search method. The longest valid compounds in a sentence are chosen. For example, in the sentence *Sir William Walton was a British composer and conductor*, there are two compounds, *'Sir_William_Walton'* and *'William_Walton'*. The longest compound *'Sir_William_Walton'* will be chosen.

The compound identified sentence is further used for disambiguation.

3.2 Stop Words Removal

An arbitrary text consists of open class words (nouns, verbs, adjectives and adverbs) and closed class words (prepositions, determiners, conjunctions, pronouns, etc.). Closed class words (also called as function words) don't carry with them much meaning and therefore are not included in WordNet. We call these words **stop words**. WN-SRAW can only disambiguate content words (i.e. the words that are defined in WordNet), which is why many stop words are automatically excluded. However, some words that are typically used as stop words have unusual usages in WordNet. For example, WordNet defines the most commonly used stop words *an* and *who* as below.

1. an : Associate in Nursing, AN – (an associate degree in nursing)
2. who: World Health Organization, WHO – (a United Nations agency to coordinate international health activities and to help governments improve health services)

[11] http://search.cpan.org/ tpederse/WordNet-Similarity-2.05/lib/WordNet/Tools.pm

Algorithm 1 Compoundify

1: **function** Compoundify ($sentence$) : $compound\ identified\ sentence$
2: $MAX\_COMPOUND\_SIZE \leftarrow 9$
3: $compound\text{-}identified\text{-}sentence \leftarrow sentence$
4: $first\text{-}index \leftarrow 0$
5: $last\text{-}index \leftarrow$ number of terms in $sentence\ -\ 1$
6: **while** $first\text{-}index < last\text{-}index$ **do**
7: $end\text{-}compound\text{-}index \leftarrow first\text{-}index + MAX\_COMPOUND\_SIZE$
8: **while** $first\text{-}index < end\text{-}compound\text{-}index$ **do**
9: $candidate\text{-}compound \leftarrow$ sequence of words in $compound\text{-}identified\text{-}sentence$ separated by '_' from $first\text{-}index$ to $end\text{-}compound\text{-}index$
10: **if** $candidate\text{-}compound$ is a valid compound defined in WordNet **then**
11: replace respective multiple words in $compound\text{-}identified\text{-}sentence$ by $candidate\text{-}compound$
12: Goto 16
13: **end if**
14: $end\text{-}compound\text{-}index \leftarrow end\text{-}compound\text{-}index - 1$;
15: **end while**
16: $first\text{-}index \leftarrow end\text{-}compound\text{-}index + 1$;
17: **end while**
18: return $compound\text{-}identified\text{-}sentence$
19: **end function**

Here is the more complete list of stop words which are almost always used as function words, but also have WordNet senses associated with more unusual usages: a, an, as, at, by, i, in, it, he, his, me, oh, ok, or, thou, us, wa, who.

To eliminate the stop words that have unusual senses in WordNet, we employ a stoplist. A **stoplist** is a list of stop words. WN-SRAW's default stoplist includes the above stop words.

The stop words removal is done immediately after compoundification. This is done for all three formats. The stoplist is checked for each content word or compound in a compound identified sentence to see if it is a stop word. All the matching terms are marked as stop words and are not further considered for disambiguation. For example, assuming WN-SRAW is using the default stoplist, after stop words removal, the sentence *The movie_star married an astronomer* would be *The movie_star married astronomer*. Stop words removal is done after compoundification because some compounds contain stop words, for example the compound *blink_of_an_eye* contains the stop word an. The text in which all compounds are identified and stop words are removed is further used by WN-SRAW algorithm.

3.3 WordNet Interface and Lemmatization

WN-SRAW uses WordNet::QueryData[12] Perl module as an interface to the WordNet database files. WordNet::QueryData Perl module allows the user direct access to the full WordNet semantic lexicon. It supports all parts-of-speech and access is generally very efficient.

Lemmatization is the process of obtaining the base forms of a word. For example, *catch#v* is the lemma of *catching#v*. In a WSD problem, identifying the correct lemma of a word before disambiguation is essential. For example, in the sentence *flowers **are** blooming*, if *are* isn't lemmatized to *be*, then we probably get some funny senses for *are* like *a unit of surface area equal to 100 square meters*. There is no further possibility of correct disambiguation in such a case.

WN-SRAW uses simple lemmatization available within WordNet and QueryData provides an interface to do that. Given a word or word#pos, it provides a list of all alternate forms (alternate spellings, conjugations, plural/singular forms, etc.). For this, it uses a simple morphological pro-

[12] http://search.cpan.org/dist/WordNet-QueryData/

cessing as provided by WordNet to identify the base form of a word. For example, given the word *looks* it will return *look#n* and *look#v* simply as WordNet would return.

3.4 Disambiguation

WN-SRAW processes a text sentence by sentence. It expects one sentence per line and one line per sentence. The input of the algorithm is a stop words eliminated sentence which contains lemmatized content words and/or compounds. The distinct units in an input sentence of WN-SRAW will be referred to as *tokens* or *instances* and the unique units in an input sentence of WN-SRAW will be referred to as *types* or *word types*.

An example of an input sentence is shown below. The input sentence contains 8 tokens or instances and 8 word types. Henceforth the terms "instance" and "token" will be used interchangeably.

> Plain sentence: *Richard Phillips Feynman was an American physicist known for expanding the theory of quantum electrodynamics.*
>
> WN-SRAW input: *richard_phillips_feynman be american physicist known expand theory quantum_electrodynamics*

Each token in the input sentence is disambiguated separately, starting with the first token and working left to right. At each stage, the token being disambiguated is called the **target**, and the surrounding tokens form the **context window**. The size of the context is determined by the user and will be referred to as **window size**. A balanced context is chosen according to the window size n. A window size of n means that there are n tokens in the context window, including the target. The window context is chosen as $ceil((n-1)/2)$ tokens on the left and $floor((n-1)/2)$ tokens on the right of the target, where

$$ceil(x) = smallest\ integer\ not\ less\ than\ x\ as\ a\ real\ number$$
$$floor(x) = largest\ integer\ not\ greater\ than\ x\ as\ a\ real\ number$$

For example, in the above input sentence, if *known* is the target, then a window size of 3 determines the context as (*physicist, expanding*) and a window size of 4 determines context as (*american,*

physicist, *expanding*). Note that the tokens at the start or end of a sentence will have unbalanced windows associated with them, since the algorithm does not cross sentence boundaries and treats each sentence independently. For example, for the target *quantum_electrodynamics* with window size 3, only *theory* will be in the context window. If the window size is 2 and the target is the first token on the left, the first token on the right is considered. In the example above for window size 2 the context for the target *richard_phillips_feynman* will be the token *be*.

Each target is disambiguated as below.

Suppose w_t is the target having senses $\{s_1, s_2, ..., s_{m_t}\}$ and $c_1, c_2, ..., c_n$ are the tokens in a context window where the window size is $n+1$. Assume that each context token c_i has m_i possible senses, denoted as $\{s^*_{i1}, s^*_{i2}, ..., s^*_{im_i}\}$. The goal of the algorithm is to select one of the senses from the set $\{s_1, s_2, ..., s_{m_t}\}$ as the most appropriate sense for the target w_t.

WN-SRAW assigns the most appropriate sense to the target by measuring the semantic relatedness between the possible senses of the target and the possible senses of each of the tokens in the context window. Semantic relatedness is computed using the relatedness function, $relatedness : (s_k, s^*_{il}) \rightarrow \mathbb{R}$, where s_k ($1 \leq k \leq m_t$) represents k^{th} sense of the target and s^*_{il} represents l^{th} sense ($1 \leq l \leq m_i$) of the i^{th} ($1 \leq i \leq n$) context token and $\mathbb{R}$ is the set of real numbers. The $relatedness$ function takes as input two senses, and outputs a real number. It is assumed that this real number is indicative of the degree of semantic similarity between the two input senses. A larger number denotes high relatedness between the two senses and a smaller number denotes low relatedness between the senses. Denote by v_{ki} the relatedness which gives the maximum value for $(s_k, s^*_{il}), 1 \leq l \leq m_i$. For each context word c_i, this v_{ki} is assigned to s_k. For each s_k, we sum these values, giving $v_k = \sum_{i=1}^{n} v_{ki}$. In other words, v_k can be calculated as shown in Equation (14). The s_k with the greatest v_k is considered to be the most appropriate sense of the target w_t.

$$v_k = \sum_{i=1}^{n} \left(\max_{1 \leq l \leq m_i} (s_k, s^*_{il}) \right) \tag{14}$$

The pseudo code of the algorithm is described in Algorithm 2 and Algorithm 3. If v_{ki} doesn't meet a certain threshold[13], meaning that the target sense is not sufficiently related to the given context

[13]These thresholds can be set using –pairScore. By default it is 0.0.

token, then the score is not assigned. Similarly, if the maximum of v_k is below certain threshold[14], then WN-SRAW concludes that there is no sufficient relatedness found with the surrounding context to disambiguate the target.

Algorithm 2 Word Sense Disambiguation algorithm

1: **function** disambiguate-all-tokens ($input[\,], n$) : $disambiguated\text{-}input[\,]$
   {/*n is the window size and $input[\,]$ is a sentence made up of tokens*/}
2: **for all** tokens w_t in $input[\,]$ **do**
3: $best\text{-}sense \leftarrow$ disambiguate-single-token ($input[\,]$, t, n)
4: $disambiguated\text{-}input[t]$ = w_t with $best\text{-}sense$ assigned
5: **end for**
6: **return** $disambiguated\text{-}input[\,]$;
7: **end function**

3.4.1 WN-SRAW as a Complete Bipartite Graph

The algorithm can also be viewed as a weighted complete bipartite graph. A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V. A complete bipartite graph is a bipartite graph such that for $\forall u \epsilon U$, $\forall v \epsilon V$, an edge (u, v) exists.

Suppose w_t is the target. Let U and V be disjoint sets such that $U = \{s_1, s_2, ..., s_{m_t}\}$ is the set of m_t possible senses of token w_t and $V = \{(1, c_1), (2, c_2), ..., (n, c_n)\}$ is the set of n context tokens. The reason for using ordered pairs for context tokens is that they can repeat. For example, when disambiguating *visited,* in the input sentence *queen egypt visited queen england*[15], with window size=5, the context token *queen* would appear twice. Assume that each context token (i, c_i) has m_i possible senses, denoted as $\{s^*_{i1}, s^*_{i2}, ..., s^*_{im_i}\}$. The goal is to select one of the senses from the set $\{s_1, s_2, ..., s_{m_t}\}$ as the most appropriate sense for the target w_t.

Let E be the set of edges between U and V, i.e $E = \{(u, v) : u \epsilon U, v \epsilon V\}$. A real valued weight is

[14]This threshold can be set using –contextScore parameter. By default it is 0.0.

[15]The original text is *The queen of Egypt visited the queen of England*

Algorithm 3 Word Sense Disambiguation algorithm

function disambiguate-single-token ($input[\,]$, t, n) : $best\text{-}sense$

{/*w_t is the t^{th} token in the $input[\,]$*/}

$w_t \leftarrow input[t]$

{/*$context[\,]$ is an array of the context tokens of the current target w_t*/}

$context[\,] \leftarrow ceil((n-1)/2)$ tokens on the left and $floor((n-1)/2)$ tokens on the right of w_t

for all senses s_k of target w_t **do**

  $v_k \leftarrow 0$

  for all c_i in $context[\,]$ **do**

    for all s_j of c_i **do**

      $temp\text{-}score[j]$

    end for

    $v_{ki} \leftarrow$ highest score in array $temp\text{-}score[\,]$

    if $v_{ki} > pair\text{-}threshold$ **then**

      $v_k \leftarrow v_k + v_{ki}$

    end if

  end for

end for

{/*At this point each sense s_k of the target has a score v_k assigned to it.*/}

{/*The sense s_k with the maximum v_k is the winning score*/}

$winning\text{-}score \leftarrow$ score which has the maximum value v_k

if $winning\text{-}score > context\text{-}threshold$ **then**

  return s_k **which has the maximum value** v_k

else

  return no relatedness with the surrounding context

end if

end function

associated with every edge (u, v) which is calculated using the weight function W. W is calculated by taking the maximum of the relatedness scores between sense u of the target and each sense s^*_{il} of the context token $v \epsilon V$ as shown is Equation (15).

$$W(u, v) = \max_{1 \leq l \leq m_i} (relatedness(u, s^*_{il})) \tag{15}$$

where $relatedness$ is the function that takes two senses as input, and outputs a real number as defined previously. A low value of the weight function indicates that there is no relatedness or similarity between the sense and the context token. A high weight value means they are strongly related. Each node $u \epsilon U$ is assigned a score by summing the weights associated with its incoming edges. The sense with the highest score is the winning sense. All tokens are disambiguated one by one in this fashion.

3.4.2 Time Complexity of the Disambiguation Algorithm

It is interesting to see a WSD system running as a stand alone application. However, it becomes more useful when integrated in real-world applications such as Machine Translation and Information Retrieval. This makes it an intermediate task and hence it is expected to be time-efficient. This section discusses the time complexity of the algorithm.

Let s be the number of sentences in the input text. Let m be the average number of tokens for this set of input sentences. Let $n + 1$ be the window size and S_{avg} be the average number of senses of polysemous tokens. S_{avg} is calculated from the text to be disambiguated. For example, if the text has the two sentences below, where the parentheses gives the number of senses in WordNet for those tokens, S_{avg} is calculates as $S_{avg} = \frac{10+12+3+12+52+4}{6} = 15.5$

king(10) counts(12) money(3)
queen(12) plays(52) flute(5)

A comparison in this context means finding similarity/relatedness between two WordNet concepts. Let T_{sim} be the time required for a comparison.

For finding relatedness between each sense of the target and a context token $(S_{avg} \times S_{avg}) = S^2_{avg}$ comparisons are needed. Now for all context terms the number of comparisons is at most $n \times S^2_{avg}$.

We need to do these comparisons for all senses of the target. So for S_{avg} senses of the target, the upper limit on the total number of comparisons is

$$n \times S_{avg}^2 \times S_{avg} = n \times S_{avg}^3 \tag{16}$$

We need to do this for all m terms in the sentence

$$n \times S_{avg}^3 \times m \tag{17}$$

Finally, for s sentences in a text, the upper bound on the total number of comparisons can be given as,

$$n \times S_{avg}^3 \times m \times s \tag{18}$$

and the upper bound on the time required by the algorithm can be described as

$$n \times S_{avg}^3 \times m \times s \times T_{sim} \tag{19}$$

Equation (18) gives the upper bound on the total number of comparisons. It assumes that for all words in the text same number of comparisons are performed. In fact, the border cases will have an unbalanced window and hence fewer number of comparisons will be performed.

Though S_{avg}^3 is going to affect the number of comparisons of the algorithm, it is going to be a constant. So from Equation (18), we can say that the complexity is linear in $input\ size \times\ window\ size$. That is the time complexity depends upon the size of the text to disambiguate and the window size. That said, since WN-SRAW doesn't cross sentence boundaries, the upper bound on n will be $2 \times average\ number\ of\ content\ terms\ in\ a\ sentence = 2 \times m$. So the values of n greater than this upper bound will not have any substantial effect on the complexity.

4 Experimental Data

The performance of all-words sense disambiguation systems is evaluated using manually sense-tagged corpora. As noted in the introduction, annotating all words in a text is a hard and time consuming task. The large number of words to annotate, inconsistencies in the annotation and very fine distinctions between senses in the sense inventories mean it is also an error prone task. In order to get accurate annotation, two or more annotators work on the same text. In this process, the annotators might disagree on certain instances. The agreement between annotators is referred to as *inter-annotator agreement*. Where the inter-annotator agreement is high, the accuracy of annotation is correspondingly high. There are various methods to resolve the disagreement between annotators. In some methods, a third person looks at points where the annotators disagree and chooses one of the senses they chose or assigns a totally different sense. Another method lists all the possible annotations in the sense-tagged corpora.

For example, consider the following senses of the verb *deal*, in which WordNet makes a fine distinction between senses.

1. cover, treat, handle, plow, deal, address – (act on verbally or in some form of artistic expression; “This book deals with incest”; “The course covered all of Western Civilization”; “The new book treats the history of China”)

2. consider, take, deal, look at – (take into consideration for exemplifying purposes; “Take the case of China”; “Consider the following case”)

The senses are very close and for the following sentence both of the above annotations are appropriate for the verb *deal*. In such a case, it would be hard for annotators to decide upon the correct sense.

> *The examination asks students to **deal** with problems in calculus.*

The annotators may disagree and the sense-tagged corpus would list both of the above senses for the verb *deal*. The sense-tagged corpora we use follow this method to resolve inter-annotator dis-

agreement. While evaluating, we consider our answer correct whenever it matches any of the listed senses.

The experiments in this thesis were carried mainly on the following corpora.

4.1 SemCor

SemCor [24] is the most widely-used freely available manually sense-tagged corpus. It comprises around 234,000 semantically annotated tokens (80% Brown corpus, 20% a novel, "The Red Badge of Courage"). The Brown Corpus was created at Brown University in 1964. It includes news articles, fiction, religious works, and scientific writings.

In SemCor, all open class words are manually sense-tagged by WordNet 1.6 senses. Since version 1.6, WordNet has undergone many changes and released newer versions. Many senses were added and others removed in newer versions. This thesis uses WordNet 3.0. So in order to use SemCor, a mapping from WordNet 1.6 senses to WordNet 3.0 senses is required. We use the version of SemCor[16] that is re-mapped to WordNet 3.0 by Rada Mihalcea[17]. In this version of SemCor, the senses that are defined in WordNet 1.6 but are not defined in WordNet 3.0 are assigned sense 0.

Figure 4 shows a sample of SemCor formatted data of the following plain text.

The petition listed the mayor's occupation as "attorney" and his age as 71.

The lemmas with $wnsn > 0$ are sense-tagged tokens. As noted before, the input of WN-SRAW contains only content words and hence *the* doesn't appear in the input text. Here is the reformatted text that will be input to WN-SRAW system that includes only sense-tagged tokens.

petition#n list#v mayor#n occupation#n attorney#n age#n

In this version of SemCor, there are overall 185,273 open class sense-tagged tokens along with their context.

[16] http://www.cs.unt.edu/ rada/downloads.html

[17] Unless specified otherwise we use SemCor 3.0

The petition listed the mayor’s occupation as “attorney” and his age as 71.

```
</s>
</p>
<p pnum=28>
<s snum=32>
<wf cmd=ignore pos=DT>The</wf>
<wf cmd=done pos=NN lemma=petition wnsn=1 lexsn=1:10:00::>petition</wf>
<wf cmd=done pos=VB lemma=list wnsn=1 lexsn=2:32:00::>listed</wf>
<wf cmd=ignore pos=DT>the</wf>
<wf cmd=done pos=NN lemma=mayor wnsn=1 lexsn=1:18:00::>mayor</wf>
<wf cmd=ignore pos=POS>ś</wf>
<wf cmd=done pos=NN lemma=occupation wnsn=1 lexsn=1:04:00::>occupation</wf>
<wf cmd=ignore pos=IN>as</wf>
<punc>“</punc>
<wf cmd=done pos=NN lemma=attorney wnsn=1 lexsn=1:18:00::>attorney</wf>
<punc>”</punc>
<wf cmd=ignore pos=CC>and</wf>
<wf cmd=ignore pos=PRP$>his</wf>
<wf cmd=done pos=NN lemma=age wnsn=1 lexsn=1:07:00::>age</wf>
<wf cmd=ignore pos=IN>as</wf>
<wf cmd=done pos=CD ot=notag>71</wf>
<punc>.</punc>
</s>
```

Figure 4: SemCor formatted data

corpus	nouns	verbs	adjectives	adverbs
SemCor	87,002 (47%)	47,570 (26%)	31,754 (17%)	18,947 (10%)
SENSEVAL-2	1,057 (47%)	509 (23%)	417 (18%)	277 (12%)
SENSEVAL-3	884 (46%)	719 (37%)	322 (17%)	12 (0.6%)

Table 2: The number of tokens broken down by part-of-speech where the token is defined in WordNet.

corpus	# tokens	# word types
SemCor	18,5273	21,513 (12%)
SENSEVAL-2	2,260	1,075 (48%)
SENSEVAL-3	1,937	952 (49%)

Table 3: Overall number of tokens and word types.

corpus	nouns (%)	verbs (%)	adjectives (%)	adverbs (%)	All (%)
SemCor	18.9	5.2	30.7	39.3	21.32
SENSEVAL-2	22.8	2.3	19.4	37.2	20.54
SENSEVAL-3	19.1	5.3	21.7	100	16.53

Table 4: Percentage of monosemous tokens per part-of-speech.

semcor	SENSEVAL-2	SENSEVAL-3
be#v (8,400)	gene#n (60)	be#v (137)
person#n (6,696)	cancer#n (54)	man#n (17)
not#r (1,703)	ringer#n (27)	have#v (16)
group#n (1,329)	say#v (24)	say#v (15)
have#v (1,126)	bell#n (22)	local#a (13)
say#v (1,005)	not#r (21)	feel#v (12)
location#n (993)	cell#n (21)	state#n (12)
make#v (757)	copy#n (17)	stranger#n (11)
man#n (576)	educational#a (16)	time#n (11)
see#v (549)	know#v (16)	legislator#n (11)
know#v (512)	find#v (16)	voter#n (11)
time#n (511)	education#n (15)	congressional#a (11)

Table 5: First n most frequent word types where the type frequency for frequently occurring word types in SemCor > 500.

4.2 SENSEVAL/SEMEVAL

SENSEVAL[18](now renamed SEMEVAL) is an international competition on WSD organized by the Association for Computational Linguistics (ACL) **S**pecial **I**nterest **G**roup on the **LEX**icon (SIGLEX). The goal of the competition is to evaluate the strengths and weaknesses of various WSD systems with respect to different words, different varieties of language, and different languages. The review of the competitions can be found in [28]. The competition includes a number of different tasks, and the SENSEVAL organization develops a set of test data to use for evaluating the systems. There have been four competitions held so far. The first one, SENSEVAL-1 took place in 1998 and consisted of a lexical-sample task. The second competition SENSEVAL-2 took place in 2001 and consisted of a lexical-sample task, an all-words task and a translation task. SENSEVAL-3 was held in 2004 and consisted in total of 14 tasks including lexical-sample and all-words tasks. The fourth edition of SENSEVAL, SENSEVAL-4/SEMEVAL-2007 consisted of total 19 tasks including an all-words coarse-grained WSD task. The fifth competition is planned for 2010.

There were several different types of data sets created for all-words tasks in these competitions. The SENSEVAL data[19] used in this thesis is the sense annotated data used for the English SENSEVAL all-words task. The data sets have the same format as of SemCor.

SENSEVAL-2 is a small subset of the Penn Treebank corpus[20] and consists of 3 Wall Street Journal articles. 2,473 of the total 4,873 words are open-class words, and 2,260 of the open-class words are found in WordNet. Table 2 shows the most frequently occurring word types in this corpus. Looking at the most frequently occurring word types *gene, cancer, cell,* we can guess that the corpus is medicine related. Still it shares common words like *know, not* and *say* with SemCor.

Like SENSEVAL-2, SENSEVAL-3 also is a small subset of the Penn Treebank corpus and consists of 3 articles. Two of them are sections of Wall Street Journal articles and one is a work of fiction from the Brown corpus. Out of 4,883 words in the set, 2,081 are open-class words and 1,937 of the open class words are found in WordNet. The inter-annotator agreement was 72.5% for people with advanced linguistics degrees. Table 2 demonstrates that SENSEVAL-3 shares a large number of the

[18]http://www.senseval.org/

[19]http://www.cs.unt.edu/ rada/downloads.html

[20]http://www.cis.upenn.edu/ treebank/

most frequent tokens with SemCor, e.g. *be, man, have, say* and *time*.

Table 2 shows the distribution of the open class words in SemCor, SENSEVAL-2 and SENSEVAL-3 data sets by part-of-speech. Only words with valid WordNet senses are included in the table. It can be seen that SENSEVAL-3 has a negligible percentage of adverbs and a somewhat higher percentage of verbs. This leads to higher percentage of polysemous tokens in SENSEVAL-3.

Table 3 shows the number of tokens and word types in the data sets. It can also be seen that because of the size of SemCor the number of word types in SemCor is pretty small (12% of the total SemCor tokens) compared to the number SENSEVAL-2 and SENSEVAL-3 types (around 48%).

Table 4 shows the proportion of monosemous tokens per part-of-speech. It shows that a significant percentage of adverbs in SemCor and all adverbs in SENSEVAL-3 are monosemous. However the overall percentage of monosemous words in SENSEVAL-3 is less.

Table 5 shows the frequently occurring types in the datasets. For SemCor, we created a cut off of frequency $= 500$. We observed that first 12 frequently occurring word types in SemCor have frequency > 500. Then we chose first 12 frequently occuring types from SENSEVAL-2 and SENSEVAL-3.

The WN-SRAW system was not evaluated on any data from Senseval-1 since there was no all-words task. The all-words coarse-grained WSD task data from the SEMEVAL-2007 competition would provide interesting data. However, this thesis doesn't use it.

5 Experimental Results

This chapter presents an extensive set of results from the experiments carried out on the data introduced in the previous chapter. At first, we discuss the general methodology of experiments and the baselines considered. Then we present a list of hypotheses and the measures used to evaluate the performance, followed by detailed discussion of each hypothesis.

The performance of WN-SRAW system is evaluated using sense-tagged corpora. The general methodology to evaluate results involves three steps. In the first step, the key (the gold standard) is extracted from the sense-tagged corpus. In the second step, part-of-speech tagged text is extracted from the same corpus ignoring sense tags. WN-SRAW is then used to disambiguate the extracted text. Finally, the answers of WN-SRAW are scored against the key.

We consider two baselines – random baseline and sense1 baseline. The random baseline is the assignment of a random sense to each instance. It serves as the lower bound of the algorithm. The random baseline randomly guesses a sense from the set of possible senses for each token and serves as a sanity check. The random baseline may be viewed as a series of dice throws. For each instance we have an N-sided die, where N is the number of senses. Then we assign the sense that is showed up on the die. This is done after lemmatization which leaves a comparatively small set of senses from which to choose a random sense. Moreover, for the parts-of-speech which are not highly polysemous (adjectives and adverbs), since there are not many choices from which to select a random sense, it is more likely that a randomly chosen sense is correct.

The sense1 baseline assigns sense1 from WordNet to each instance. It is common that dictionary makers (lexicographers) often try to organize senses so that the more common ones are more visible or obvious. This is discussed more in Miller, 1994 [23]. Using the same idea, WordNet lists the senses of a word according to their frequencies which are calculated from SemCor. The senses with high frequencies appear at the top. For example, for the noun *bank*, if *sloping land besides a body of water* sense occurs 25 times in SemCor, and *financial institution* sense occurs 20 times, WordNet will list the *sloping land* sense first[21]. In case of WordNet, this most common sense is referred to as

[21] Some words in WordNet do not have frequencies as they don't appear in SemCor (e.g. *ringer*). In that case the senses are listed in a random order.

sense1.

In a text, for many words the distribution of senses is highly skewed. Supervised methods do very well at least in part because they can learn the sense distribution from the training data and can make predictions based on that. Like supervised methods, the sense1 scheme also makes use of the knowledge of sense distribution of SemCor and this simple idea turns out to be very effective which makes it hard to outperform the sense1 scheme for all-words sense disambiguation systems. If each instance in SemCor is assigned sense1, we get a very high accuracy of 75%, which is very hard to achieve for most of the disambiguation systems. Interestingly, the sense1 scheme also performs well on SENSEVAL-2 ($\approx 66\%$) and SENSEVAL-3 ($\approx 67\%$). This raises the question – why even care about other WSD approaches if sense1 scheme performs so well. An important point to note here is that the sense1 scheme doesn't make use of the surrounding context for disambiguation and simply relies on the statistical information about sense frequencies. Consequently though it works well for SemCor and corpora containing related topics, it would not generalize well for texts in other domains. The most common sense of a word depends upon the domain of a text. For example, if *building material* sense of the instance *cement* is common in SemCor, it will be listed first in WordNet which won't help while disambiguating a text in dentistry domain. Testing on such corpora might show the unreliability of sense1 scheme. To make the most frequent sense scheme work effectively for texts in different domains, it is required that a sense-tagged corpora in that domain is available. As we noted before, this is an expensive and time consuming task.

It is important to note that WN-SRAW doesn't use sense frequency information for disambiguation. It treats all senses of a word as equally likely. Therefore, WN-SRAW could be easily used for different domains without requiring any sense-tagged corpora from that domain.

Now that we know the general methodology of experiments and know about the baselines, we will present a list of hypotheses for which we designed experiments.

1. If the context window around a polysemous target token is expanded, there will be more related tokens available to measure against that target, which will lead to a more accurate disambiguation.

2. If an all-words sense disambiguation system is not using any frequency count information,

it will show the same performance on instances where sense1 is not correct as on overall polysemous instances.

3. The degree of difficulty in disambiguating a token is proportional to the number of senses of that token (polysemy).

4. A significant percentage of word sense disambiguation error is caused by just a few highly frequent word types.

5. Part-of-speech tagged text will be disambiguated more accurately than raw text.

6. Given any two parts-of-speech, the more polysemous will be less accurately disambiguated.

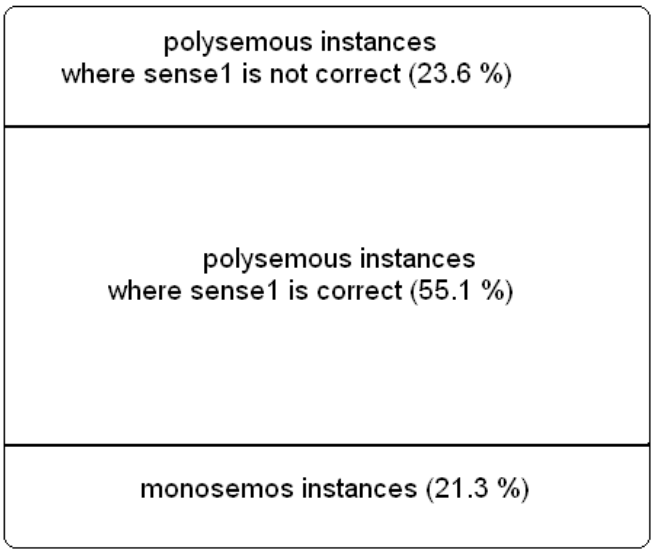

Figure 5: Instance space of the all-words sense disambiguation, showing proportion of instances in SemCor. Total Number of instances = 185,273.

Figure 5 shows the instance space of the all-words sense disambiguation problem. As we can see, for SemCor, 21.3% of the instances are monosemous, which are very easy to disambiguate, while the remaining 78.8% are polysemous, which are challenging to disambiguate. The difficulty of disambiguation depends upon where the instance falls in the instance space. Therefore to know where exactly the error lies, it is required to evaluate an instance based on its level of difficulty. For that, we

partition the instance space mainly into two classes. The first class contains monosemous instances and the second one contains polysemous instances. Again considering the difficulty of outperforming the most frequent sense (sense1 in case of WordNet), we further divide the class of polysemous instances into two classes – polysemous instances where sense1 is correct and polysemous instances where sense1 is not correct.

We introduce the following scoring options to evaluate the instances based on their difficulty level.

1. –score poly of the scorer program[22] of WN-SRAW system: Disambiguation of a monosemous instance involves assignment of the only available sense to the instance. This leads to very easy disambiguation guaranteeing 100% accuracy on those instances. However, unfortunately, it is the polysemous instances that occur in a text more often. For example, in the previous chapter we noted that SemCor and SENSEVAL-2 have around 79% polysemous instances while SENSEVAL-3 has about 83% polysemous instances. To see how well an all-words sense disambiguation system is performing, it is required to see its performance on only polysemous instances in a text.

   The option –score poly of the scorer program of WN-SRAW system allows to evaluate only polysemous instances in a text.

2. –score s1nc using scorer program[23] of WN-SRAW system: As discussed before, sense1 results are very high for the available sense-tagged corpora. So if the system is assigning sense1 by mistake or as a fall back strategy, it is easy to get misleading results. This option allows to score only polysemous instances where sense1 is not correct and hence avoids the possibility of getting misleading results. Other relevance of this scoring option is to show if there is any sense1 bias associated with a particular similarity/relatednes measure. For example, it is said that the first gloss in WordNet tends to be longer which could possibly create a sense1 bias for lesk.

   Note that the instances that will be evaluated using this option will be polysemous and will be contained in the instances evaluated in –score poly option.

[22] http://search.cpan.org/ tpederse/WordNet-SenseRelate-AllWords-0.19/utils/allwords-scorer2.pl

[23] http://search.cpan.org/ tpederse/WordNet-SenseRelate-AllWords-0.19/utils/allwords-scorer2.pl

3. –usemono option of the WN-SRAW system assigns the only available sense to the monosemous instances. As we noted, an arbitrary text contains a significant proportion of monosemous instances. Therefore, disambiguating only monosemous instances also results in a respectable accuracy. We used this option to see if the performance of WN-SRAW is only because of the easy monosemous instances. The trends of the results will be similar to –score poly, with a little boost in the overall performance because of the inclusion of monosemous instances.

The results are reported using *precision, recall,* and the *F-score. Precision* is the number of instances assigned correct senses divided by the total number of instances attempted by the system, and *recall* is the number of instances assigned correct senses divided by the total number of sense tagged instances in the corpus. Precision tells how well the system doing in the *attempted* instances and recall tells how well the system is doing over *all* instances. The F-score combines precision and recall as shown in equation (20), where p is precision and r is recall.

$$F = \frac{1}{\alpha\frac{1}{p} + (1-\alpha)\frac{1}{r}}, 0 \leq \alpha \leq 1 \tag{20}$$

The constant α determines how precision and recall should be weighted. $\alpha = 0.5$ will give an equal weight to p and r, while $\alpha > 0.5$ would give more weight to p. In this thesis same weight to p and r is given and hence $F\text{-}score$ is calculated using equation (21).

$$F = \frac{2pr}{p+r} \tag{21}$$

Sometimes the similarity and relatedness measures are unable to attempt instances because there is no relatedness found with the surrounding context or there are no instances in the context with which to find relatedness. This is very common in case of similarity measures because they can only be applied for *is-a* relations in WordNet. For example, in the following input sentence, a similarity measure will not be able to disambiguate the verb *eat* because there is no other verb in the sentence to find relatedness with[24].

[24]WN-SRAW has an option to coerce the part-of-speech of surrounding words to that of the target, although we haven't experimented much with that. http://search.cpan.org/dist/WordNet-SenseRelate-AllWords/doc/README.pod#Part_of_Speech_Coercion

eat#v dinner#n fancy#a restaurant#n

The instances for which the measure is able to assign answers, is what determines the coverage of the measure. In general, similarity measures have low coverage because they can only be applied to *is-a* hierarchies, more specifically for noun-noun and verb-verb pairs. On the other hand, relatedness measures can be applied for all parts-of-speech and can exploit all relations, resulting in high coverage. This relates to the recall concept we introduced, in that recall measures the performance over all instances.

Now we present our analysis of each hypothesis. Details of each experiment will be discussed as results of that experiment are presented. Unless specified otherwise, all results in this chapter are presented in terms of F-score. Precision and recall results along with the timing information[25] are included in Appendix A.3.

[25]The machine used to run these experiments is a Linux system using kernel 2.6.24. It has 4 Intel(R) Xeon(R) CPU, 2.93GHz processors each with 4 cores, and has a total memory of 3,635,528 KB ($\approx 32GB$).

5.1 Hypothesis 1: If the context window around a polysemous target token is expanded, there will be more related tokens available to measure against that target, which will lead to a more accurate disambiguation.

Experiment 1: Evaluated only polysemy instances using –score poly option by expanding window size. Six similarity measures namely path, wup, lch, res, lin, jcn and two relatedness measures[26] lesk and vector as they are implemented in WordNet::Similarity package were used.

Observations and Analysis: The intuition behind this hypothesis is that the words that are far apart might be strongly related with the target and can give clues for disambiguation. For example, consider the following sentence with *instrument* as the target.

> original: *Sitar is a wonderful instrument.*
> WN-SRAW input (raw format): *sitar be wonderful* ***instrument***

Considering the strong connection between *sitar* and *instrument,* the algorithm should choose *musical device* sense of the target. But, for a smaller window size, say window=3, *sitar* won't be in the context of the target and therefore similarity and relatedness measures will not be able to find the intended sense.

On the other hand, for a bigger window size, say, window=7, *sitar* will be in the context which would lead to choose the *musical device* sense of the target. The intuition is that, expanding window should increase the precision. In terms of recall, if there are more tokens in the context, the chance of finding relatedness with at least one them is higher and hence increased window size would lead to a higher recall.

The results show increase in recall as per our intuition which in turn results in increased F-score, however, precision doesn't increase with increased window size.

Figures 6, 7 and 8 show F-score results which demonstrate that for all similarity measures, F-score increases with increased window size. In case of relatedness measures, a very slight increase in F-score is observed. All three corpora SemCor (Figure 6), SENSEVAL-2 (Figure 7) and SENSEVAL-3

[26] Due to performance constraints vector_pairs and hso are not used

(Figure 8) show similar trend.

The precision results in Figures 9, 10 and 11 show that though we expected precision to increase with bigger window sizes, it didn't. In fact for the similarity measures, it dropped down. This might be because more context words also lead to more noise which can mislead the algorithm.

The recall results in figures 12, 13 and 14 show that recall consistently increased with increased window size. Especially, expanding window helped in case of low recall similarity measures.

These results show some disagreements with results reported by Michelizzi [19]. Michelizzi concluded that a small window size tends to result in high precision but low recall. He observed especially good results for window size 2. This is because for window size=2, Michelizzi assign sense1 to the first instance in a sentence. For example, in the sentence below, if window size=2 and the target is *blue*, since there is no word in the left context, Michelizzi assigns sense1 to the target. This induces huge sense1 bias resulting in good precision for window size = 2.

WN-SRAW input (raw format): ***blue*** *mountain be look beautiful*

Without relying on sense1, in such a case, WN-SRAW considers right context word[27]. For the above example, WN-SRAW will have *mountain* in the context while disambiguating the target *blue*.

Conclusion: The hypothesis implies that expanding window context should lead to a more accurate disambiguation but the precision results show the opposite.

[27]This was changed effective version 0.17 of WordNet::SenseRelate::AllWords

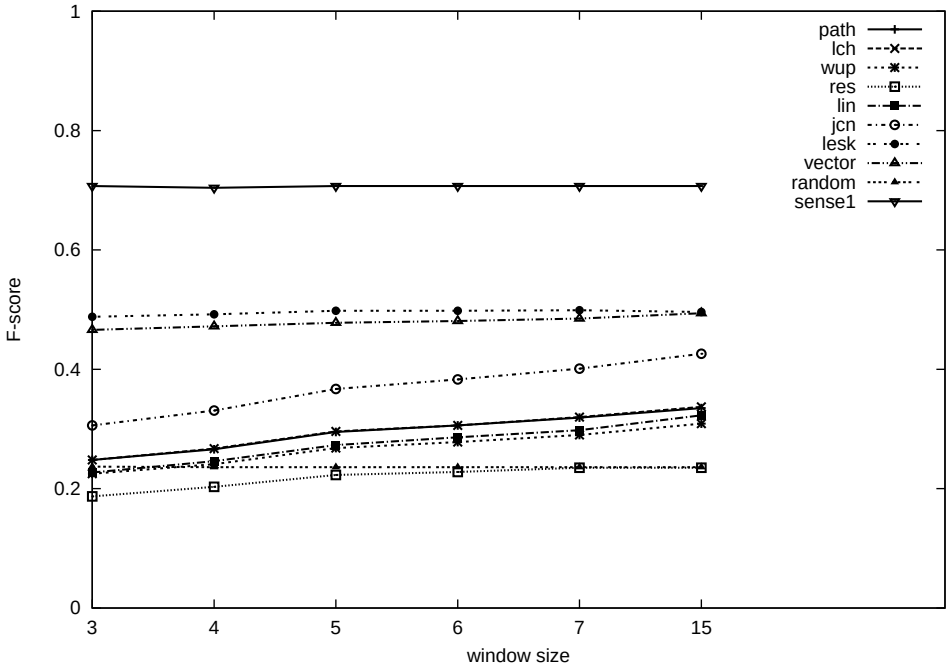

Figure 6: SemCor F-score results with –score poly option. Number of instances = 145,773.

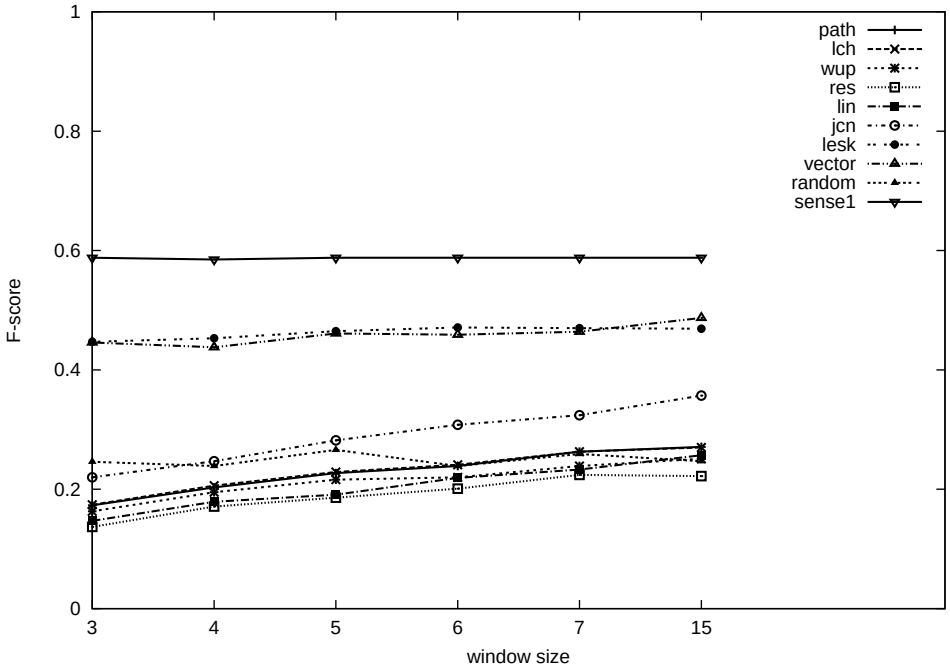

Figure 7: SENSEVAL-2 F-score results with –score poly option. Number of instances =1,796.

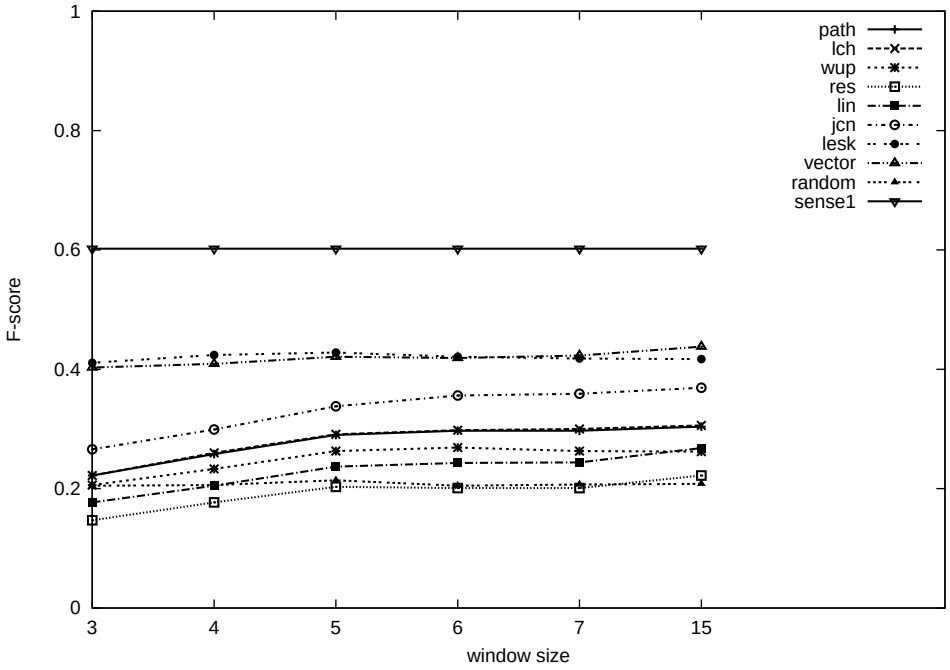

Figure 8: SENSEVAL-3 F-score results with –score poly option. Number of instances =1,617.

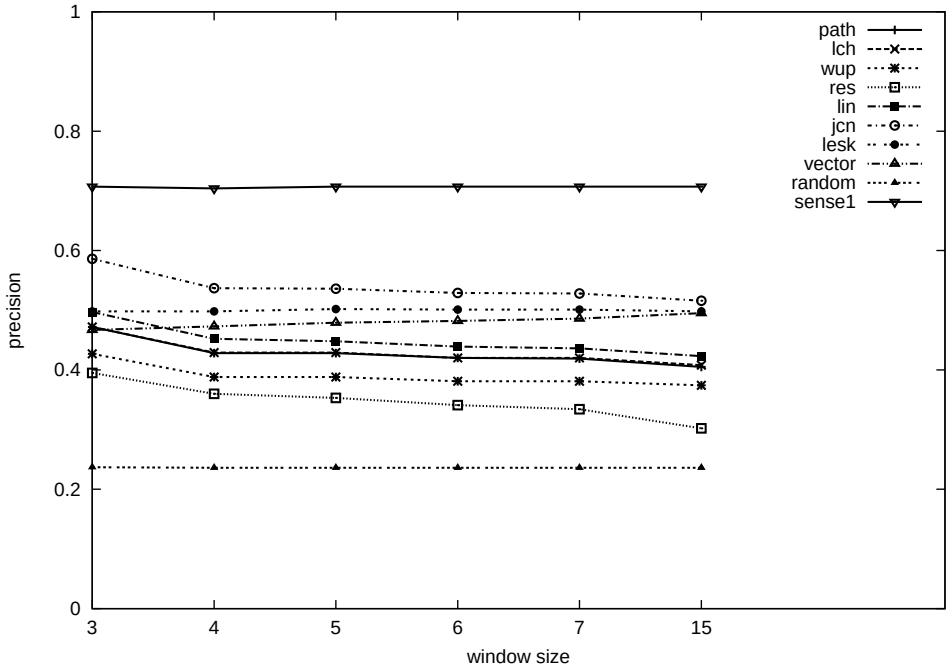

Figure 9: SemCor Precision results with –score poly option. Number of instances = 145,773.

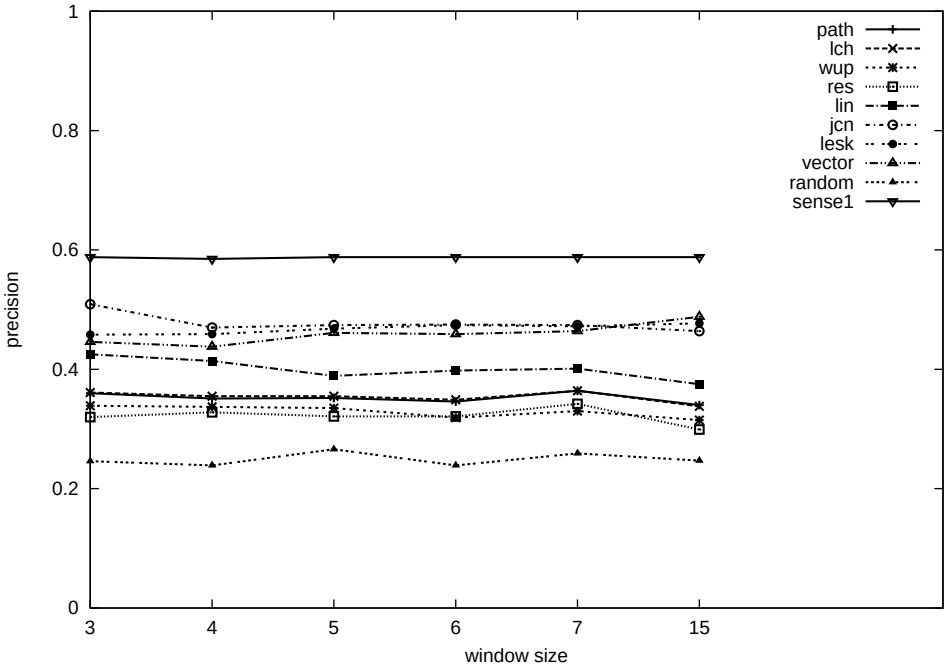

Figure 10: SENSEVAL-2 precision results with –score poly option. Number of instances =1,796.

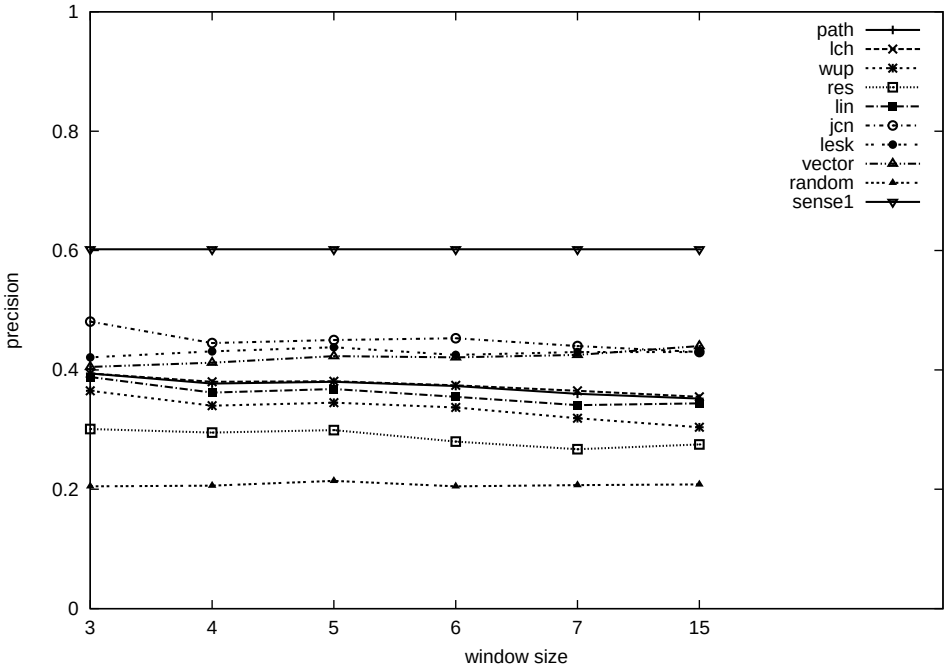

Figure 11: SENSEVAL-3 Precision results with –score poly option. Number of instances =1,617.

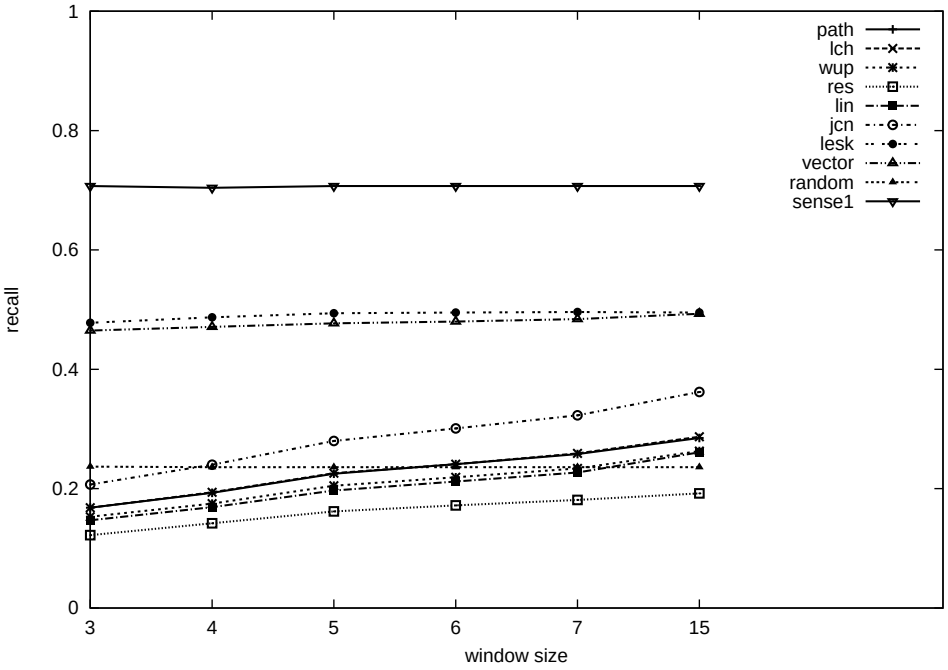

Figure 12: SemCor Recall results with –score poly option. Number of instances = 145,773.

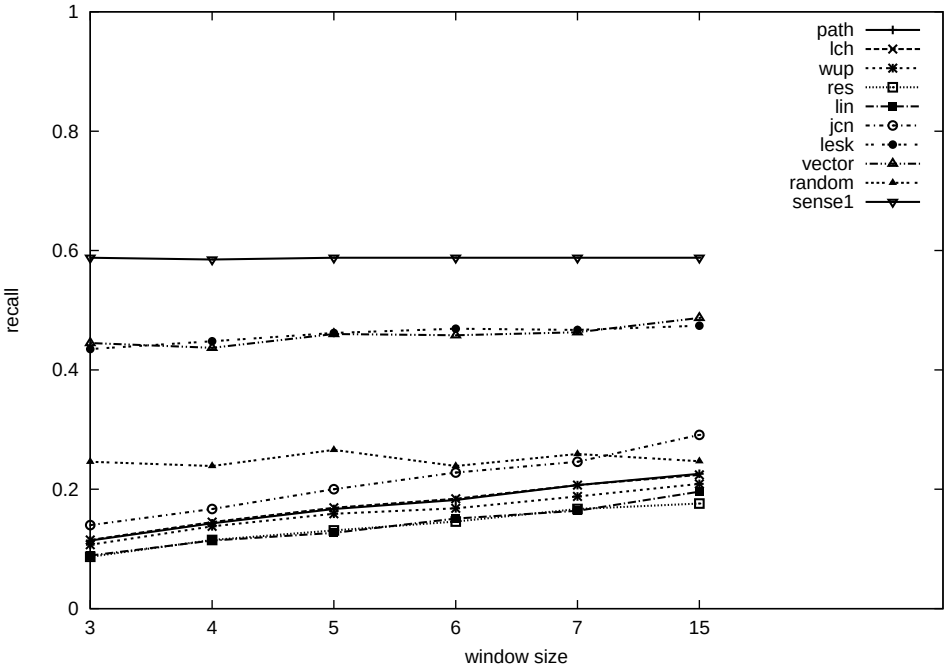

Figure 13: SENSEVAL-2 Recall results with –score poly option. Number of instances =1,796.

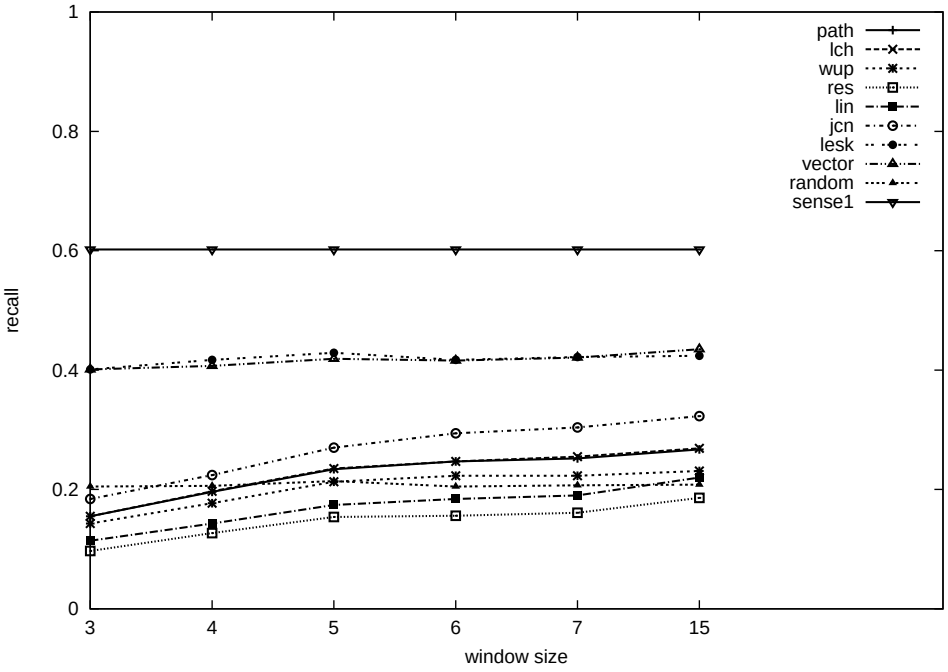

Figure 14: SENSEVAL-3 Recall results with –score poly option. Number of instances =1,617.

5.2 Hypothesis 2: If an all-words sense disambiguation system is not using any frequency count information, it will show the same performance on instances where sense1 is not correct as on overall polysemous instances.

Experiment 2: We evaluated the polysemy instances where sense1 is not correct using –score s1nc option by expanding window size. Six similarity measures namely path, wup, lch, res, lin, jcn and two relatedness measures lesk and vector as they are implemented in WordNet::Similarity package were used.

Observations and Analysis: As noted before, the sense1 heuristic works pretty well for the corpora used in this thesis. For about 76% instances in SemCor sense1 heuristic gives the correct answer. But what about remaining 24% instances?

Unlike sense1 heuristic, WN-SRAW doesn't rely on frequency count information. Instead it considers all senses of a word as equally likely. So naturally we expected that WN-SRAW will give same performance on the instances where sense1 heuristic doesn't work as on overall polysemous instances.

The results in Figure 15, 16 and 17 however show that the results drop down for the instances where sense1 is not correct. The performance lowers by a large margin compared to the performance of all polysemous instances. WN-SRAW achieves the best F-score of 0.499 on polysemous instances while the best F-score of 0.219 for the polysemous instances where sense1 is not correct. This indicates that if the answer is not sense1, WSD turns out to be a harder problem. These results also show that the polysemous instances where sense1 is not correct contribute significantly to the overall error.

The trend of the results is same as that of Experiment 1. All measures other than lesk and vector are performing worse than the random baseline for smaller window sizes. This is because of the very low recall of similarity measures. It can be seen that these measures converge to random baseline for bigger window sizes.

The reason for overall low results is not so clear. But our intuition is that some of the similarity and relatedness measures (especially lesk) have sense1 bias associated with them. In case of WordNet,

generally the first sense refers to the most common sense. It is believed that first senses in WordNet tend to be longer, which may be because lexicographers try to make it as clear as possible by putting as much information they have. This would create a bias for lesk because longer glosses increase the probability of finding more overlaps.

Conclusion The experimental results do not support the hypothesis.

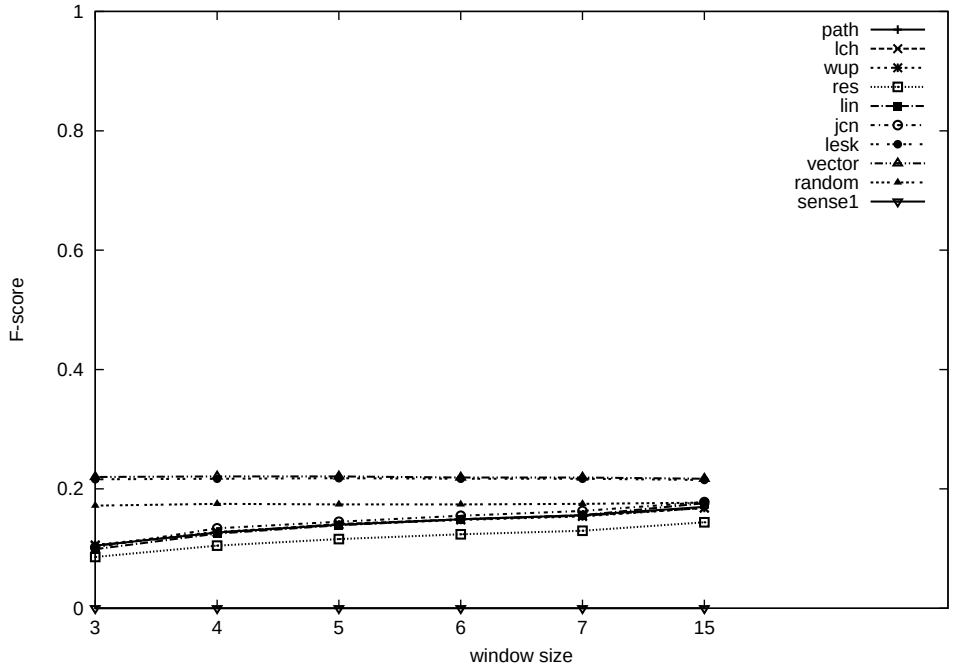

Figure 15: SemCor results with –score s1nc option. Number of instances = 43,730.

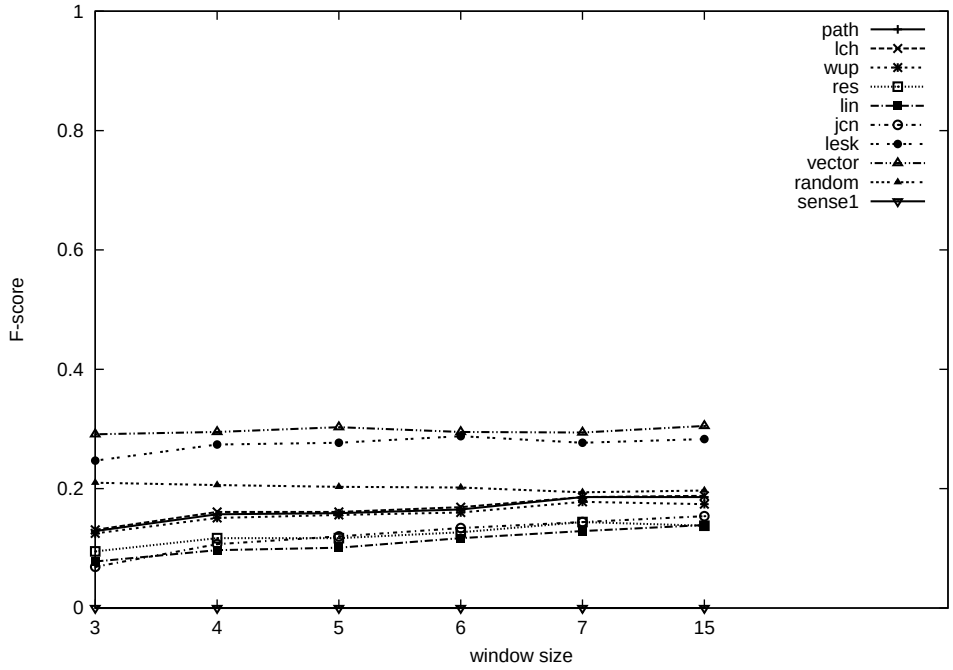

Figure 16: SENSEVAL-2 results with –score s1nc option. Number of instances =752.

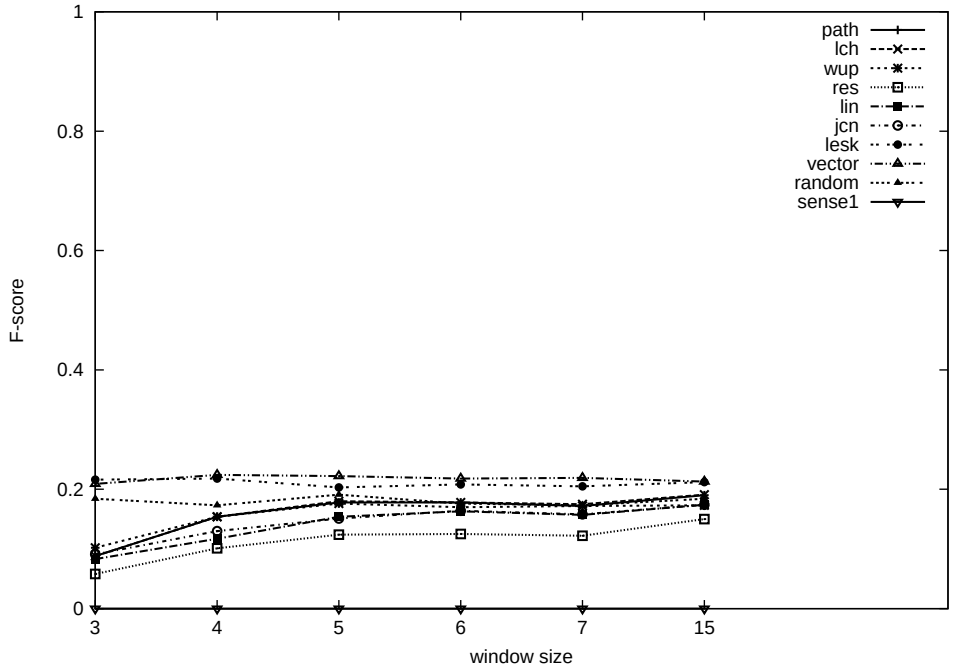

Figure 17: SENSEVAL-3 results with –score s1nc option. Number of instances =664.

5.3 Hypothesis 3: The degree of difficulty in disambiguating a token is proportional to the number of senses of that token (polysemy).

Experiment 3: In order to observe the effect of polysemy on disambiguation, the SemCor corpus is partitioned into n classes, where each class $1 \leq i \leq n$ represents the instances having i possible senses. We evaluate instances in each class using WN-SRAW system with representative measures from each category, more specifically with gloss measure lesk, information content based measure jcn and path-based measure lch. The measures are used with the window sizes that give best results. We strictly didn't back off to sense1 in any case because we wanted to see the real effect of polysemy on the difficulty of disambiguation. The Spearman's rank correlation coefficient was calculated between polysemy and F-score using a freely available software R which provides an environment for statistical computing and graphics[28].

Observations and Analysis: In general, it is easy to disambiguate monosemous instances. However, in an arbitrary text, as noted by Fellbaum [6], a significant proportion of instances are polysemous due to the fact that frequently used types are polysemous. For instance, as shown in Table 6, the most frequently occurring types *be#v* and *make#v* from SemCor have 13 and 49 senses in WordNet respectively. To illustrate the proportion of polysemous instances in total number of instances, in SemCor about 78% instances are polysemous.

There are mixed conclusions about polysemy and difficulty of disambiguation. Preiss [33] argues that polysemy is not an ideal measure of difficulty. On the other hand Daelemans [10] concludes that the fluctuations in accuracy of disambiguation largely depend on the polysemy and entropy of the ambiguous words.

Tables 7, 8 and 9 show the precision, recall and F-score of first 25 classes $(n = 25)$ using lesk, jcn and lch respectively. A high negative correlation of -0.820 for lesk, -0.840 for jcn and -0.721 for lch demonstrates that the difficulty of disambiguation increases with increased polysemy. This also confirms the low F-score for verbs which are in general highly polysemous. The tables also show that the number of instances decrease with increase in the number of senses except for a few cases. The reason for more instances for polysemy=13 is that the most frequently occurring word type

[28] http://www.r-project.org/

semcor	Polysemy
be#v (8,400)	13
person#n (6,696)	3
not#r (1,703)	1
group#n (1,329)	3
have#v (1,126)	19
say#v (1,005)	11
location#n (993)	4
make#v (757)	49
man#n (576)	11
see#v (549)	24
know#v (512)	11
time#n (511)	10

Table 6: Most frequent types in SemCor where word type frequency > 500. Polysemy represents the total number of senses in WordNet.

be#v has 13 senses in WordNet. Similarly 19 senses of *have#v* leads to more number of instances for polysemy=19.

To summarize, we found a high negative correlation between polysemy and F-score suggesting that polysemy is a measure of difficulty and difficulty of disambiguation increases with increased polysemy. This confirms the conclusion in Daelemans 2002 [10]. It is important to note here that Daelemans's method of all-words sense disambiguation is supervised and uses sense distribution information. Therefore they conclude that difficulty depends upon polysemy as well as sense distribution entropy. However, WN-SRAW doesn't make use of word sense distribution information for disambiguation and hence the difficulty is independent of the sense distribution entropy except when used with information content measures. As we noted in background, content based measures in some sense, use the sense distribution information when used with WordNet. But gloss based and path based measures do not need any such information and consider all senses of an instance as equally probable.

Conclusion: We found a high negative correlation between polysemy and F-score demonstrating the hypothesis.

Polysemy	P	R	F	# instances
1	1.000	1.000	1.000	28,673 (19.67 %)
2	0.677	0.666	0.672	23,417 (16.06 %)
3	0.680	0.673	0.677	25,525 (17.51 %)
4	0.515	0.513	0.514	18,776 (12.88 %)
5	0.473	0.470	0.471	13,210 (9.06 %)
6	0.412	0.410	0.411	9,944 (6.82 %)
7	0.381	0.379	0.380	9,056 (6.21 %)
8	0.363	0.362	0.363	5,123 (3.51 %)
9	0.329	0.328	0.328	4,726 (3.24 %)
10	0.302	0.301	0.302	5,465 (3.75 %)
11	0.351	0.347	0.349	5,437 (3.73 %)
12	0.296	0.296	0.296	2,355 (1.62 %)
13	0.532	0.529	0.530	11,117 (7.63 %)
14	0.325	0.324	0.324	1,502 (1.03 %)
15	0.262	0.260	0.261	873 (0.60 %)
16	0.237	0.236	0.236	1,275 (0.87 %)
17	0.353	0.353	0.353	589 (0.40 %)
18	0.393	0.393	0.393	135 (0.09 %)
19	0.128	0.128	0.128	1,150 (0.79 %)
20	0.207	0.206	0.207	306 (0.21 %)
21	0.323	0.321	0.322	823 (0.56 %)
22	0.324	0.324	0.324	244 (0.17 %)
23	0.176	0.174	0.175	69 (0.05 %)
24	0.098	0.097	0.097	723 (0.50 %)
25	0.119	0.119	0.119	202 (0.14 %)

Table 7: Polysemy results with wntagged format, window=7, measure= lesk, contextScore=0.0, pairScore=0.0, –score n with lesk stoplist and no forcepos. Total number of instances = 145,773. Overall P=0.499, R=0.495, F=0.497. Spearman's rank correlation rho for Polysemy and F = -0.820

Polysemy	P	R	F	# instances
1	1.000	1.000	1.000	28,673 (19.67 %)
2	0.770	0.441	0.561	23,417 (16.06 %)
3	0.716	0.514	0.598	25,525 (17.51 %)
4	0.603	0.401	0.482	18,776 (12.88 %)
5	0.497	0.368	0.423	13,210 (9.06 %)
6	0.446	0.320	0.373	9,944 (6.82 %)
7	0.425	0.276	0.334	9,056 (6.21 %)
8	0.406	0.325	0.361	5,123 (3.51 %)
9	0.332	0.257	0.290	4,726 (3.24 %)
10	0.338	0.235	0.277	5,465 (3.75 %)
11	0.409	0.327	0.363	5,437 (3.73 %)
12	0.261	0.197	0.224	2,355 (1.62 %)
13	0.352	0.268	0.304	11,117 (7.63 %)
14	0.309	0.243	0.272	1,502 (1.03 %)
15	0.257	0.179	0.211	873 (0.60 %)
16	0.191	0.168	0.179	1,275 (0.87 %)
17	0.267	0.192	0.223	589 (0.40 %)
18	0.505	0.407	0.451	135 (0.09 %)
19	0.260	0.203	0.228	1,150 (0.79 %)
20	0.357	0.324	0.340	306 (0.21 %)
21	0.186	0.090	0.121	823 (0.56 %)
22	0.169	0.143	0.155	244 (0.17 %)
23	0.172	0.145	0.157	69 (0.05 %)
24	0.155	0.130	0.141	723 (0.50 %)
25	0.080	0.054	0.065	202 (0.14 %)

Table 8: Polysemy results with wntagged format, window=15, measure= jcn, contextScore=0.0, pairScore=0.0, –score n with no measure config, no forcepos and no stoplist. Total number of instances = 145,773. Overall P=0.528, R=0.323, F=0.401. Spearman's rank correlation rho for Polysemy and F = -0.840

Polysemy	P	R	F	# instances
1	1.000	1.000	1.000	28,673 (19.67 %)
2	0.596	0.342	0.435	23,417 (16.06 %)
3	0.597	0.430	0.500	25,525 (17.51 %)
4	0.433	0.289	0.347	18,776 (12.88 %)
5	0.341	0.254	0.291	13,210 (9.06 %)
6	0.339	0.244	0.284	9,944 (6.82 %)
7	0.284	0.184	0.223	9,056 (6.21 %)
8	0.247	0.198	0.220	5,123 (3.51 %)
9	0.248	0.193	0.217	4,726 (3.24 %)
10	0.283	0.197	0.232	5,465 (3.75 %)
11	0.202	0.162	0.180	5,437 (3.73 %)
12	0.169	0.128	0.146	2,355 (1.62 %)
13	0.458	0.349	0.396	11,117 (7.63 %)
14	0.211	0.166	0.186	1,502 (1.03 %)
15	0.173	0.120	0.142	873 (0.60 %)
16	0.206	0.181	0.193	1,275 (0.87 %)
17	0.186	0.134	0.156	589 (0.40 %)
18	0.092	0.074	0.082	135 (0.09 %)
19	0.390	0.305	0.342	1,150 (0.79 %)
20	0.194	0.176	0.185	306 (0.21 %)
21	0.216	0.104	0.141	823 (0.56 %)
22	0.367	0.311	0.337	244 (0.17 %)
23	0.051	0.043	0.047	69 (0.05 %)
24	0.152	0.129	0.139	723 (0.50 %)
25	0.137	0.094	0.111	202 (0.14 %)

Table 9: Polysemy results with wntagged format, window=15, measure= lch, contextScore=0.0, pairScore=0.0, –score n with no measure config, no forcepos and no stoplist. Total number of instances = 145,773. Overall P=0.420, R=0.259, F=0.320. Spearman's rank correlation rho for Polysemy and F=-0.721

5.4 Hypothesis 4: A significant percentage of word sense disambiguation error is caused by just a few highly frequent word types.

Experiment 4: To see in which cases the system doesn't predict the correct answer and whether the senses assigned are close to the correct sense, we carried experiments with frequently occurring types. For this, all instances of a most frequently occurring type are evaluated. The experiments were mainly carried using lesk measure with window size 7, which gives the best overall results. The measure was configured to use lesk stoplist[29].

Observations and Analysis: In general, word types in a coherent text follow Zipfian distribution as shown in Figure 18. This means that a few word types occur very frequently, while many word types only occur a few times. Therefore in the problem of all-words sense disambiguation, not all word types have the same contribution in the overall results. For example, a word that occurs 5 times won't shift the F-score of the system in any significant way no matter what you do with it, while one that occurs 9000 times can affect the overall results in a very big way.

More formally, the contribution of each word to the disambiguation can be described as below. Consider a text with n tokens and t types. Let p_i be the precision of $i^{th}(1 \leq i \leq t)$ type. Let f_i be the fraction such that $f_i = \frac{\#\ occurrences\ of\ i^{th}\ type}{n}$. Then the overall precision can be described as P=$\sum_{i=1}^{t} p_i f_i$. That is, the contribution of the i^{th} type to the overall precision is given by $p_i f_i$. So it is important to improve p_i for greater values of f_i. In other words, to increase the overall precision of WSD it is required that the frequently occurring types are disambiguated correctly.

Table 9 gives the precision, recall and F-score of most frequently occurring types in SemCor. It shows that for some cases WN-SRAW does very well and for others very poorly. For example, it does pretty well on frequently occurring polysemous nouns *person#n, group#n, location#n* but performs very poorly on the frequently occurring verbs *have#v, make#v, say#v and see#v*. The noun *person*, which occurs 6696 times in SemCor, results in a F-score of 0.993 while the verb *make*, which occurs 757 times in SemCor gives a very low F-score of 0.085. Again the polysemy might be playing a role here because frequently occurring nouns in SemCor have in average 6 senses

[29]Refer lesk stoplist in Appendix A.2

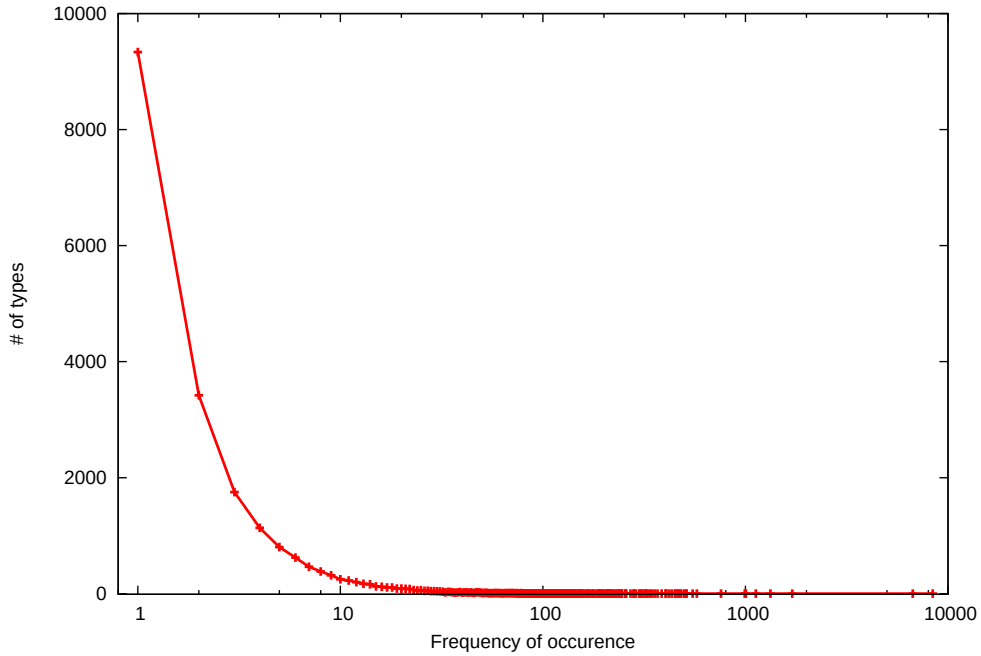

Figure 18: Word types in SemCor follow Zipfian distribution. Total number of types = 21,513

in contrast with average 21 senses in case of verbs.

We present individual type results using confusion matrices $i \times j$ where each ij^{th} entry represents the count of how many times the actual member of class i was predicted as class j. In other words, the rows represent the answer key results (the gold standard) while the columns represent WN-SRAW results. Diagonal entries which represent the correct prediction are displayed in bold. We ignore the instances having more than one possible annotation in the key. For example, the following SemCor instance *be* (lemma=be), which has two possible annotations 2 and 1 (wnsn=2;1), will be ignored. This leads to a little difference between the counts in Table 9 and the counts mentioned here.

<wf cmd=done rdf=is pos=VB lemma=be wnsn=2;1 lexsn=2:42:06::;2:42:03::>'s<wf>

Table 11, 12 and 13 show the confusion matrices for the verb *say#v* from SemCor, SENSEVAL-2 and SENSEVAL-3 respectively. Table 11 shows the skewed sense distribution of this particular word type where 85% of the instances have sense1 as the correct sense. The sense assignment distribution in the tables shows that most of the times sense1 is confused with sense6 and sense5. For instances,

word type	P	R	F	#Instances	Polysemy
be#v	0.624	0.621	0.623	8,400 (4.5%)	13
person#n	1.000	0.987	0.993	6,696 (3.6%)	3
not#r	1.000	0.984	0.992	1,703 (0.91%)	1
group#n	0.981	0.981	0.981	1,329 (0.71%)	3
have#v	**0.124**	**0.123**	**0.124**	1,126 (0.61%)	19
say#v	**0.215**	**0.210**	**0.212**	1,005 (0.54%)	11
location#n	0.955	0.952	0.952	993(0.0.53%)	4
make#v	**0.085**	**0.085**	**0.085**	757 (0.41%)	49
man#n	0.674	0.672	0.673	576(0.31%)	11
see#v	**0.053**	**0.053**	**0.053**	549 (0.29%)	24
know#v	**0.280**	**0.268**	**0.274**	512 (0.28%)	11
time#n	**0.103**	**0.103**	**0.103**	511 (0.%28)	10

Table 10: Frequently occurring types from SemCor where the instance frequency account for at least 0.27% of the SemCor data (i.e. instance frequency > 500). Total SemCor instances = 185,273, measure=lesk, window size=7 and using –word

	1	2	3	4	5	6	7	8	9	10	11	Key
1	**117**	38	19	34	147	452	19	0	0	0	3	829
2	21	**9**	1	6	23	55	4	0	0	0	0	119
3	3	1	**0**	2	4	6	0	0	0	0	0	16
4	0	0	0	**0**	0	1	0	0	0	0	0	1
5	0	0	0	2	**3**	3	0	0	0	0	0	8
6	0	0	0	1	0	**0**	1	0	0	0	0	2
7	1	0	0	0	0	0	**0**	0	0	0	0	1
8	1	0	0	0	0	0	0	**0**	0	0	0	1
Ans	143	48	20	45	177	517	24	0	0	0	3	977

Table 11: Confusion matrix of the verb *say* from SemCor measure=lesk with lesk stoplist and window size=7, P=0.134, R=0.130, F=0.132, *say#v* has total 11 senses.

	1	2	3	4	5	Key
1	**1**	1	0	0	1	3
2	0	**1**	0	0	5	6
3	0	0	**0**	0	1	1
4	0	0	0	**0**	0	0
5	0	0	0	0	**0**	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	2	2	0	0	6	10
Ans	3	4	0	0	13	20

Table 12: Confusion matrix of the verb *say* from SENSEVAL-2 measure=lesk and window size=7, P=0.083, R=0.083, F=0.083, *say#v* has total 11 senses.

	1	2	3	4	5	6	Key
1	**0**	0	0	0	1	1	2
2	2	**0**	1	0	1	2	6
3	0	0	**0**	0	0	0	0
4	0	0	0	**0**	0	0	0
5	0	0	0	0	**0**	0	0
6	0	0	0	0	0	**0**	0
7	0	0	0	0	0	0	0
8	1	0	0	0	1	0	2
9	0	0	0	0	0	1	1
Ans	3	0	1	0	3	4	11

Table 13: Confusion matrix of the verb *say* from SENSEVAL-3 measure=lesk with lesk stoplist and window size=7, P=0.000, R=0.000, F=0.000, *say#v* has total 11 senses.

in Table 10, 462 instances have been assigned sense6 where sense1 is the correct sense. In case of SENSEVAL-3 (Table 13), no prediction was correct among total 10 instances. Again we can see that 64% instances were assigned sense5 or sense6. Here are the three senses in WordNet.

1. sense1 (1861) state, say, tell – (express in words; “He said that he wanted to marry her”; “tell me what is bothering you”;“state your opinion”; “state your name”)

2. sense5 (8) order, tell, enjoin, say – (give instructions to or direct somebody to do something with authority; ”I said to him to go home”; ”She ordered him to do the shopping”; ”The mother told the child to get dressed”)

3. sense6 (4) pronounce, articulate, enounce, sound out, enunciate, say – (speak, pronounce, or utter in a certain way; “She pronounces French words in a funny way”; “I cannot say ‘zip wire”’; “Can the child sound out this complicated word?”)

Consider the following instance of *say#v* where lesk assigns sense6 instead of sense1.

	1	2	Key
1	**6**	15	21
Ans	6	15	21

Table 14: Confusion matrix of the verb *ringer* from SENSEVAL-2 measure=lesk and window size=7, P=0.222, R=0.222, F=0.222

original: *"This is a poor boy's bill", said person.*
WN-SRAW input: *be#v poor#a boy#n bill#n* ***say#v*** *person#n*

The main reason of assigning sense6 in this case is say#v#6 finds a strong relatedness with person#n#1 which is *human being* sense of *person#n*. Since sense6 has words *pronounce, articulate, enounce, sound out, enunciate,* it finds many common words with person#n#1 like *nose, mouth, speak, word etc*. We also observed that in SemCor, *say#v* and *person#n* co-occur approximately 450 number of times. This could be the reason why lesk chooses sense6 over sense1 for 452 instances.

One would think that these senses might be related strongly which results in confusion of assignment. But lesk relatedness score between say#v#1 and say#v#6, which is 4, doesn't show strong relatedness between the two senses as compared with the maximum relatedness score of 27 between say#v#1 and say#v#8. So in this particular case, we can say that lesk mis-prediction is not just because of fine distinction between WordNet senses. In fact, in some cases we observed that it predicts a sense with a completely different meaning.

For example, as shown in Table 14, the results of frequently occurring word type *ringer#n* in SENSEVAL-2, show that the algorithm confuses sense1 and sense2 a significant number of times, even though these are completely different senses of the noun *ringer*.

1. sense1 toller, bell ringer, ringer – (a person who rings church bells (as for summoning the congregation))
2. sense2 ringer, dead ringer, clone – (a person who is almost identical to another)
3. sense3 ringer – (a contestant entered in a competition under false pretenses)

4. sense4 ringer – ((horseshoes) the successful throw of a horseshoe or quoit so as to encircle a stake or peg)

It was also observed that if there are related instances in the context, we disambiguated *ringer#n* correctly. For example, in the following case there are instances like *bell#n* and *ring#v* in the context which help to predict the correct sense.

original: *But there still are n't enough ringers to ring more than six of the eight bells.*
WN-SRAW input: *still#r enough#a ringer#n ring#v more#a bell#n*[30]

But in the sentence below, there are no instances in the context which tell about the sense of the target instance *ringer#n*. Given only these instances, it would be hard even for human beings to disambiguate the target instance correctly.

original: *Now , only one local ringer remains : 64-year-old Derek Hammond.*
WN-SRAW input: *now#r only#r local#a ringer#n remain#v*

Another example which is very hard for lesk is the noun *time*. We observed that most of the times sense5 was assigned in place of sense1.

1. sense1 (219) time, clip – (an instance or single occasion for some event; "this time he succeeded"; "he called four times"; "he could do ten at a clip")
2. sense5 (36) time – (the continuum of experience in which events pass from the future through the present to the past)

For example, in the following sentence, the word time#n is sense tagged as time#n#5 where the correct tagging should be time#n#1.

Original: *New bonds would be issued every **time** a portion of the old ones are paid off by tax authorities.*
WN-SRAW input: *bond#n issue#v every#a time#n portion#n old#a pay_off#v tax#n authorities#n*

[30]The reason for skipping content words from the original text is that they are not sense tagged in the corpus.

	1	2	3	4	5	6	7	8	9	10	Key
1	**2**	1	61	0	121	0	5	0	0	5	195
2	1	**8**	52	0	81	0	3	2	0	5	152
3	0	3	**17**	0	53	0	0	0	0	3	76
4	0	0	9	**1**	24	0	1	0	0	1	36
5	0	1	5	0	**26**	0	0	0	0	3	35
6	0	3	1	0	0	**0**	0	0	0	2	6
7	0	0	2	0	5	0	**0**	0	0	0	7
8	0	0	0	0	3	0	0	**1**	0	0	4
9	0	0	0	0	1	0	0	0	**0**	0	1
Ans	3	16	147	1	314	0	9	3	0	19	512

Table 15: Confusion matrix of the noun *time* from SemCor measure=lesk with lesk stoplist and window size=7, P=0.106, R=0.106, F=0.106, *time#n* has total 10 senses.

Because of the word *continuum* in the definition of *time#n#5*, there is a strong relatedness found between *time#n#5* and *portion#n#1* which results in choosing sense5 over sense1.

In order to see how much the first 12 most frequently occurring word types contribute to the overall error, we evaluated polysemous instances in SemCor by excluding the frequently occurring and poorly performing word types[31] *have#v, make#v, time#v, know#v, see#v and say#v* . We used lesk with window size 7 which gives the best performance. The results showed that the F-score increased from 0.497 (all instances) to 0.510 (excluding poorly performing instances). We also evaluated polysemous instances in SemCor by assigning the correct sense (from the key) to the above poorly performing word types which increased F-score from 0.497 to 0.547.

In both cases, the reason F-score didn't increase a great deal may be because, as shown in Figure 17, the overall proportion of the instances of frequently occurring word types is not that significant compared to the total instances of the word types that occur only a few times.

[31]This can be done using –exceptword option of the scorer program of WN-SRAW.

Conclusion: The experimental evidence supports the hypothesis.

5.5 Hypothesis 5: Part-of-speech tagged text will be disambiguated more accurately.

Experiment 15: To see if knowing parts-of-speech information for raw text is helpful, we carried out experiments with tagged and raw text. The experiments were mainly carried out on SemCor. SemCor is parts-of-speech tagged, so it is required to extract raw text ignoring parts-of-speech information. In SemCor, WordNet compounds are already identified and in the experiment we use them as they are. The first part of the experiment includes extracting raw text by ignoring parts-of-speech tags. A key file is also extracted for evaluating the disambiguation results. The extracted raw text is then disambiguated by disabling compoundify [32] and using lesk measure. The results are evaluated against the extracted SemCor key. We used window size of 5 since with this context size lesk performs quite well without taking too long[33].

In the second part, we extract plain text from SemCor which contains content words as well as function words[34]. Then the plain text is part-of-speech tagged using a general purpose freely available tagger, the Brill tagger[35] [2]. The Brill tagger is used with the default settings and Penn Treebank tags are assigned to the plain text. Later these tags are mapped to the WordNet tags[36]. Only content words from this part-of-speech tagged text are extracted since for disambiguation we are only interested in content words. We observed the part-of-speech tagging accuracy of 92.11% on the content words. Considering the state of the art part-of-speech accuracy of around 97%, the reason for the observed low accuracy is that the frequently occurring functions words such as *the*, *an*, *a* were not evaluated. The mapped Brill tagged text is then disambiguated as a wntagged formatted text.

Observations and Analysis: In case of raw format, it is crucial that the method assigns the correct part-of-speech tags to the instances. If the part-of-speech tag assignment is not correct then while finding relatedness, WN-SRAW won't be looking at the correct part-of-speech tag and hence it won't be able to do WSD correctly. In such cases, there is no further chance to improve the

[32] WN-SRAW provides an option to disable compoundifying via –nocompoundify flag. This tells WN-SRAW not to identify any other compounds. If this option is not used, the algorithm identifies compounds which are not there in the key. We observed this for the compound *be_well*.

[33] Refer Table 31 in Appendix A.3

[34] Function words such as *the, an, of* are immensely important for part-of-speech tagging.

[35] http://duluthted.googlepages.com/RULE_BASED_TAGGER_V.1.14.tar.Z

[36] WordNet supports only four tags n, v, a, r. Refer Appendix A.1 for the mapping.

disambiguation. For example, in the following sentence if the system identifies the word *hand* as a noun instead of a verb, there is no further possibility of correct disambiguation.

***Hand** me the spoon.*

Using part-of-speech tagged text can improve the performance of the algorithm in terms of speed and precision. A part-of-speech tagged word will have fewer possible senses than its raw version. For example, the word *look* has 14 senses but *look#n* has only 4 senses. Hence part-of-speech tagging will reduce the average number of senses in the corpus (S_{avg} in equation (5) of Chapter 3) which will result in improved performance in terms of time.

Table 16 shows the comparison of tagged and raw text experiments which reveal that part-of-speech tagging is helpful for all parts-of-speech and improves the overall performance by 5.9%. Tagged text performs especially well for adjectives and adverbs. We can see an improvement of 9.3% in case of adjectives and 14.6% for adverbs. This might be because for raw text, as shown in Table 18, lesk tends to mis-tag many adjectives as nouns. As nouns have very rich hierarchical structure in WordNet, it might be possible that the measure is finding more relatedness with noun form of those instances. For example, in the following sentence, lesk is more likely to choose *blue#n#2* (which in fact should be *blue#a#1*) which is *blue clothing* because of the context word *dress*.

*The **blue** dress was pretty.*

For nouns we can see an improvement of 2.7% and 7.3% for verbs. Table 18 shows that 91.5% of the total nouns were recognized correctly indicating that the algorithm did not have much trouble while assigning part-of-speech tags to nouns. It didn't perform that well on verbs predicting 21% of the verbs incorrectly. Out of 21% of incorrectly tagged verbs, 19.8% of verbs were tagged as nouns. The algorithm couldn't recognize adverbs and adjectives around 33% of the total number of instances. We can see that 25% of adverbs were tagged as adjectives and 9% as nouns. This is reflected in the low precision for adverbs in case of raw format. Assignment of 20% of verbs and 21% of adjectives to nouns also explain lower precision for verbs and adjectives in case of raw format. In case of Brill tagged text (Table 17), we can see the similar trend. Most error is in predicting verbs and adjectives. Many times verbs were tagged as nouns and adverbs as adjectives.

		POS annotated	Brill tagged	raw
Nouns	P	0.544	0.535	0.504
	R	0.539	0.525	0.501
	F	0.542	0.530	0.503
Verbs	P	0.398	0.389	0.313
	R	0.391	0.380	0.310
	F	0.394	0.384	0.311
Adjectives	P	0.582	0.541	0.422
	R	0.574	0.487	0.420
	F	0.578	0.513	0.421
Adverbs	P	0.473	0.436	0.283
	R	0.454	0.418	0.279
	F	0.464	0.427	0.281
All	P	0.502	0.484	0.419
	R	0.494	0.469	0.416
	F	0.498	0.476	0.417

Table 16: Tagged and raw format experiments. Measure used is lesk with window=5 with lesk stoplist, –nocompoundify, –score poly. 135,572 attempted out of 143,431 total instances for Brill tagged text and 139,753 attempted out of 143,431 total instances for raw text. The POS annotated text is the part of speech annotated SemCor text (# instances = 145,773).

Note that the number of instances in case of Brill tagged text is less than in raw text. This is because for some words, the Brill tagger assigns a part-of-speech tag where word#pos is not defined in WordNet.

	Noun	Verb	Adj.	Adv.	key
Noun	**82,703**	470	875	99	84,147
Verb	1,014	**44,362**	36	7	45,419
Adj.	1,273	120	**25,341**	559	27,293
Adv.	194	17	734	**14,706**	15,651
Ans.	85,184	44,969	26,986	14,706	**172,510**

Table 17: SemCor Brill tagged text confusion matrix. Includes only the instances where word#pos of the Brill tagged text is defined in the WordNet

	Noun	Verb	Adj.	Adv.	Key
Noun	**78,561**	6,245	1,002	48	85,856
Verb	9,300	**37,199**	373	6	46,878
Adj	6,762	3,166	**21,220**	227	31,375
Adv	1,595	365	3,591	**12,274**	17,825
Ans	96,218	46,975	26,186	12,274	**181,934**

Table 18: SemCor raw text confusion matrix. Includes only the attempted instances, i.e the instances where the relatedness is found with the surrounding instances using lesk.

To summarize, the results show that using part-of-speech taggers before WSD is helpful. It improves performance in terms of F-score. The overall F-score was improved by 5.9%. Using part-of-speech tagged text was especially useful for adjectives and adverbs. We saw an improvement of 9.3% in case of adjectives and 14.6% for adverbs. For nouns the improvement was 2.7% and 7.3% for verbs.

Conclusion: The experimental results show that knowing part-of-speech information helps in disambiguation, in support of the hypothesis.

corpus	nouns	verbs	adjectives	adverbs
SemCor	5.40 (0.544)	11.31 (0.390)	5.31 (0.582)	4.03 (0.469)
SENSEVAL-2	5.24 (0.520)	10.49 (0.303)	4.36 (0.570)	4.51 (0.509)
SENSEVAL-3	6.22 (0.460)	12.14 (0.335)	4.63 (0.482)	–

Table 19: Average polysemy per part-of-speech for polysemous instances. The parenthesis show lesk F-score for the part-of-speech.

5.6 Hypothesis 6: Given any two parts-of-speech, the more polysemous will be less accurately disambiguated.

Experiment 6: First average polysemy per part-of-speech is calculated. Then polysemous instances are evaluated for each part-of-speech using –score poly.

Observations and Analysis: Not all parts-of-speech have the same degree of polysemy. For example, words like *accurately* or *aloud* don't have as many interpretations as the words *see* or *make* have. In general, adverbs and adjectives are less polysemous than verbs and nouns.

Table 19 shows the average polysemy per part-of-speech[37]. Now, according to Hypothesis 3, the degree of difficulty in disambiguating a word is directly proportional to the polysemy. Therefore, we expect to get best F-score for adverbs and adjectives followed by nouns and finally the least F-score for verbs.

Figures 20 to 31 show the parts-of-speech results for polysemous instances[38]. F-scores in these Figures and Table 19 reveal that, the system gets best results for adjectives, followed by nouns and then adverbs. The system gets worst results for verbs. All pairs follow the hypothesis except for the noun-adverb pair. This indicates that polysemy is not the only thing that counts for difficulty and there are some other factors as well which affect the overall performance. Our intuition here is that this might be because of the structure of WordNet[39]. As we noted in background, the *is-a* hierarchy for nouns is a distinguishing characteristic of WordNet. The noun hierarchy is deep and

[37] All adverbs in SENSEVAL-3 are monosemous resulting in no entry for SENSEVAL-3 adverbs in Table 19.

[38] The reason for no results in Figure 31 is that SENSEVAL-3 doesn't have any polysemous adverbs.

[39] This intuition is invalid for vector measure as it doesn't use WordNet structure to find relatedness.

rich with relations. Adverbs are not arranged in hierarchies. They have relations like antonymy and pertainymy, however, these relation links are very sparse. The reason for adjectives performing better than adverbs might be that the average polysemy of adverbs and adjectives is very close. In fact for SENSEVAL-2 average polysemy of adjectives is less than adverbs. Moreover, adjectives have relatively more relations than adverbs. There are attribute relation links between adjectives and nouns. This might be the explanation of the low results for less polysemous adverbs.

The *is-a* hierarchies for verbs in WordNet are many and shallow. Figure 16 shows an example of how shallow verb hierarchies are. The most specific verb *scamper* is just three edges away from the most general verb *locomote*. There are other relations for verbs such as derived-from, hyponym and entailment for verbs. But they are not as rich as noun relations. Our intuition is that high polysemy and the WordNet structure for verbs make it hardest to disambiguate, giving the lowest F-score.

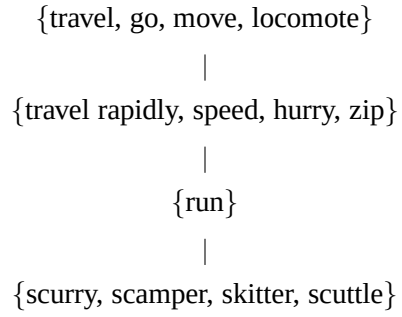

Figure 19: An illustration of verb hierarchy in WordNet

Conclusion: The hypothesis implies that adverbs will be more accurate than nouns but the evidence says the opposite.

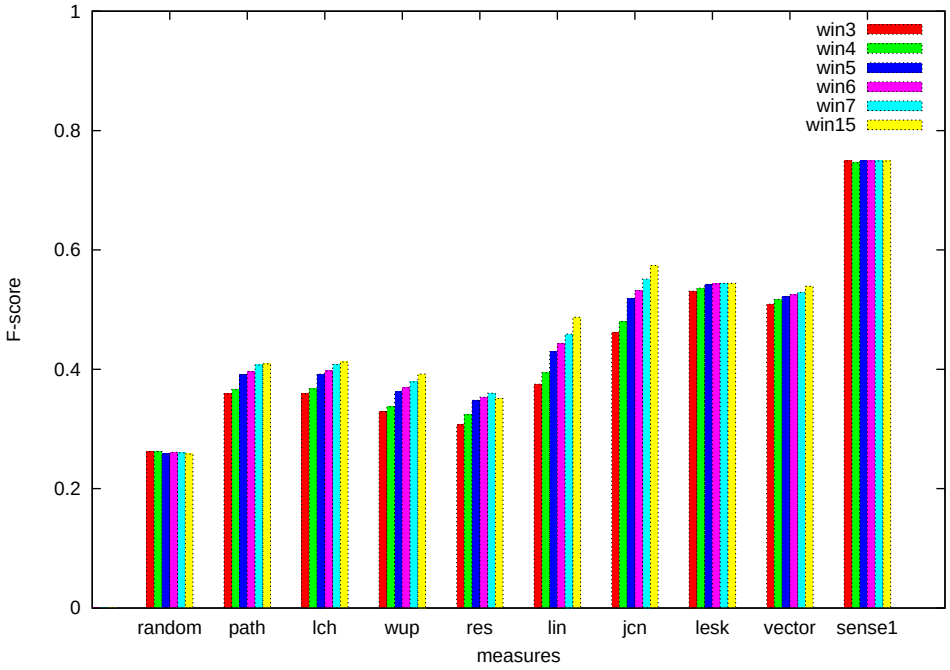

Figure 20: SemCor noun results with –score poly option.

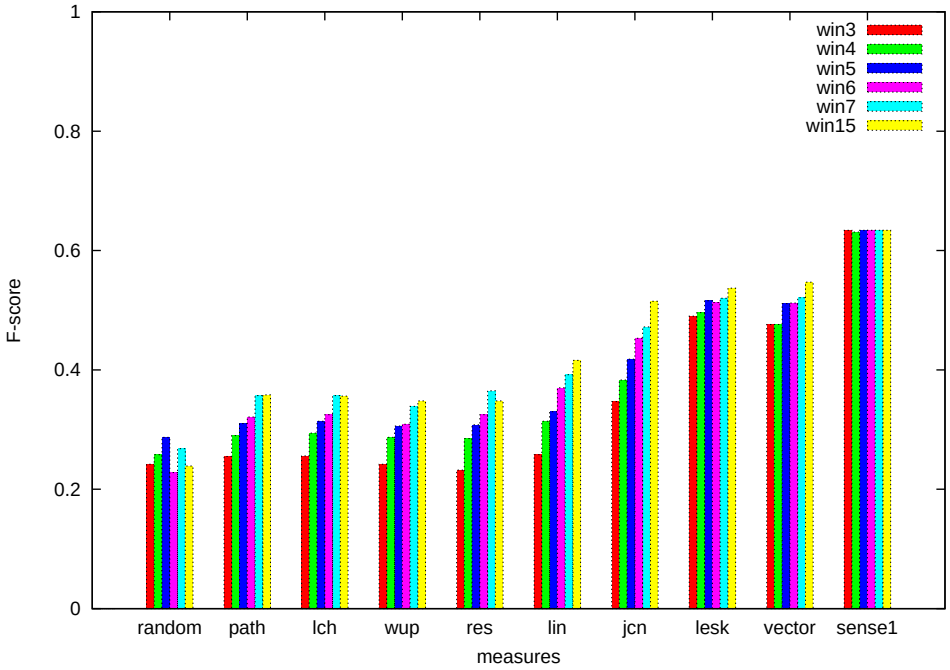

Figure 21: SENSEVAL-2 noun results with –score poly option.

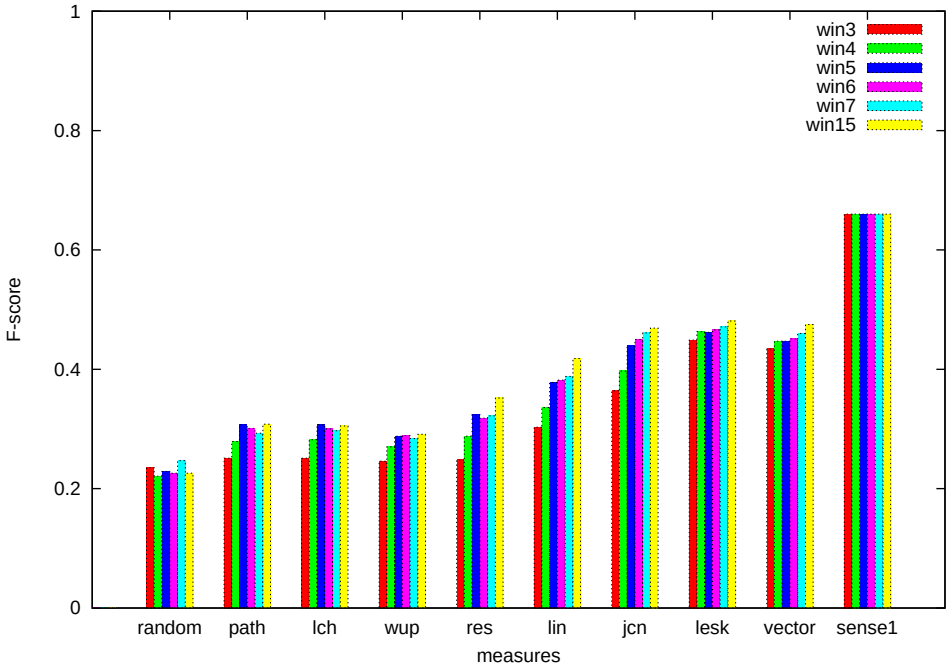

Figure 22: SENSEVAL-3 noun results with –score poly option.

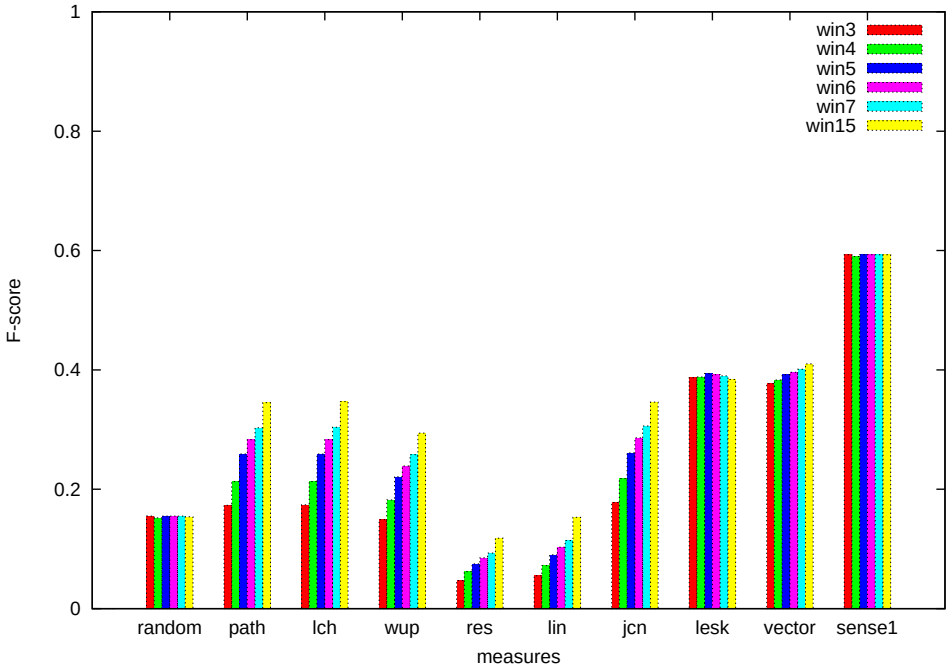

Figure 23: SemCor verb results with –score poly option.

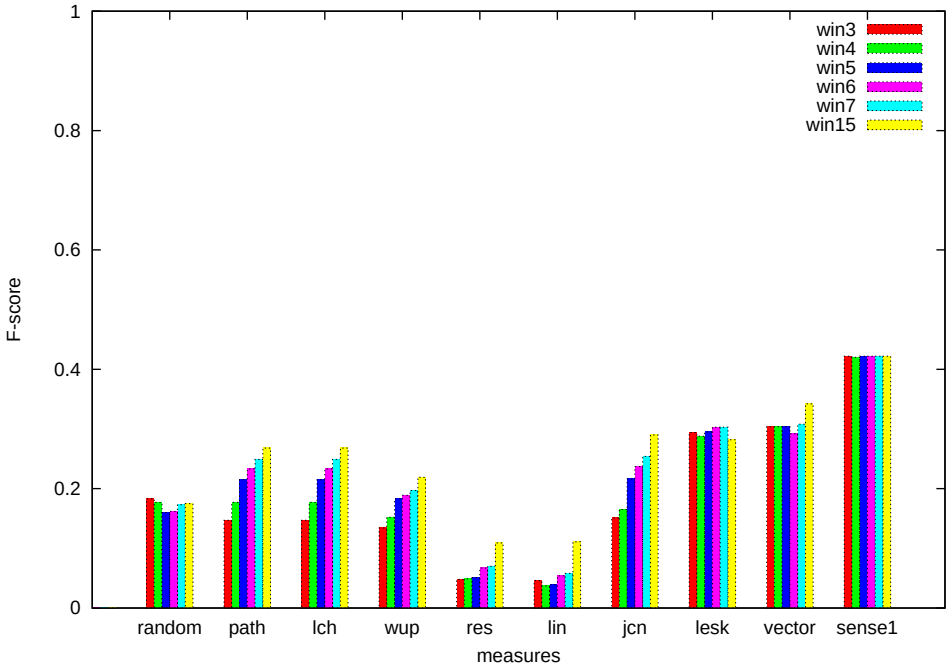

Figure 24: SENSEVAL-2 verb results with –score poly option.

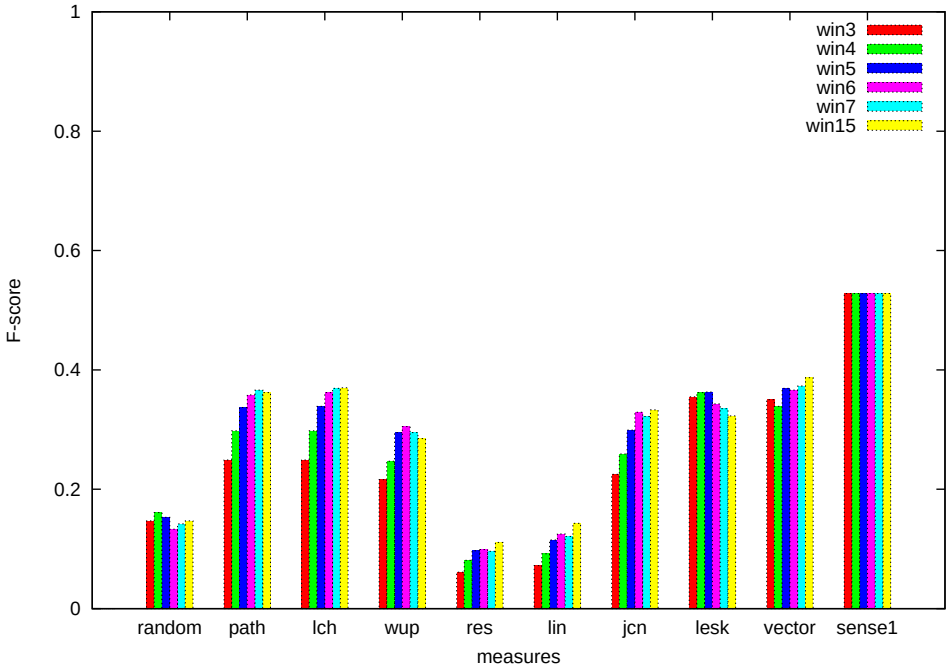

Figure 25: SENSEVAL-3 verb results with –score poly option.

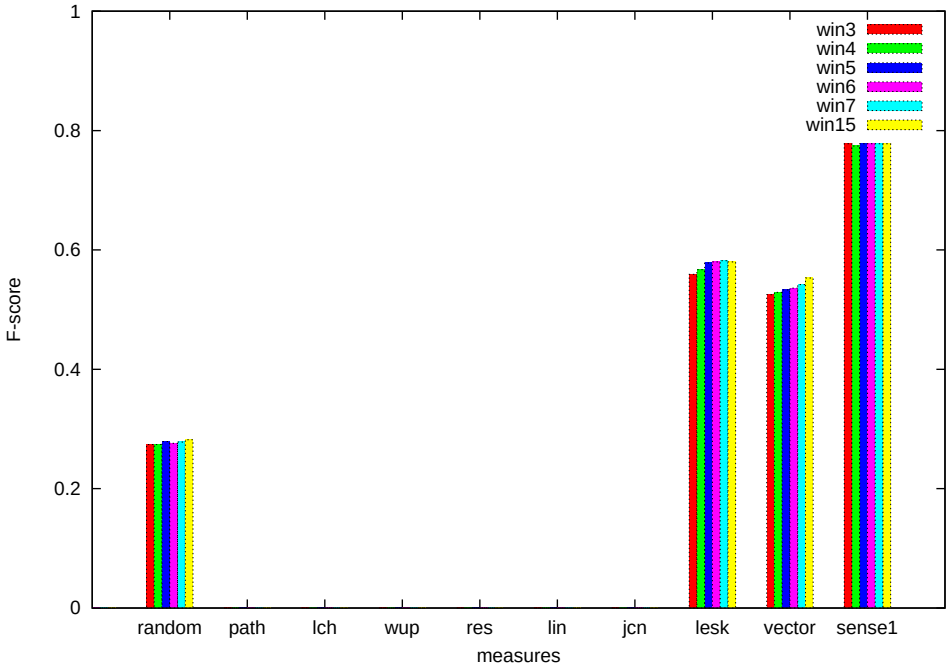

Figure 26: SemCor adjective results with –score poly option.

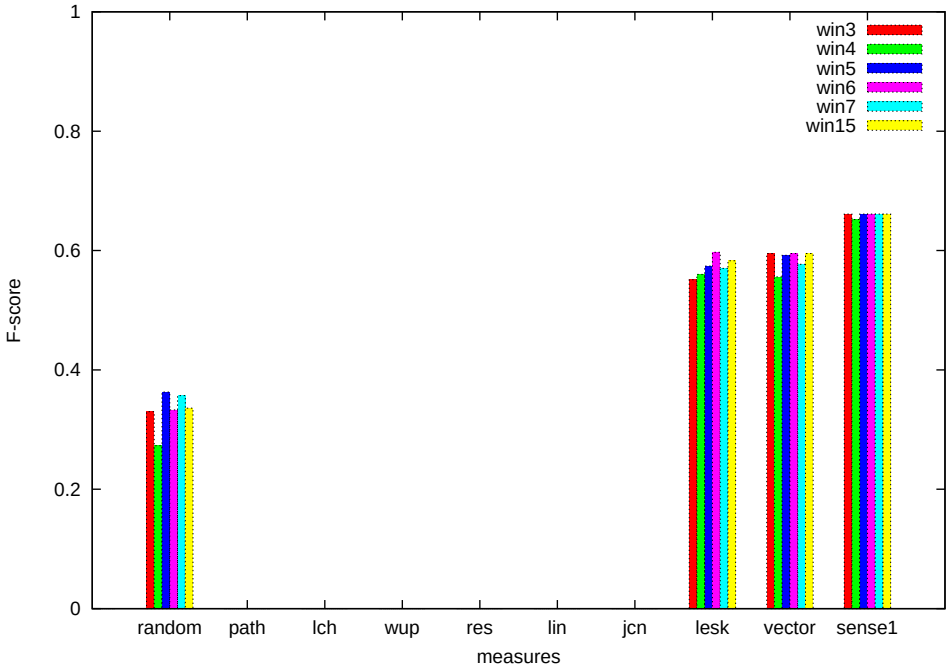

Figure 27: SENSEVAL-2 adjective results with –score poly s1nc option.

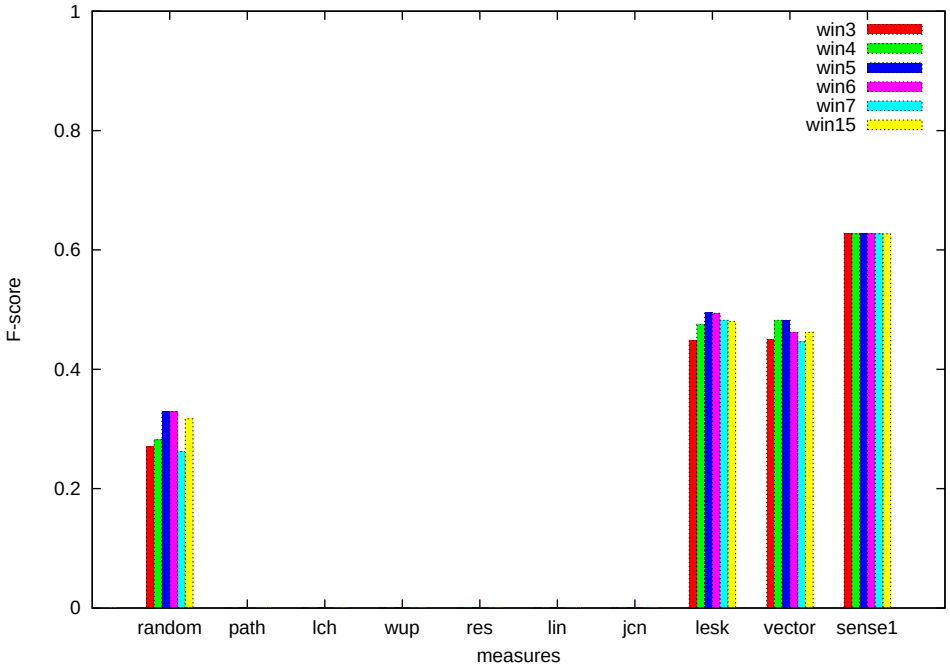

Figure 28: SENSEVAL-3 adjective results with –score poly option.

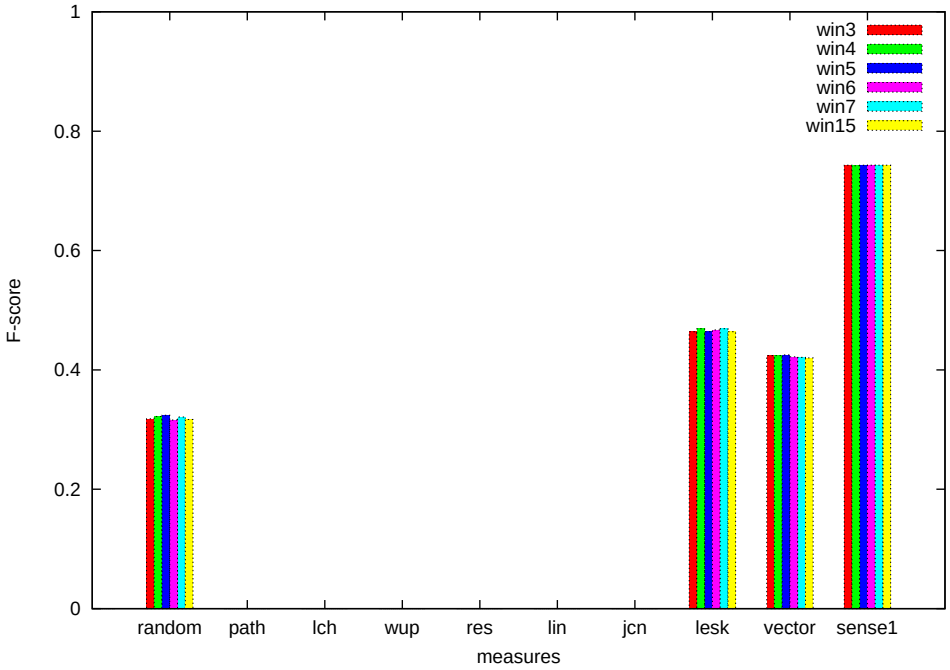

Figure 29: SemCor adverb results with –score poly option.

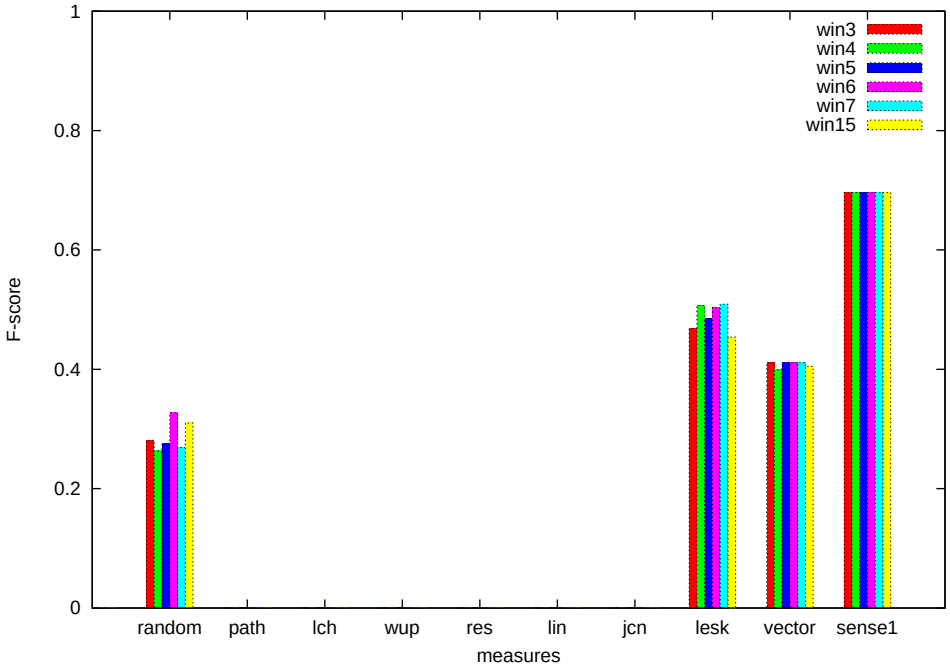

Figure 30: SENSEVAL-2 adverb results with –score poly option.

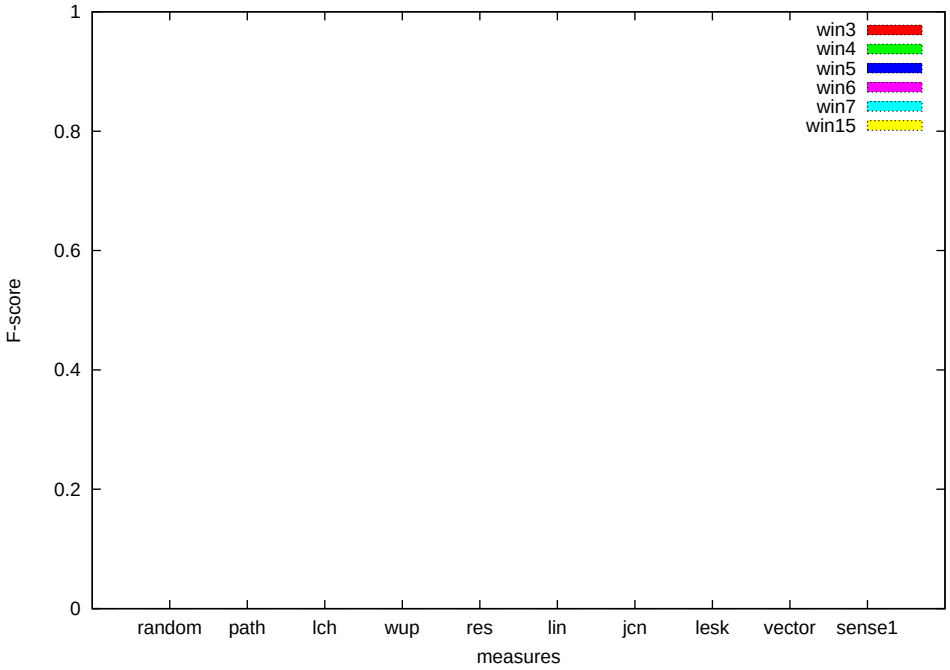

Figure 31: SENSEVAL-3 adverb results with –score poly option.

5.7 Other Observations

In a coherent text, not all instances are polysemous. It also contains easy to disambiguate monosemous instances. As we noted in Chapter 4, the corpora we are using have around 20% monosemous instances. The disambiguation of a monosemous instance involves assigning the only available sense to the instance. This naturally boosts the overall performance because of the significant proportion of monosemous instances.

As shown in Figures 32, 33 and 34, when monosemous instances are assigned the only available sense, similar trend as of Experiment 1 was observed with the difference that monosemous instances boost the overall F-score. The results also demonstrate the percentage of monosemous words in all three corpora. Similar to Experiment 1, the best performing measures are lesk and vector.

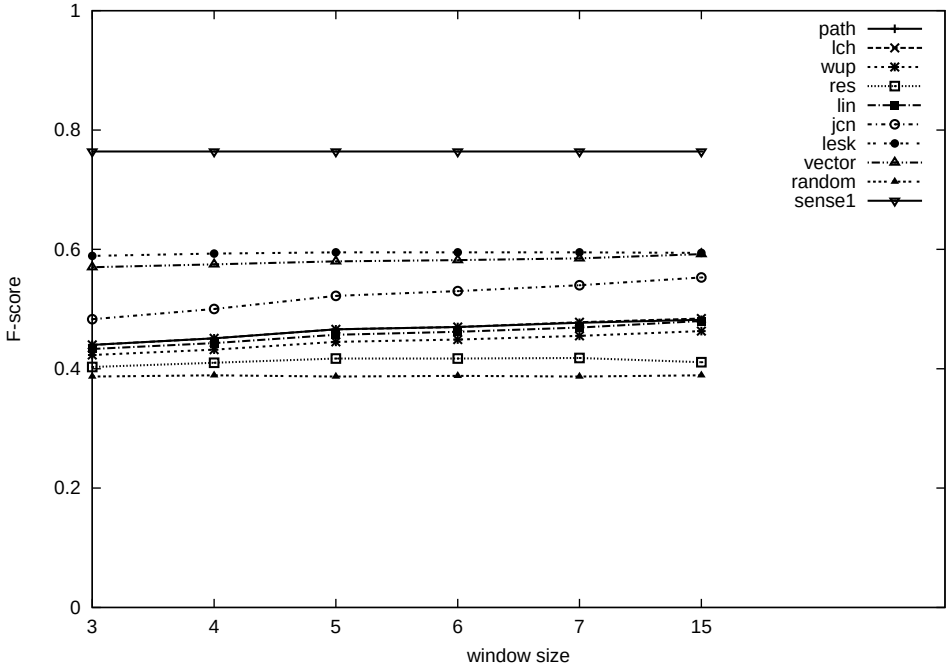

Figure 32: SemCor results with –usemono option. Number of instances = 185273.

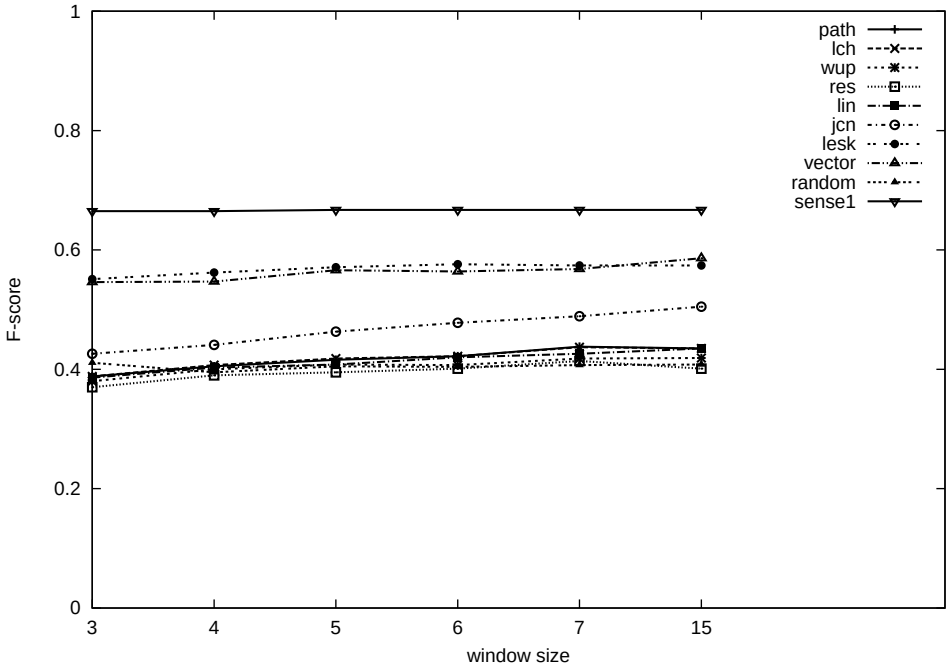

Figure 33: SENSEVAL-2 results with -usemono option. Number of instances =2260.

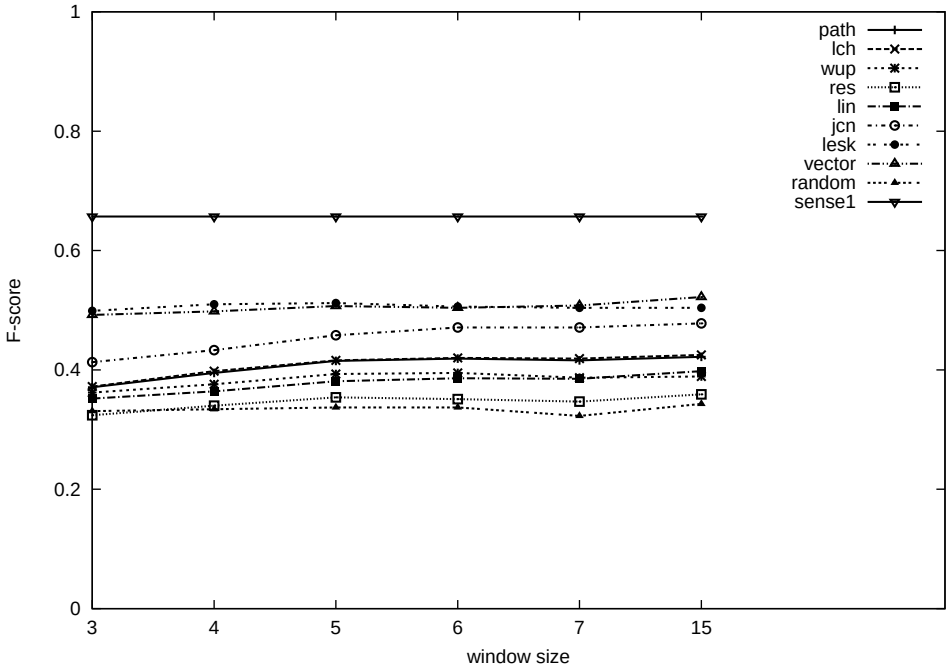

Figure 34: SENSEVAL-3 results with –usemono option. Number of instances =1937.

In 2001, Budanitsky and Hirst [3] report that jcn measure gives best results in the correction of word spelling errors compared to the measures proposed by lch, lin, res, hso while, in 2005, Pedersen [31] found that the extended measure of gloss overlap (lesk) together with JiangConrath (jcn) measure and the gloss vector (vector) measure outperforms the other measures in the disambiguation of noun instances of SENSEVAL-2 lexical sample data.

The summary of best performing measures for all-words sense disambiguation problem using WN-SRAW system is shown in Tables 20, 21 and 22. We observed that among the measures we used, overall lesk and vector performed best. For verbs vctor_{15} gave the best results. This might be because of the fuzziness of this measure and its ability to find hidden semantic similarity. For adverbs and adjectives $\text{lesk}_{6/7/15}$ performed better than other measures by a large margin. As we noted, relations for adverbs and adjectives are very sparse and hence to find relatedness for these parts-of-speech, it is required to exploit all possible relations exhaustively. This is exactly what lesk does.

Other than these experiments, we also carried experiments with different values for $context\text{-}threshold$ and $pair\text{-}threshold$[40]. No significant improvement with a particular threshold value was observed. Higher threshold values result in lower recall which results in overall lower F-score.

[40]This can be done using –contextScore and –pairScore options.

POS	SemCor	SENSEVAL-2	SENSEVAL-3
Nouns	$jcn_{15}(0.574)$	$vector_{15}(0.547)$	$lesk_{15}(0.481)$
Verbs	$vector_{15}(0.410)$	$vector_{15}(0.342)$	$vector_{15}(0.387)$
Adj.	$lesk_{7}(0.582)$	$lesk_{6}(0.597)$	$lesk_{6}(0.494)$
Adv.	$lesk_{7}(0.469)$	$lesk_{7}(0.509)$	-

Table 20: Best performing measures for polysemous instances (–score poly option), subscript denotes the window size and the parenthesis denotes F-score

POS	SemCor	SENSEVAL-2	SENSEVAL-3
Nouns	$lin_{15}(0.269)$	$vector_{15}(0.398)$	$lin_{15}(0.296)$
Verbs	$lesk_{5}(0.149)$	$vector_{15}(0.203)$	$vector_{5}(0.170)$
Adj.	$lesk_{5}(0.308)$	$lesk_{6}(0.368)$	$lesk_{15}(0.312)$
Adv.	$lesk_{15}(0.289)$	$lesk_{6}(0.343)$	-

Table 21: Best performing measures for instances where sense1 is not correct (–score s1nc option), subscript denotes the window size and the parenthesis denotes F-score

POS	SemCor	SENSEVAL-2	SENSEVAL-3
Nouns	$jcn_{15}(0.658)$	$vector_{15}(648)$	$lesk_{15}(0.481)$
Verbs	$vector_{15}(0.440)$	$vector_{15}(0.358)$	$vector_{15}(0.387)$
Adj.	$lesk_{7}(0.708)$	$lesk_{6}(0.676)$	$lesk_{6}(0.494)$
Adv.	$lesk_{7}(0.681)$	$lesk_{6}(0.689)$	$all_{all}(1.000a)$

Table 22: Best performing measures for monosemous and polysemous instances (–usemono option), subscript denotes the window size and the parenthesis denotes F-score

6 Related Work

Word Sense Disambiguation is a central problem in Natural Language Processing and has a long history of research. A variety of approaches including supervised, unsupervised, knowledge based and combination approaches have been tried. This chapter gives overview of some of the prominent approaches.

6.1 Miller, et al., 1994

Miller, et al. [23] was the first to explain how to determine the most frequent sense heuristic (sense1 scheme of WN-SRAW) which is still hard to beat for most of the WSD systems. This paper proposes benchmarks for WSD systems. A semantic concordance[41] [24] that combines passages from the Brown corpus with the WordNet lexical database [6] was used to explore three different heuristics for word sense disambiguation baselines.

This paper argues that the lower bound of 75% of a WSD system suggested by Gale, Church and Yarowsky [7] is plausible but there is no clear method described to find this lower bound. In their paper, they start with a lowest lower bound which is the proportion of monosemous words in the text corpus.

In the guessing heuristic, if the word is monosemous then the available sense is assigned to it. For all polysemous words, a random sense from n possible WordNet senses is chosen with a probability $1/n$. They applied this on 103 passages from the Brown corpus and observed an accuracy of 45% on 101,284 words and 26.8% on 76,067 polysemous words. This gives a baseline for a Word Sense Disambiguation system.

The second heuristic is the most frequent sense heuristic in which semantic concordance is used to determine which sense occurred most often. They used the version of the sense-tagged corpus [24] that was available in August 1993. In the training phase, the sense frequencies are estimated for open class words broken down by part-of-speech. For example, the frequencies of all 14 senses $\{s_1, s_2, ..., s_{14}\}$ of the word *look* with possible part-of-speech tags $pos = \{n, v\}$ would be tabulated

[41] This is 1.6 version of SemCor we use.

according to how frequently look#pos#s_i occurs in the corpus. It was observed that polysemous words occur frequently enough and can provide a good estimates on a relatively smaller sample size. In the testing phase, if the word is monosemous then it is assigned the only available sense. For all polysemous words, if the word is found in the training set, then the most frequent sense according to the semantic concordance is assigned to it. If $m > 1$ senses have the same frequency then a sense among these m senses is randomly selected with probability $1/m$. If the word is polysemous and does not occur in the training corpus then a random sense is chosen as described in the guessing heuristic. They tested this heuristic on a new passage from the Brown Corpus [42] and observed the accuracy of 62.5% for all words and 50.8% for only polysemous words which was lower than expected. They checked if the passage they chose is unusual in some ways. To test with some other data, they tried leave-one-out method on the sense-tagged corpus and observed increased accuracy of 66.9% on all words and 56.4% on only polysemous words. It was also observed that assigning random senses using guessing heuristic, reduced the overall performance.

The third heuristic described is to use the prior occurrences of words together in the same sentence. The co-occurrence matrices of the senses of the words are calculated. In addition to the most frequent sense information, the semantic concordance also keeps the information about the senses that tend to occur together in the same sentence. Using this information co-occurrence matrices are created. An entry in a co-occurrence matrix represents a word sense and the other word it co-occur with in any sentence. In this heuristic, if the word is monosemous then the available sense is assigned to it. Then for all polysemous words, the co-occurrence matrix is checked to see which sense from the training corpus is more likely to occur with the surrounding words in the test sentence and the sense with the highest frequency is assigned. If more than one senses co-occur the one with the maximum frequency in the training corpus is chosen by breaking the ties with a random choice. If the polysemous word doesn't occur in the training corpus with the words in the test sentence then the most frequent sense heuristic is used to estimate the sense. Excluding the words which have been assigned a sense using guessing heuristic, the results of 68.6% on all words and 57.7% on polysemous words have been reported. These results are a little less than the most frequent sense heuristic.

[42] passage P7, an excerpt from a novel that was classified by Francis and Kucera [4] as "Imaginative Prose: Romance and Love Story"

The experiments demonstrated that a considerable improvement could be achieved by having the knowledge of sense frequencies. The authors expect that given a large semantic concordance, the co-occurrence heuristic would do better since it will capture the topical context. The authors discuss how representative these results are. These results are restricted to the Brown corpus which contains a wide selection of general English text. But in a restricted domain of discourse, most frequent sense heuristic would do much better. They also discuss how the most frequent sense heuristic would be affected by the fine distinctions made by the reference lexicon. Finally the authors discuss how these heuristics should be used. These heuristics do not consider the local context which might be useful for WSD. So given a system that exploits the local context, these heuristics could be used as a fall back strategy when the local identification fails.

6.2 Mihalcea and Faruque, 2004

The SENSELEANER system, presented by Mihalcea and Faruque [20], implements a new minimally supervised method for disambiguating all content words in a text using WordNet. The goal of this system is to use as little training data as possible and at the same time generalize the learning function well so that it can handle all content words in a text.

The system is trained on SemCor. For the words that do not occur in the training corpus, information is drawn from the WordNet. The algorithm has three steps. In the preprocessing step, the text is tokenized and annotated with part-of-speech before building models. Then the compounds and named entities are identified.

The second step is the semantic language model learning step. A separate training data set is build for each part-of-speech in SemCor. For each content word in the training set, a feature vector is created. The class label of each feature vector is a word and its sense. Different types of features are considered for each part-of-speech. If the word is a noun, the first noun, verb, or adjective before the target noun, within a window of at most five words to the left and its part-of-speech are used as features. For verbs, the first word before and the first word after the target verb and its part-of-speech are used as features. In case of adjectives, two different modes are used. In the first mode, the first noun after the target adjective, within a window of at most five words is used and in the other model the first word before and first word after the adjective and its part-of-speech is used.

Then using the Timbl memory based learning algorithm [10], a general semantic language model is learned for each part-of-speech using the above features. To annotate new text, similar feature vectors are created for all the content words. A class (i.e. a word and its sense) is predicted for each vector. If the predicted word matches with the target word then the predicted sense is assigned to the target word. Otherwise no sense is assigned to the word in this stage and the word is left for annotation in later stage. Note that although general models are built for each part-of-speech, they can't be used to disambiguate the words which are unknown to the training models.

The third step is the semantic generalization step which is especially helpful to disambiguate the unseen words. The syntactic dependencies along with the conceptual hierarchy of a word in WordNet are used in this step. The raw text from the training set is extracted and is parsed using the Link parser [42]. All the dependency pairs are stored. The part-of-speech and sense information to each open word in the dependency pair is added. A feature vector is created using the words in the dependency pair, their part-of-speech, and the references to all hypernym synsets in WordNet related to that dependency pair in case of verbs and nouns. This is used to generalize the learning which helps in disambiguating unknown words. For each dependency pair a positive feature vector is created for all the senses that appear in the training set and negative for all others. In test phase, the dependency pairs are extracted again using the Link parser. Then feature vectors are created for each dependency pair which consists of the combination of all senses of each of the word in the dependency pair. Then each feature vector is labeled positive or negative by using Timbl memory based learner [10] and the previously learned models.

The SENSELEANER system achieved an average accuracy of 64.6% in the SENSEVAL-3 English all words task. This is a significant improvement over the most frequent sense baseline of 60.9%. Similar to WN-SRAW, verbs were observed to be the most difficult word class.

Compared to WN-SRAW system this system is supervised and needs manually annotated training data. WN-SRAW uses only relatedness measures for disambiguation in contrast with SENSELEANER system that uses parsing and co-occurrences.

6.3 Navigli and Lapata, 2007

Navigli and Lapata [26] propose an unsupervised graph based algorithm for Word Sense Disambiguation which uses knowledge from a reference lexicon. The graph connectivity measures are well studied and have been considered for studying the structure of hyperlink environment and in social network analysis. In this paper the measures of graph connectivity have been compared and contrasted on the basis of how well the measures perform on the Word Sense Disambiguation task.

A sense inventory is used as background knowledge. They use WordNet and its enriched version [25]. However they claim that the method is independent of the reference lexicon. The disambiguation is done in a sentence by sentence fashion. Given a sentence, a graph $G : V \rightarrow E$ is created using nodes representing all the senses of words in that sentence. The graph is edgeless in the beginning. Then for each node $v \epsilon V$, the WordNet graph is searched in a depth first manner for the remaining $V - \{v\}$ nodes. If any of the remaining nodes is found without going too deep, all the intermediate nodes along with the connecting edges from the WordNet graph are copied into G. That is the subgraph of the WordNet graph which is relevant for disambiguating the sentence is considered. After creating the graph, the important node among the other possible nodes representing senses is identified by using graph connectivity measures. The connectivity measures can be either local or global. In local connectivity measures, each node is ranked according to the chosen connectivity measure and the top ranked sense is chosen for each word in the sentence. In case of global connectivity measures, each interpretation of the sentence is scored individually and the interpretation with highest score is chosen.

Various local and global connectivity measures that can be used to decide the *important* node in order to assign a sense have been described. These are independent of the adopted reference lexicon and the graph construction algorithm. The in-degree measure measures the importance of vertex v by its degree while in eigenvector based measures, each connection has a weight associated with it and the contribution of each node connecting to v is determined by the corresponding weight. For example, PageRank measure determines the importance of a node v recursively based on a Markov chain model. In case of KPP a vertex v is considered important if it is relatively close to all other vertices. The betweenness measure considers a node important if it is involved in a large number of paths compared to the total set of paths. In case of Maxflow measure, more the

flow conveyed from the source to the sink, the more relevant the sink is. Three global connectivity measures which consider the structure of the graph as a whole rather than the individual nodes are described. Compactness measure represents the extent of cross referencing in a graph. In graph entropy measure, the entropy of graph G is calculated as

$$H(G) = -\sum_{v \epsilon V} p(v) \log(p(v))$$

where $p(v)$ is the vertex probability determined by the degree distribution. Edge density measure calculates the edge density as the ratio of edges in a graph over the number of edges of a complete graph.

To test the performance of disambiguation, the experiments were carried using two different knowledge sources and two different corpora. To contrast and compare different connectivity measures, SemCor and SENSEVAL-3 have been used as the test corpora and WordNet as the sense inventory. They also carried experiments using extended WordNet [25] and observed a little improvement in the performance. To avoid combinatorial explosion in case of global connectivity measures, they used simulated annealing heuristics to explore the hypothesis space of interpretations. Random sense baseline is used as a lower bound and most frequent sense baseline is used as an upper bound. They report the best results of 31.8% on polysemous words in SemCor using WordNet and 40.5% using extended WordNet. WN-SRAW gives the best result of 49.7% on polysemous words in the same setting using WordNet. On SENSEVAL-3 with extended WordNet and considering polysemous as well as monosemous words, F-score of 61.9% on nouns, 62.8% on adjectives and 36.1% on verbs have been reported. WN-SRAW results are not comparable with these results because the experiments differ in the reference lexicon they use.

The experiments indicated that local measures perform better than the global measures and KPP yield the best performance in all cases. They also observed a large improvement in the performance when enriched WordNet with thousands of relatedness edges was used. InDegree and PageRank performed comparable to KPP with enriched WordNet. Finally they mention combining contrasting connectivity measures in order to improve overall performance.

6.4 Preiss, et al., 2009

As we have seen before, the most frequent sense (MFS) baseline is very hard to beat for most of the Word Sense Disambiguation systems. Preiss et al. [34] focus on refining the most frequent sense baseline by improving every stage of disambiguation such as lemmatization and part-of-speech tagging. The proposed supervised system uses a ranking algorithm and a Wikipedia Similarity measure. The system chooses an alternative answer when high confidence is observed. The MFS refining system thus benefits from a very low recall but high precision WSD system.

They start with a MFS baseline which has a F-score of 58.4% on SENSEVAL-3. At first the system is refined using the method of determining predominant sense. In 2007, McCarthy et al. [18] observed that for nouns and adjectives that occur in SemCor fewer than 5 times, the automatically determined predominant sense outperforms MFS baseline. For such words Preiss et al. switch to the sense determined by predominant sense method. This method uses an automatically created thesaurus [15] and scores a sense for a word by weighting normalized semantic similarity scores by the distributional similarity[43] scores for the neighbors of the word in a thesaurus constructed via distributional similarity. For every word, the predominant sense is chosen which maximizes the score. For the verbs that occur fewer than 5 times in SemCor, subcategorization similarity is employed rather than Lesk similarity. Combining MFS, the predominant sense method and the subcategorization method, the F-score was improved from 58.4% to 58.6%.

The performance of MFS baseline also depends upon the accuracy of the lemmatizer and part-of-speech tagger employed. They found that without any lemmatizing of the test input, the maximum *F-score* of a basic MFS system was in mid-50's and with a perfect lemmatized input it was in min-60's. They evaluated the performance of three different lemmatizers, the Lemmatizing backend of the XTAG project (XTAG Research Group, 2001), Celex (Baayen et al., 1995), and the Lemmatizing component of an enhanced TBL tagger [2] on SENSEVAL-3, and found that a simple voting system performs better than any of the individual systems. They also observed that hyphenated words were a problem for lemmatizers and removing such words increased the accuracy by 0.9%. They also evaluated the accuracy of 3 different part-of-speech taggers on SENSEVAL-3 task and observed that a simple voting achieves highest accuracy.

[43] They use the lesk similarity measure as implemented by the WordNet::Similarity package [32]

The MFS baseline may be different for different parts-of-speech. To avoid having explicit features for each part-of-speech, they implemented the "feature-focus" algorithm as described in [16]. This algorithm allows to explore large feature space efficiently. It can be used as a high-precision classifier which returns an answer only when a predictive feature that strongly predicts a particular sense is observed. It learns category indices by learning a weighted bipartite graph. The space and time efficiency is achieved by aggressively pruning edges. Though the large scale and multi class features might not be relevant in case of WSD, more features can be explored using this approach without building huge models. In this paper they use Semcor-3 and SENSEVAL English Lexical Sample data as training data and trained a sparse category index classifier with the following features: using words, lemmas and part-of-speech as tokens. The conjunction of the preceding and following unigrams and bigrams and all lemmas to the left and right in the sentence with decayed activation are used. The experiments were carried out on SENSEVALtest sets. The results are not much better than MFS baseline. They believe that the training set size is crucial and they would like to try much larger data set.

The other approach Preiss et al. use for WSD is using Wikipedia. The approach presented uses the article names and link structure within Wikipedia to find articles that are most related to a WordNet sense or context. For mapping the sense inventory to Wikipedia, they search for content words of the sense or context words in the article titles. If it is found, they use that article, and if it is not found, then the substrings are searched in the article names. The Green method as described in [27] is used to determine the importance of one node over others. The importance is determined by a measure based on the link structure within a graph. It models a random walk with certain constraints for the walker and gives the scores based on how many times a certain node was visited. The Green method produces a vector containing scores for all of the articles weighting their similarity to the initial graph nodes. The vectors are compared using the cosine of the angle between the two vectors. Using this method they got a precision of 25% and recall of 0.5% on non-monosemous words.

Preiss et al. evaluated how the systems are complementing each other using the formula in equation (9) and found that though the individual recall is very low, the systems complement each other well.

$$1 - \frac{|wrong\ in\ s_i\ and\ s_j|}{|wrong\ in\ \ s_j|} \tag{22}$$

A number of combining techniques were investigated. They observed that combining using simple voting doesn't show improvement in the MFS baseline (58.6%). Better results of 58.9% were observed when simple stacking was used.

6.5 Guo and Diab, 2009

Guo and Diab [8] suggest a modification to the unsupervised graph-based in-degree algorithm of [41] for disambiguating all content words in a text. The modifications include using JCN measure instead of LCH to find similarity between verb-verb pairs, augmenting the basic Lesk similarity measure and augmenting WordNet synsets with SemCor examples. They report the highest state-of-the-art result of 62.7% on SENSEVAL-2 using WordNet 1.7.1.

The algorithm explained in [8] requires part-of-speech tagged input. They start with the in-degree graph-based algorithm suggested by [41]. The in-degree algorithm presents the problem as a weighted graph with senses as nodes and similarity between senses as weights on edges. The In-degree measures the importance of a vertex v by summing the weights of the edges that are incident on it. The sense with the maximum in-degree is chosen as the correct sense. In the original algorithm suggested by [41], Guo and Diab explored the best similarity measure for each part-of-speech. They suggest using JCN for noun pairs, LCH for verb pairs and Lesk within adjectives and adverbs and across different parts-of-speech pairs. The authors' method differs in that it uses JCN instead of LCH for verb to verb similarity calculation based on the empirical observation on SENSEVAL-3 data.

The other modification is the extension of similarity measures used. They take their cue from the extended Lesk measure as explained by [32]. Guo and Diab's method differs in the following way. The original extended Lesk measure mainly uses this for finding similarity based on Lesk [14]. The authors employ this for other similarity measures as well. The other difference is they do not expand the target word that is to be disambiguated but only expand the neighboring words. They experimented with expanding the target word and observed that the unreliability of some of the relations is detrimental to the algorithm's performance.

The third modification is augmenting WordNet synsets with SemCor examples. Since Lesk measure

finds the overlap between the glosses, it is heavily dependent on the length of the gloss, i.e. longer the gloss, there is a high probability of more overlaps. Some of the synsets in WordNet have very brief glosses which leads to very few or no matches with the surrounding words. The idea is to augment such glosses with SemCor examples. They set a cap of 30 addition examples per synset. A total of 26875 synsets in WordNet 1.7.1 and a total of 25940 synsets in Wordnet 3.0 are augmented with SemCor examples.

The experiments were carried on three English All Words data sets, namely SENSEVAL-2, SENSEVAL-3 and SEMEVAL. The Penn Treebank tags in the test data have been used. Guo and Diab did experiments with three different versions of WordNet, 1.7.1 for ease of comparison with previous systems, 2.1 for SEMEVAL data and 3.0 to see if the performance hold across different WordNet versions. For evaluation, the scorer2 C program has been used. The authors mainly consider the results of [41] as their baseline. The authors claim that their algorithm is unsupervised though they use SemCor to augment WordNet because they don't use any annotated data.

Guo and Diab observed how independent modification and a combination of modifications contributes to the improvement. Using JCN for verb pairs instead of LCH outperformed across all data sets. Using SemCor expansions impacted more in case of SENSEVAL-2 and SEMEVAL data set. They suspect this might be because of the high number of polysemous words in case of Senseval3. They also observed that combining Semcor expansion with Expanded Lesk yielded best results for SENSEVAL-2 They observed no huge difference across different WordNet versions though they saw WordNet 3.0 yielding a slight higher results. In part-of-speech results, they observed that expanding Lesk had only impact on nouns while all parts-of-speech except for nouns were benefitted from SemCor expansion. They also observed that verbs and adverbs were benefitted by a combination of expanded Lesk and expanded SemCor.

The authors discuss that the cases of meronymy were hard for their system to capture. The lack of a method to help identify multiwords also contributed to the error. Finally the authors mention that exploring the incorporation of multiword chunks, document level lexical chains and syntactic features in the modeling of the Lesk overlap measure.

6.6 Schwartz and Gomez, 2009

Schwartz and Gomez [40] propose a word sense disambiguation method for all words using web selectors[44]. In particular they start with the method suggested to disambiguate nouns using web selectors [39] and generalize it to for other parts-of-speech. The experiments show that the context selectors may assist target selectors in case of nouns and verbs.

As we have discussed several times, the main difficulty in Word Sense Disambiguation is limited annotated training data. One approach for this is gathering more data from the web by searching for usages of monosemous relatives [21]. The other is by limiting the pre-chosen relatives/substitutes by context. Similar to this approach, Schwartz and Gomez also use context in the web search but uses a wildcard in the search rather than incorporate a knowledge-base to construct queries with pre-chosen relatives. The later half of the algorithm uses a knowledge-base through similarity and relatedness measures. The difference between this work and noun disambiguation work explained in [39] is the inclusion of selectors for adverbs.

The algorithm runs in two steps, acquisition of selectors and application of selectors. The first step of the acquisition is to construct a query with the target word replaced with a wildcard character. The words under the same part-of-speech in WordNet are matched. Then, the search query is truncated until sufficient selectors are obtained or the query size becomes too small. Assuming that the results obtained from the larger queries subsume the results obtained from smaller queries, the results from larger queries are removed. To find similarity and relatedness of the target word with the selectors, WordNet::Similarity package[45] [32] configured with WordNet 2.1 has been used. An information content measure proposed by Resnik [38] was used for target selectors of nouns and verbs and adapted Lesk algorithm [1] was used for adjectives and adverbs. The maximum similarity and relatedness between a word c_t and a selector w_s is calculated using the formula based on Resnik's word similarity [38]

$$maxsr(c_t, w_s) = \max_{c_s \epsilon w_s}[meas(c_t, c_s)]$$

where c_s is a sense of the selector and meas is a similarity or relatedness measure.

[44] Selectors are words which may take the place of another given word within its local context.

[45] http://search.cpan.org/dist/WordNet-Similarity/

The general approach of disambiguation is to find the sense of a target word which is most similar to all target selectors and most related to the context selectors. Target selectors are the selectors of the word being disambiguated and context selectors are the selectors of other words in the sentence. In other words, target selectors are similar and context selectors are related. For each selector w_s, the probability of that selector appearing in a web query q is calculated. Then the similarity and relatedness of each sense of the target word is found with each selector. The similarity and relatedness value from selectors is scaled by the ratio of web query length to the original length because the accuracy becomes weaker with a shorter query size. To get the combined Similarity/Relatedness for a given target word sense they aggregate the normalized sums from all types of selectors.

The experiments were run on the SemEval07 task7: coarse-grained English all-words[46]. The system runs on fine-grained senses but is evaluated by checking if the predicted fine-grained sense maps to the correct coarse-grained sense. The observed F-score of 76.02% show that this system performed better than the median system in the SemEval07 task. The results are just below the top unsupervised system UPV-WSD [5]. The part-of-speech results indicate that they performed quite well on adverbs and nouns and achieved noun results above MFS baseline.

Schwartz and Gomez then analyze selector acquisition. They observed that the overall percentage for which the selectors were acquired was low because it was unable to find text on the web matching local context. They observed that most selectors came from shorter queries and an average web query to pick up a selector of 6.7 words. As one shortens the query to receive more quantity the quality goes down. They refer this as the *quality selector sparsity* problem.

They also explored the influence of selector types. It was observed that noun and verb sense disambiguation benefited from all types of context selectors. Noun context selectors were helpful for adjective and adverb disambiguation. They didn't see any clear trends in case of other context selectors. They also found that the best results occurred with scale values above 1, which indicates that context selectors should be given more influence.

Finally Schwartz and Gomez conclude that in order to overcome the quality selector sparsity problem, automatic Alternative Query Construction might be useful. They also talk about refining similarity and relatedness measures for adverbs and adjectives in order to improve the results.

[46]http://nlp.cs.swarthmore.edu/semeval/tasks/index.php

7 Conclusions

The thesis starts by formalizing the algorithm of Michelizzi [19] for all-words sense disambiguation in mathematical and graph theoretic notations. The time complexity of the algorithm is also examined.

The thesis tries to resolve previous mixed conclusions about polysemy and the difficulty of disambiguation. Preiss [33] argues that polysemy is not an ideal measure of difficulty. On the other hand Daelemans [10] concludes that the fluctuations in accuracy of disambiguation largely depend on the polysemy and entropy of the ambiguous words. In this thesis we further present results on the relation of polysemy to the difficulty of disambiguation. We found a high negative correlation between polysemy and F-score suggesting that polysemy is a good measure of difficulty, the difficulty of disambiguation increasing with increased polysemy. This confirms the conclusion in Daelemans 2002 [10]. It is important to note the difference between Daelemans's method and WN-SRAW. The former method is supervised and the latter is unsupervised. WN-SRAW doesn't make use of word sense distribution information for disambiguation and hence in most of the cases difficulty doesn't depend upon the sense distribution entropy.

Not all words in a text occur with the same frequency. Words like *have* and *be* occur more frequently than words like *metamorphosis*. In order to improve overall disambiguation we need in particular to see if the most frequently occurring words are disambiguated correctly. This thesis provides the disambiguation results of frequently occurring words and presents our analysis of the difficulty in disambiguating some of these frequently occurring words. The overall results indicate that a significant percentage of word sense disambiguation error is caused by just a few highly frequent word types.

This thesis also examines to what degree the errors in parts-of-speech contribute to the overall disambiguation error. To this end, the tagged and raw text experiments were carried out. The text was part-of-speech-tagged using the widely used Brill tagger. The results show that using part-of-speech taggers before WSD is helpful. It improves performance in terms of F-score as well as time. The overall F-score was improved by 5.9%. Using part-of-speech-tagged text was especially useful for adjectives and adverbs. We saw an improvement of 9.3% in case of adjectives and 14.6% for

adverbs. For nouns the improvement was 2.7% and for verbs 7.3%.

To summarize, the experimental results provide evidence in favor of the following hypotheses.

1. The degree of difficulty in disambiguating a word is proportional to the number of senses of that word (polysemy).
2. A significant percentage of word sense disambiguation error is caused by just a few highly frequent word types.
3. Part-of-speech tagged text is disambiguated more accurately than raw text.

Inspired by the question raised by George Miller [22], asking how much context is required for WSD, we evaluate our algorithm with different context sizes in order to determine the effects of expanding the context. To better determine sources of error, we score the results in different ways. The results revealed that expanding the context window around a polysemous target word improves the recall (coverage) significantly but lowers the precision, suggesting that expanding the context may add significant noise.

WN-SRAW being unsupervised, it doesn't use any sense distribution information. It treats all senses of a word as equally likely. Therefore we expected that it would yield similar results for instances where sense1 in WordNet is not the correct sense, as for those instances in which sense1 is correct. However, our results show that if sense1 in WordNet is not the correct sense, disambiguation becomes harder. We speculate that this might be because of a sense1 bias associated with some of the similarity and relatedness measures. The other possibility is that these instances simply have an unexpected sense as a correct answer, such that the context is relatively unhelpful for disambiguation.

We also observed that given any two parts-of-speech, the more polysemous will be less accurately disambiguated, except for the case of noun and adverb. We speculate that this might be because the WordNet structure for adverbs is not very rich.

The experiments with $context\text{-}threshold$ and $pair\text{-}threshold$ show that no significant improvement with a particular threshold value can be achieved. Higher threshold values result in lower recall which results in overall lower F-score.

For our experimental results, overall $lesk_7$ and $vector_{15}$ performed best. The vector measure performed especially well for verbs and and the lesk measure performed best for adjectives and adverbs. The information content measure jcn performed best for nouns.

To summarize briefly, this research presents an extensive set of experimental results on issues central to the future direction of WSD, together with our related observations.

8 Future Work

The underlying hypothesis of the method used in this thesis is that surrounding words tell us about the intended meaning of a word. Thus for an accurate disambiguation, selecting the appropriate context is crucial. That said, choosing the appropriate context that gives the best clues for disambiguation is a hard task. WN-SRAW allows for the selection of a balanced context around the target word. But that doesn't always lead to a successful disambiguation. Varying the context window according to the situation may lead to more accurate disambiguation and is certainly an issue to be further examined.

For example, in the first sentence shown below, the context words *dinner* and *ravioli* give important clues for disambiguation, and are also very close to the target word *squash*. In this case a window size 5 would lead to correct disambiguation of *squash*. On the other hand, in the second sentence the relevant context words *player* and *basketball* are far from the target word. To have the relevant words in the context a very big window size has to be used. A window size of 21 would have the relevant words in the context, however, it may also add significant noise.

> I ate **squash** *ravioli* for *dinner* in the restaurant by the lake.
> The *basketball player*, who had been feeling very sick before, shot the **ball** into the net for the victory.

Avoiding context words that might add noise and including words that are relevant to the target would help in improving disambiguation accuracy. Therefore flexible context selection according to the situation may allow us to take full advantage of similarity and relatedness measures and thus would help in disambiguation.

It would also be interesting to experiment with left and right context in order to know which of them has greater influence on disambiguation. It is possible that the importance of context depends upon the part-of-speech of the target. As shown in the example below, for adjectives the right context would give more clues for disambiguation.

> *The blue dress was pretty.*

In general it would be useful to know which context is useful for which part-of-speech. This would in turn help in selecting context according to the part-of-speech for better disambiguation. To know which context influences the disambiguation results would also guide us in choosing the context in case of unbalanced windows.

The WN-SRAW algorithm throws out syntax and relies completely on semantics for disambiguation. The input of WN-SRAW contains only content words and doesn't contain function words. Completely throwing away the syntax sometimes makes the problem even harder.

For example, in the sentence below, if *in* is thrown away as a function word *interest* would be assigned the *banking interest* sense.

Original: *I have an interest in investment banking.*
WN-SRAW input: *have interest investment bank*

Incorporating the notion of syntax in WN-SRAW would definitely help to improve disambiguation.

There is room to make the algorithm more efficient. This can be done in various ways. In a coherent text, it is likely that the same word pairs appear a number of times. Instead of scoring it every time it occurs, it would be helpful to cache the pairs along with their similarity scores. For example, in a short excerpt below, the pair *jury#n* and *say#n* occurs 3 times and caching the scores of this pair could make the algorithm more efficient.

jury#n far#r **say#v term#n** end#n presentment#n group#n have#v overall#a charge#n election#n deserve#v praise#n thanks#n location#n manner#n election#n conduct#v september#n october#n **term#n jury#n** charge#v location#n person#n investigate#v report#n possible#a irregularity#n hard-fought#a primary#n win#v person#n only#r relative#a handful#n report#n receive#v **jury#n say#v** consider#v widespread#a interest#n election#n number#n voter#n size#n city#n **jury#n say#v** find#v georgia#n registration#n election#n law#n be#v outmoded#a inadequate#a often#r ambiguous#a

Figure 35: An excerpt from SemCor reformatted text.

Disambiguation using WN-SRAW can easily be done in parallel. Since WN-SRAW doesn't cross sentence boundaries and disambiguates text based on its local context, the input data can be par-

titioned on sentence level where each processor would get its share of sentences. The results of disambiguation can be combined easily because there is no dependence between sentences. This will help for lesk and vector measures especially when used with bigger window sizes.

The graph formulation of the algorithm may also allow us to use some of the classic graph theoretic algorithms which could help improve efficiency.

In general polysemy means more noise but at the same time it increases the possibility of finding some relatedness with the surrounding context. Monosemous instances are hard for similarity and relatedness measures, in that they are the worst case for finding relatedness because there is only one sense with which to find relatedness. It's relatively likely therefore that the method won't be able to find any relatedness with the surrounding context, resulting in low recall. To probe the performance of the measures, and as a matter of scientific curiosity, it would be interesting to see how well the measures perform only on monosemous instances.

Our results show that if sense1 in WordNet is not the correct answer, disambiguation becomes harder. Our intuition here is that this might be because of a sense1 bias associated with some of the similarity and relatedness measures. It is believed (without any experimental evidence) that first gloss in WordNet tends to be longer, increasing the probability of finding more overlaps in case of lesk. It can be easily verified by counting the number of content words in each gloss.

From the experimental results, it is known that some measures work better for certain parts-of-speech than for others. In particular, for verbs vector_{15} gave the best results and for adverbs and adjectives $\text{lesk}_{6/7/15}$ performed better than other measures by a large margin. A method that combines the best performing measures based on part-of-speech may be a natural direction in which to improve the overall performance.

We mainly focussed on lesk in the analysis of experimental results. Considering the best performance of vector measure, especially for verbs, it would be useful to experiment more with vector measure. The advantage of the vector measure over lesk is that vector goes beyond exact string matching and makes real use of context, without relying on WordNet structure to find relatedness.

We carried some experiments to see how much error part-of-speech tagging contribute to the overall error. For that we used the freely available Brill Tagger. It would be interesting to see the effect

of using other part-of-speech taggers. We use simple lemmatization techniques, as provided by WordNet. Using sophisticated lemmatization techniques might also help improve overall results. Finally it would be interesting to see the results using the resources such as extended WordNet [25].

A Appendix

A.1 Penn Treebank tags to WordNet tags mapping

$JJ \rightarrow$ ‘a'

$JJR \rightarrow$ ‘a'

$JJS \rightarrow$ ‘a'

$CD \rightarrow$ ‘a'

$RB \rightarrow$ ‘r'

$RBR \rightarrow$ ‘r'

$RBS \rightarrow$ ‘r'

$RP \rightarrow$ ‘r'

$WRB \rightarrow CLOSED$

$CC \rightarrow CLOSED$

$IN \rightarrow$ ‘r'

$DT \rightarrow CLOSED$

$PDT \rightarrow CLOSED$

$CC \rightarrow CLOSED$

$PRP\$ \rightarrow CLOSED$

$PRP \rightarrow CLOSED$

$WDT \rightarrow CLOSED$

$'WP\$' \rightarrow CLOSED$

$NN \rightarrow$ ‘n'

$NNS \rightarrow$ ‘n'

$NNP \rightarrow$ ‘n'

$NNPS \rightarrow$ ‘n'

$PRP \rightarrow CLOSED$

$WP \rightarrow CLOSED$

$EX \rightarrow CLOSED$

$VBP \rightarrow$ ‘v'

$VB \rightarrow$ ‘v'

$VBD \rightarrow$ ‘v'

$VBG \rightarrow$ ‘v'

$VBN \rightarrow$ ‘v'

$VBZ \rightarrow$ ‘v’

$VBP \rightarrow$ ‘v’

$MD \rightarrow$ ‘v’

$TO \rightarrow CLOSED$

$POS \rightarrow$ ‘$undef$’

$UH \rightarrow CLOSED$

. $\rightarrow$ ‘$undef$’

:$\rightarrow$ ‘$undef$’

, $\rightarrow$ ‘$undef$’

_ $\rightarrow$ ‘$undef$’

\$ $\rightarrow$ ‘$undef$’

($\rightarrow$ ‘$undef$’

) $\rightarrow$ ‘$undef$’

” $\rightarrow$ ‘$undef$’

$FW \rightarrow NOINFO$

$SYM \rightarrow$ ‘$undef$’

$LS \rightarrow$ ‘$undef$’

A.2 lesk and vector stoplist

a

aboard

about

above

across

after

against

all

along

alongside

although

amid

amidst

among
amongst
an
and
another
anti
any
anybody
anyone
anything
around
as
astride
at
aught
bar
barring
be
because
before
behind
below
beneath
beside
besides
between
beyond
both
but
by
circa
concerning
considering
despite

down
during
each
either
enough
everybody
everyone
except
excepting
excluding
few
fewer
following
for
form
from
having
he
her
hers
herself
him
himself
his
hisself
i
idem
if
ilk
in
including
inside
into
is

it
its
itself
like
many
me
mine
minus
more
most
myself
naught
near
neither
nobody
none
nor
nothing
notwithstanding
of
off
on
oneself
onto
opposite
or
other
otherwise
our
ourself
ourselves
outside
over
own

past
pending
per
plus
regarding
round
save
self
several
she
since
so
some
somebody
someone
something
somewhat
such
suchlike
sundry
than
that
the
thee
theirs
them
themselves
there
they
thine
this
thou
though
through

throughout
thyself
till
to
tother
toward
towards
twain
under
underneath
unless
unlike
until
up
upon
us
various
versus
via
vis-a-vis
we
what
whatall
whatever
whatsoever
when
whereas
wherewith
wherewithal
which
whichever
whichsoever
while
who

whoever
whom
whomever
whomso
whomsoever
whose
whosoever
with
within
without
worth
ye
yet
yon
yonder
you
you-all
yours
yourself
yourselves

A.3 Result Tables

A.3.1 SemCor Tables

measure	P	R	F	Att(%)	Time
path	.472	.168	.248	35.68	02:47:51
lch	.472	.168	.248	35.68	03:34:01
wup	.427	.153	.225	35.68	03:15:04
res	.395	.122	.187	3.98	02:51:50
lin	.497	.147	.227	29.56	03:02:08
jcn	.586	.207	.306	35.39	03:11:21
lesk	.498	.478	.488	95.90	08:33:20
vector	.467	.465	.466	99.58	12:20:44
random	.237	.237	.237	100.00	00:01:11
sense1	.707	.707	.707	100.00	00:01:50

Table 23: SemCor results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 145,773. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.207	.070	.105	34.05	02:35:22
lch	.207	.071	.105	34.05	03:47:50
wup	.209	.071	.106	34.05	03:24:05
res	.194	.056	.086	28.68	02:37:41
lin	.231	.063	.099	27.37	02:58:05
jcn	.201	.068	.102	33.80	03:15:28
lesk	.220	.212	.216	96.67	07:37:00
vector	.220	.219	.220	99.54	10:54:00
random	.172	.172	.172	100.00	00:00:59
sense1	.000	.000	.000	100.00	00:01:33

Table 24: SemCor results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 43,730. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.679	.325	.440	47.92	03:27:26
lch	.679	.325	.440	47.92	04:47:52
wup	.653	.313	.423	47.92	04:20:08
res	.658	.290	.403	44.13	03:26:18
lin	.721	.309	.433	42.93	03:56:37
jcn	.748	.357	.483	47.64	04:07:32
lesk	.600	.579	.589	96.58	07:49:28
vector	.571	.569	.570	99.65	11:22:33
random	.387	.387	.387	100.00	00:00:56
sense1	.764	.764	.764	100.00	00:01:30

Table 25: SemCor results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.693	.693	.693	100.00	00:56:36
lch	.693	.693	.693	100.00	01:36:23
wup	.680	.680	.680	100.00	01:23:32
res	.682	.682	.682	100.00	00:57:38
lin	.710	.710	.710	100.00	01:12:36
jcn	.726	.726	.726	100.00	01:20:06
lesk	.606	.606	.606	100.00	08:42:49
vector	.572	.572	.572	100.00	12:20:07
random	.387	.387	.387	100.00	00:01:02
sense1	.764	.764	.764	100.00	00:01:33

Table 26: SemCor results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.428	.193	.266	45.15	01:43:27
lch	.429	.194	.267	45.15	02:24:07
wup	.388	.175	.241	45.15	02:36:56
res	.360	.142	.203	39.34	01:44:27
lin	.452	.169	.246	37.36	01:56:23
jcn	.537	.240	.331	44.59	02:01:28
lesk	.498	.487	.492	97.69	09:44:47
vector	.473	.471	.472	99.58	12:46:22
random	.236	.236	.236	100.00	00:00:59
sense1	.704	.704	.704	100.00	00:01:31

Table 27: SemCor results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 145,773. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.206	.092	.127	44.37	01:50:16
lch	.206	.092	.127	44.37	03:50:02
wup	.208	.092	.128	44.37	04:03:02
res	.192	.072	.105	37.59	01:52:47
lin	.237	.085	.125	35.88	02:49:28
jcn	.220	.097	.134	44.01	02:58:52
lesk	.220	.215	.217	98.08	11:12:52
vector	.221	.220	.221	99.55	11:47:51
random	.175	.175	.175	100.00	00:01:00
sense1	.000	.000	.000	100.00	00:01:33

Table 28: SemCor results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 43,730. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.629	.351	.451	55.90	01:38:15
lch	.629	.352	.451	55.90	02:18:48
wup	.603	.337	.432	55.90	02:32:42
res	.607	.310	.410	51.003	01:39:39
lin	.670	.331	.443	49.43	01:50:58
jcn	.701	.389	.500	55.44	01:58:10
lesk	.599	.587	.593	98.05	09:29:51
vector	.576	.574	.575	99.65	10:36:50
random	.389	.389	.389	100.00	00:01:01
sense1	.764	.764	.764	100.00	00:01:32

Table 29: SemCor results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.662	.662	.662	100.00	05:50:49
lch	.662	.662	.662	100.00	07:29:24
wup	.647	.647	.647	100.00	07:26:47
res	.652	.652	.652	100.00	05:59:51
lin	.686	.686	.686	100.00	05:57:53
jcn	.703	.703	.703	100.00	06:53:59
lesk	.602	.602	.602	100.00	12:29:00
vector	.576	.576	.576	100.00	13:31:42
random	.386	.386	.386	100.00	00:00:57
sense1	.764	.764	.764	100.00	00:01:30

Table 30: SemCor results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.428	.225	.295	52.69	01:58:58
lch	.429	.226	.296	52.69	03:21:59
wup	.388	.205	.268	52.69	03:00:30
res	.353	.162	.223	46.01	02:01:45
lin	.448	.197	.273	43.91	02:35:51
jcn	.536	.280	.367	52.13	02:40:01
lesk	.502	.494	.498	98.48	09:45:05
vector	.479	.477	.478	99.59	15:08:09
random	.236	.236	.236	100.00	00:01:00
sense1	.707	.707	.707	100.00	00:01:32

Table 31: SemCor results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 145,773. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.205	.107	.140	52.19	03:26:22
lch	.205	.107	.141	52.19	04:58:35
wup	.205	.107	.140	52.19	04:29:45
res	.189	.084	.116	44.30	03:27:56
lin	.233	.099	.139	42.41	04:10:57
jcn	.212	.110	.145	51.80	04:10:10
lesk	.220	.217	.218	98.98	11:26:06
vector	.221	.220	.221	99.56	14:42:47
random	.174	.174	.174	100.00	00:01:06
sense1	.000	.000	.000	100.00	00:01:38

Table 32: SemCor results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 43,730. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.609	.377	.466	61.94	01:59:41
lch	.610	.378	.466	61.94	03:22:43
wup	.582	.361	.445	61.94	02:56:56
res	.579	.326	.417	56.35	02:00:59
lin	.647	.354	.457	54.64	02:34:18
jcn	.685	.421	.522	61.50	02:34:46
lesk	.599	.592	.595	98.95	10:06:28
vector	.581	.579	.580	99.66	13:00:18
random	.387	.387	.387	100.00	00:00:56
sense1	.764	.764	.764	100.00	00:01:29

Table 33: SemCor results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.645	.645	.645	100.00	01:59:30
lch	.645	.645	.645	100.00	03:19:17
wup	.628	.628	.628	100.00	02:55:34
res	.631	.631	.631	100.00	02:01:44
lin	.671	.671	.671	100.00	02:33:05
jcn	.692	.692	.692	100.00	02:33:36
lesk	.600	.600	.600	100.00	10:06:07
vector	.581	.581	.581	100.00	13:32:29
random	.387	.387	.387	100.00	00:00:57
sense1	.764	.764	.764	100.00	00:01:29

Table 34: SemCor results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.420	.241	.306	57.38	06:42:37
lch	.420	.241	.306	57.38	08:35:51
wup	.381	.219	.278	57.38	08:15:27
res	.341	.172	.228	5.33	06:57:42
lin	.439	.212	.286	48.20	07:18:20
jcn	.529	.301	.383	56.92	07:38:43
lesk	.501	.495	.498	98.82	12:49:41
vector	.482	.480	.481	99.59	16:30:26
random	.236	.236	.236	100.00	00:01:12
sense1	.707	.707	.707	100.00	00:01:46

Table 35: SemCor results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 145,773. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.203	.118	.149	57.86	02:52:44
lch	.203	.117	.149	57.86	04:20:06
wup	.202	.117	.148	57.86	04:15:22
res	.187	.092	.124	49.42	02:54:58
lin	.231	.110	.149	47.49	03:28:46
jcn	.213	.122	.155	57.57	03:32:27
lesk	.218	.216	.217	99.14	13:36:46
vector	.220	.219	.219	99.56	14:59:18
random	.174	.174	.174	100.00	00:01:05
sense1	.000	.000	.000	100.00	00:01:37

Table 36: SemCor results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 43,730. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.592	.390	.470	65.74	06:37:08
lch	.593	.390	.470	65.74	08:32:58
wup	.566	.372	.449	65.74	08:09:10
res	.557	.333	.417	59.81	06:46:38
lin	.629	.365	.462	58.08	07:14:50
jcn	.670	.438	.530	65.37	07:39:39
lesk	.598	.593	.595	99.17	14:55:28
vector	.583	.581	.582	99.66	16:49:48
random	.388	.388	.388	100.00	00:01:04
sense1	.764	.764	.764	100.00	00:01:38

Table 37: SemCor results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.633	.633	.633	100.00	07:50:31
lch	.633	.633	.633	100.00	09:59:32
wup	.615	.615	.615	100.00	09:20:20
res	.616	.616	.616	100.00	08:00:33
lin	.661	.661	.661	100.00	08:32:45
jcn	.684	.684	.684	100.00	08:55:21
lesk	.599	.599	.599	100.00	15:37:23
vector	.583	.583	.583	100.00	17:22:29
random	.388	.388	.388	100.00	00:00:56
sense1	.764	.764	.764	100.00	00:01:30

Table 38: SemCor results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.419	.258	.319	61.54	06:18:01
lch	.420	.259	.320	61.54	08:34:33
wup	.381	.234	.290	61.54	07:46:31
res	.334	.181	.235	54.16	06:19:09
lin	.436	.227	.298	52.02	07:15:51
jcn	.528	.323	.401	61.12	07:25:06
lesk	.501	.496	.499	99.08	13:25:58
vector	.486	.484	.485	99.59	20:13:56
random	.236	.236	.236	100.00	00:01:13
sense1	.707	.707	.707	100.00	00:01:43

Table 39: SemCor results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 145,773. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.203	.127	.156	62.49	02:54:06
lch	.202	.126	.155	62.49	05:07:20
wup	.201	.125	.154	62.49	04:23:08
res	.186	.100	.130	53.54	02:55:15
lin	.229	.118	.156	51.57	03:47:56
jcn	.213	.132	.163	62.17	03:48:47
lesk	.218	.217	.217	99.27	14:16:12
vector	.219	.219	.219	99.56	18:16:50
random	.175	.175	.175	100.00	00:01:00
sense1	.000	.000	.000	100.00	00:01:32

Table 40: SemCor results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 43,730. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.584	.403	.477	69.12	03:02:45
lch	.584	.404	.478	69.12	04:57:20
wup	.557	.385	.455	69.12	04:24:47
res	.542	.341	.418	62.89	03:05:22
lin	.617	.378	.469	61.15	03:53:35
jcn	.663	.456	.540	68.78	03:51:52
lesk	.597	.593	.595	99.33	14:23:47
vector	.586	.584	.585	99.66	20:36:28
random	.387	.387	.387	100.00	00:00:57
sense1	.764	.764	.764	100.00	00:01:29

Table 41: SemCor results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.624	.624	.624	100.00	08:20:08
lch	.625	.625	.625	100.00	10:31:27
wup	.605	.605	.605	100.00	09:36:49
res	.603	.603	.603	100.00	08:29:14
lin	.652	.652	.652	100.00	08:55:22
jcn	.679	.679	.679	100.00	09:20:05
lesk	.598	.598	.598	100.00	16:14:43
vector	.586	.586	.586	100.00	20:07:15
random	.387	.387	.387	100.00	00:01:06
sense1	.764	.764	.764	100.00	00:01:37

Table 42: SemCor results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.405	.285	.335	7.32	13:41:09
lch	.408	.287	.337	7.32	18:35:59
wup	.374	.263	.309	7.32	16:48:03
res	.302	.192	.235	63.57	13:58:16
lin	.423	.261	.323	61.77	15:55:35
jcn	.516	.362	.426	7.12	16:24:24
lesk	.498	.495	.496	99.33	27:31:56
vector	.495	.493	.494	99.59	70:09:24
random	.236	.236	.236	100.00	00:01:12
sense1	.707	.707	.707	100.00	00:01:52

Table 43: SemCor results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 145,773. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.201	.147	.170	73.41	07:55:14
lch	.198	.146	.168	73.41	11:57:42
wup	.198	.145	.168	73.41	10:23:42
res	.183	.119	.144	65.01	07:59:34
lin	.227	.143	.176	63.17	09:36:08
jcn	.210	.154	.178	73.23	09:43:03
lesk	.216	.214	.215	99.39	30:33:21
vector	.217	.216	.217	99.56	67:57:55
random	.177	.177	.177	100.00	00:01:03
sense1	.000	.000	.000	100.00	00:01:36

Table 44: SemCor results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 43,730. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.558	.425	.482	76.20	13:55:10
lch	.559	.426	.484	76.20	18:20:41
wup	.535	.408	.463	76.20	16:37:35
res	.497	.350	.411	7.49	14:00:15
lin	.587	.405	.480	69.00	15:49:18
jcn	.640	.487	.553	76.04	16:22:36
lesk	.595	.592	.594	99.49	31:27:55
vector	.593	.591	.592	99.66	68:36:04
random	.389	.389	.389	100.00	00:01:14
sense1	.764	.764	.764	100.00	00:01:49

Table 45: SemCor results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.601	.601	.601	100.00	10:15:40
lch	.602	.602	.602	100.00	14:28:01
wup	.583	.583	.583	100.00	12:47:43
res	.563	.563	.563	100.00	10:19:37
lin	.628	.628	.628	100.00	11:56:38
jcn	.664	.664	.664	100.00	12:08:43
lesk	.596	.596	.596	100.00	31:32:26
vector	.594	.594	.594	100.00	71:40:55
random	.388	.388	.388	100.00	00:01:02
sense1	.764	.764	.764	100.00	00:01:35

Table 46: SemCor results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 185,273. 'Att' is 'Attempted'.

	Nouns			Verbs			Adjectives			Adverbs		
mea	P	R	F	P	R	F	P	R	F	P	R	F
path	.486	.284	.359	.420	.109	.173	.000	.000	.000	.000	.000	.000
lch	.486	.284	.359	.420	.109	.174	.000	.000	.000	.000	.000	.000
wup	.446	.261	.329	.361	.094	.149	.000	.000	.000	.000	.000	.000
res	.431	.239	.307	.179	.027	.047	.000	.000	.000	.000	.000	.000
lin	.538	.287	.375	.231	.032	.055	.000	.000	.000	.000	.000	.000
jcn	.629	.364	.461	.432	.112	.178	.000	.000	.000	.000	.000	.000
lesk	.540	.523	.531	.396	.379	.387	.572	.547	.559	.487	.442	.464
vector	.510	.509	.509	.379	.376	.377	.526	.525	.525	.425	.424	.424
random	.262	.262	.262	.155	.155	.155	.274	.274	.274	.318	.318	.318
sense1	.749	.749	.749	.593	.593	.593	.778	.778	.778	.743	.743	.743

Table 47: SemCor results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.243	.142	.179	.126	.032	.051	.000	.000	.000	.000	.000	.000
lch	.243	.142	.179	.127	.032	.051	.000	.000	.000	.000	.000	.000
wup	.242	.142	.179	.133	.034	.054	.000	.000	.000	.000	.000	.000
res	.214	.119	.153	.121	.019	.032	.000	.000	.000	.000	.000	.000
lin	.254	.136	.177	.145	.020	.036	.000	.000	.000	.000	.000	.000
jcn	.229	.133	.168	.140	.035	.056	.000	.000	.000	.000	.000	.000
lesk	.258	.252	.255	.150	.145	.148	.307	.296	.301	.275	.252	.263
vector	.264	.263	.264	.147	.146	.147	.298	.298	.298	.279	.277	.278
random	.195	.195	.195	.125	.125	.125	.222	.222	.222	.252	.252	.252
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 48: SemCor results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.622	.411	.495	.518	.155	.238	1.00	.307	.470	1.00	.393	.564
lch	.622	.410	.494	.519	.155	.238	1.00	.307	.470	1.00	.393	.564
wup	.594	.392	.472	.471	.140	.216	1.00	.307	.470	1.00	.393	.564
res	.594	.378	.462	.402	.079	.131	1.00	.307	.470	1.00	.393	.564
lin	.674	.416	.515	.455	.082	.139	1.00	.307	.470	1.00	.393	.564
jcn	.727	.476	.575	.528	.157	.242	1.00	.307	.470	1.00	.393	.564
lesk	.630	.614	.622	.431	.412	.421	.708	.686	.697	.702	.662	.682
vector	.604	.602	.603	.411	.408	.410	.671	.671	.671	.652	.651	.652
random	.400	.400	.400	.204	.204	.204	.500	.500	.500	.590	.590	.590
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 49: SemCor results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.672	.672	.672	.570	.570	.570	.846	.846	.846	.843	.843	.843
lch	.672	.672	.672	.570	.570	.570	.846	.846	.846	.843	.843	.843
wup	.653	.653	.653	.555	.555	.555	.846	.846	.846	.843	.843	.843
res	.655	.655	.655	.558	.558	.558	.846	.846	.846	.843	.843	.843
lin	.708	.708	.708	.572	.572	.572	.846	.846	.846	.843	.843	.843
jcn	.741	.741	.741	.573	.573	.573	.846	.846	.846	.843	.843	.843
lesk	.635	.635	.635	.442	.442	.442	.712	.712	.712	.706	.706	.706
vector	.604	.604	.604	.413	.413	.413	.672	.672	.672	.652	.652	.652
random	.402	.402	.402	.204	.204	.204	.501	.501	.501	.591	.591	.591
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 50: SemCor results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.441	.313	.366	.388	.147	.213	.000	.000	.000	.000	.000	.000
lch	.442	.314	.367	.388	.147	.213	.000	.000	.000	.000	.000	.000
wup	.406	.288	.337	.331	.126	.182	.000	.000	.000	.000	.000	.000
res	.401	.272	.324	.164	.038	.062	.000	.000	.000	.000	.000	.000
lin	.500	.326	.394	.208	.043	.072	.000	.000	.000	.000	.000	.000
jcn	.583	.408	.480	.398	.150	.218	.000	.000	.000	.000	.000	.000
lesk	.541	.532	.536	.393	.382	.388	.574	.561	.567	.483	.455	.469
vector	.518	.516	.517	.384	.381	.383	.529	.528	.529	.425	.424	.424
random	.262	.262	.262	.152	.152	.152	.274	.274	.274	.322	.322	.322
sense1	.746	.746	.746	.590	.590	.590	.774	.774	.774	.742	.742	.742

Table 51: SemCor results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.249	.178	.207	.129	.048	.070	.000	.000	.000	.000	.000	.000
lch	.248	.177	.207	.129	.048	.070	.000	.000	.000	.000	.000	.000
wup	.245	.175	.204	.140	.052	.076	.000	.000	.000	.000	.000	.000
res	.218	.150	.178	.120	.029	.046	.000	.000	.000	.000	.000	.000
lin	.267	.178	.213	.149	.032	.053	.000	.000	.000	.000	.000	.000
jcn	.249	.176	.206	.167	.062	.091	.000	.000	.000	.000	.000	.000
lesk	.258	.256	.257	.149	.145	.147	.308	.303	.305	.278	.262	.270
vector	.267	.266	.266	.146	.145	.145	.302	.301	.302	.280	.278	.279
random	.193	.193	.193	.127	.127	.127	.235	.235	.235	.267	.267	.267
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 52: SemCor results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.580	.444	.503	.469	.194	.275	1.00	.307	.470	1.00	.393	.564
lch	.581	.445	.504	.469	.194	.275	1.00	.307	.470	1.00	.393	.564
wup	.554	.425	.481	.420	.174	.246	1.00	.307	.470	1.00	.393	.564
res	.557	.413	.474	.331	.089	.141	1.00	.307	.470	1.00	.393	.564
lin	.634	.456	.531	.380	.094	.151	1.00	.307	.470	1.00	.393	.564
jcn	.690	.523	.595	.478	.197	.279	1.00	.307	.470	1.00	.393	.564
lesk	.630	.622	.626	.429	.417	.423	.708	.697	.703	.696	.670	.683
vector	.610	.608	.609	.417	.413	.415	.674	.673	.673	.652	.651	.652
random	.403	.403	.403	.206	.206	.206	.502	.502	.502	.591	.591	.591
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 53: SemCor results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.621	.621	.621	.540	.540	.540	.846	.846	.846	.843	.843	.843
lch	.622	.622	.622	.540	.540	.540	.846	.846	.846	.843	.843	.843
wup	.602	.602	.602	.520	.520	.520	.846	.846	.846	.843	.843	.843
res	.609	.609	.609	.527	.527	.527	.846	.846	.846	.843	.843	.843
lin	.670	.670	.670	.546	.546	.546	.846	.846	.846	.843	.843	.843
jcn	.707	.707	.707	.544	.544	.544	.846	.846	.846	.843	.843	.843
lesk	.633	.633	.633	.436	.436	.436	.710	.710	.710	.698	.698	.698
vector	.610	.610	.610	.418	.418	.418	.674	.674	.674	.652	.652	.652
random	.400	.400	.400	.204	.204	.204	.502	.502	.502	.588	.588	.588
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 54: SemCor results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.438	.353	.391	.399	.192	.259	.000	.000	.000	.000	.000	.000
lch	.439	.353	.392	.400	.192	.259	.000	.000	.000	.000	.000	.000
wup	.406	.326	.362	.340	.164	.221	.000	.000	.000	.000	.000	.000
res	.399	.309	.348	.163	.049	.075	.000	.000	.000	.000	.000	.000
lin	.501	.375	.429	.210	.057	.090	.000	.000	.000	.000	.000	.000
jcn	.585	.465	.518	.404	.193	.261	.000	.000	.000	.000	.000	.000
lesk	.544	.539	.542	.398	.391	.394	.582	.574	.578	.473	.454	.464
vector	.523	.521	.522	.394	.391	.392	.533	.533	.533	.425	.424	.425
random	.258	.258	.258	.155	.155	.155	.279	.279	.279	.324	.324	.324
sense1	.749	.749	.749	.593	.593	.593	.778	.778	.778	.743	.743	.743

Table 55: SemCor results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.249	.201	.222	.132	.062	.085	.000	.000	.000	.000	.000	.000
lch	.249	.201	.222	.133	.063	.085	.000	.000	.000	.000	.000	.000
wup	.243	.196	.217	.141	.067	.090	.000	.000	.000	.000	.000	.000
res	.217	.170	.191	.120	.037	.056	.000	.000	.000	.000	.000	.000
lin	.266	.202	.230	.146	.042	.065	.000	.000	.000	.000	.000	.000
jcn	.247	.198	.220	.155	.073	.099	.000	.000	.000	.000	.000	.000
lesk	.257	.255	.256	.150	.148	.149	.309	.307	.308	.281	.272	.276
vector	.269	.268	.268	.146	.145	.146	.297	.296	.297	.272	.271	.272
random	.190	.190	.190	.130	.130	.130	.223	.223	.223	.260	.260	.260
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 56: SemCor results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.566	.477	.518	.463	.236	.313	1.00	.307	.470	1.00	.393	.564
lch	.567	.477	.518	.463	.236	.313	1.00	.307	.470	1.00	.393	.564
wup	.541	.456	.495	.412	.210	.278	1.00	.307	.470	1.00	.393	.564
res	.540	.442	.486	.298	.100	.150	1.00	.307	.470	1.00	.393	.564
lin	.623	.496	.552	.349	.108	.164	1.00	.307	.470	1.00	.393	.564
jcn	.683	.570	.621	.469	.238	.316	1.00	.307	.470	1.00	.393	.564
lesk	.630	.625	.628	.431	.424	.428	.709	.703	.706	.688	.675	.681
vector	.614	.613	.613	.426	.422	.424	.677	.676	.676	.653	.651	.652
random	.402	.402	.402	.205	.205	.205	.499	.499	.499	.594	.594	.594
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 57: SemCor results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.595	.595	.595	.523	.523	.523	.846	.846	.846	.843	.843	.843
lch	.595	.595	.595	.523	.523	.523	.846	.846	.846	.843	.843	.843
wup	.574	.574	.574	.497	.497	.497	.846	.846	.846	.843	.843	.843
res	.579	.579	.579	.499	.499	.499	.846	.846	.846	.843	.843	.843
lin	.650	.650	.650	.524	.524	.524	.846	.846	.846	.843	.843	.843
jcn	.694	.694	.694	.526	.526	.526	.846	.846	.846	.843	.843	.843
lesk	.632	.632	.632	.435	.435	.435	.710	.710	.710	.688	.688	.688
vector	.615	.615	.615	.427	.427	.427	.677	.677	.677	.652	.652	.652
random	.403	.403	.403	.203	.203	.203	.499	.499	.499	.589	.589	.589
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 58: SemCor results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.430	.367	.396	.394	.221	.283	.000	.000	.000	.000	.000	.000
lch	.431	.368	.397	.395	.221	.283	.000	.000	.000	.000	.000	.000
wup	.401	.342	.369	.333	.187	.239	.000	.000	.000	.000	.000	.000
res	.390	.323	.353	.160	.058	.085	.000	.000	.000	.000	.000	.000
lin	.497	.400	.443	.208	.068	.103	.000	.000	.000	.000	.000	.000
jcn	.581	.491	.532	.399	.223	.286	.000	.000	.000	.000	.000	.000
lesk	.545	.541	.543	.395	.389	.392	.583	.578	.580	.474	.460	.467
vector	.526	.525	.525	.398	.394	.396	.536	.536	.536	.423	.422	.422
random	.261	.261	.261	.155	.155	.155	.276	.276	.276	.316	.316	.316
sense1	.749	.749	.749	.593	.593	.593	.778	.778	.778	.743	.743	.743

Table 59: SemCor results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.248	.214	.230	.136	.075	.097	.000	.000	.000	.000	.000	.000
lch	.247	.213	.229	.136	.075	.097	.000	.000	.000	.000	.000	.000
wup	.242	.210	.225	.140	.077	.100	.000	.000	.000	.000	.000	.000
res	.218	.184	.200	.118	.044	.064	.000	.000	.000	.000	.000	.000
lin	.268	.221	.242	.147	.050	.075	.000	.000	.000	.000	.000	.000
jcn	.251	.216	.232	.155	.085	.110	.000	.000	.000	.000	.000	.000
lesk	.256	.255	.255	.148	.146	.147	.307	.305	.306	.283	.276	.280
vector	.266	.266	.266	.147	.146	.147	.291	.290	.291	.274	.272	.273
random	.194	.194	.194	.127	.127	.127	.230	.230	.230	.258	.258	.258
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 60: SemCor results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.553	.488	.519	.449	.263	.331	1.00	.307	.470	1.00	.393	.564
lch	.554	.489	.519	.449	.263	.332	1.00	.307	.470	1.00	.393	.564
wup	.531	.468	.498	.395	.231	.292	1.00	.307	.470	1.00	.393	.564
res	.526	.453	.487	.277	.109	.156	1.00	.307	.470	1.00	.393	.564
lin	.613	.516	.560	.326	.118	.173	1.00	.307	.470	1.00	.393	.564
jcn	.674	.590	.629	.453	.264	.334	1.00	.307	.470	1.00	.393	.564
lesk	.630	.627	.629	.428	.422	.425	.709	.705	.707	.686	.676	.681
vector	.617	.615	.616	.429	.426	.427	.679	.678	.678	.651	.650	.651
random	.402	.402	.402	.207	.207	.207	.502	.502	.502	.589	.589	.589
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 61: SemCor results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.578	.578	.578	.506	.506	.506	.846	.846	.846	.843	.843	.843
lch	.579	.579	.579	.506	.506	.506	.846	.846	.846	.843	.843	.843
wup	.558	.558	.558	.475	.475	.475	.846	.846	.846	.843	.843	.843
res	.560	.560	.560	.475	.475	.475	.846	.846	.846	.843	.843	.843
lin	.638	.638	.638	.505	.505	.505	.846	.846	.846	.843	.843	.843
jcn	.686	.686	.686	.510	.510	.510	.846	.846	.846	.843	.843	.843
lesk	.632	.632	.632	.431	.431	.431	.710	.710	.710	.686	.686	.686
vector	.617	.617	.617	.430	.430	.430	.679	.679	.679	.651	.651	.651
random	.402	.402	.402	.205	.205	.205	.504	.504	.504	.591	.591	.591
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 62: SemCor results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.431	.386	.407	.392	.247	.303	.000	.000	.000	.000	.000	.000
lch	.432	.387	.408	.393	.248	.304	.000	.000	.000	.000	.000	.000
wup	.401	.360	.379	.333	.210	.258	.000	.000	.000	.000	.000	.000
res	.386	.337	.360	.158	.066	.093	.000	.000	.000	.000	.000	.000
lin	.499	.425	.459	.207	.079	.114	.000	.000	.000	.000	.000	.000
jcn	.586	.521	.551	.396	.249	.306	.000	.000	.000	.000	.000	.000
lesk	.545	.542	.544	.393	.388	.390	.584	.581	.582	.474	.464	.469
vector	.530	.528	.529	.402	.399	.401	.542	.541	.542	.422	.420	.421
random	.260	.260	.260	.155	.155	.155	.278	.278	.278	.321	.321	.321
sense1	.749	.749	.749	.593	.593	.593	.778	.778	.778	.743	.743	.743

Table 63: SemCor results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.252	.227	.239	.135	.084	.104	.000	.000	.000	.000	.000	.000
lch	.250	.225	.237	.134	.084	.104	.000	.000	.000	.000	.000	.000
wup	.245	.221	.232	.140	.087	.107	.000	.000	.000	.000	.000	.000
res	.223	.197	.209	.115	.049	.069	.000	.000	.000	.000	.000	.000
lin	.269	.234	.250	.145	.058	.083	.000	.000	.000	.000	.000	.000
jcn	.253	.227	.239	.158	.098	.121	.000	.000	.000	.000	.000	.000
lesk	.256	.255	.256	.147	.145	.146	.303	.302	.303	.292	.287	.289
vector	.266	.265	.266	.148	.147	.148	.283	.283	.283	.274	.272	.273
random	.193	.193	.193	.127	.127	.127	.230	.230	.230	.265	.265	.265
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 64: SemCor results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.548	.503	.525	.443	.289	.350	1.00	.307	.470	1.00	.393	.564
lch	.549	.504	.526	.444	.289	.350	1.00	.307	.470	1.00	.393	.564
wup	.525	.482	.503	.391	.255	.308	1.00	.307	.470	1.00	.393	.564
res	.517	.465	.489	.262	.116	.161	1.00	.307	.470	1.00	.393	.564
lin	.609	.536	.570	.314	.128	.182	1.00	.307	.470	1.00	.393	.564
jcn	.673	.614	.642	.448	.291	.353	1.00	.307	.470	1.00	.393	.564
lesk	.631	.628	.629	.426	.421	.424	.709	.706	.708	.684	.677	.681
vector	.619	.618	.618	.433	.430	.431	.683	.682	.682	.650	.649	.650
random	.402	.402	.402	.204	.204	.204	.500	.500	.500	.589	.589	.589
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 65: SemCor results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.567	.567	.567	.493	.493	.493	.846	.846	.846	.843	.843	.843
lch	.568	.568	.568	.494	.494	.494	.846	.846	.846	.843	.843	.843
wup	.545	.545	.545	.459	.459	.459	.846	.846	.846	.843	.843	.843
res	.543	.543	.543	.454	.454	.454	.846	.846	.846	.843	.843	.843
lin	.629	.629	.629	.488	.488	.488	.846	.846	.846	.843	.843	.843
jcn	.682	.682	.682	.497	.497	.497	.846	.846	.846	.843	.843	.843
lesk	.632	.632	.632	.429	.429	.429	.710	.710	.710	.685	.685	.685
vector	.620	.620	.620	.434	.434	.434	.683	.683	.683	.650	.650	.650
random	.401	.401	.401	.203	.203	.203	.504	.504	.504	.590	.590	.590
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 66: SemCor results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.419	.401	.409	.380	.316	.345	.000	.000	.000	.000	.000	.000
lch	.421	.403	.412	.383	.318	.347	.000	.000	.000	.000	.000	.000
wup	.400	.383	.392	.324	.269	.294	.000	.000	.000	.000	.000	.000
res	.361	.342	.351	.155	.095	.118	.000	.000	.000	.000	.000	.000
lin	.503	.472	.487	.210	.120	.153	.000	.000	.000	.000	.000	.000
jcn	.588	.561	.574	.382	.316	.346	.000	.000	.000	.000	.000	.000
lesk	.545	.543	.544	.386	.382	.384	.581	.580	.580	.466	.461	.464
vector	.540	.538	.539	.412	.409	.410	.553	.553	.553	.421	.419	.420
random	.258	.258	.258	.154	.154	.154	.282	.282	.282	.317	.317	.317
sense1	.749	.749	.749	.593	.593	.593	.778	.778	.778	.743	.743	.743

Table 67: SemCor results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.254	.245	.249	.141	.117	.127	.000	.000	.000	.000	.000	.000
lch	.250	.240	.245	.141	.117	.128	.000	.000	.000	.000	.000	.000
wup	.248	.239	.243	.143	.118	.129	.000	.000	.000	.000	.000	.000
res	.230	.219	.224	.116	.074	.090	.000	.000	.000	.000	.000	.000
lin	.277	.262	.269	.151	.090	.113	.000	.000	.000	.000	.000	.000
jcn	.259	.248	.253	.156	.129	.141	.000	.000	.000	.000	.000	.000
lesk	.255	.254	.254	.144	.142	.143	.297	.297	.297	.291	.288	.289
vector	.265	.264	.264	.144	.143	.143	.281	.281	.281	.279	.277	.278
random	.196	.196	.196	.134	.134	.134	.221	.221	.221	.264	.264	.264
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 68: SemCor results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.533	.515	.524	.420	.352	.383	1.00	.307	.470	1.00	.393	.564
lch	.535	.517	.525	.422	.354	.385	1.00	.307	.470	1.00	.393	.564
wup	.519	.501	.510	.368	.309	.336	1.00	.307	.470	1.00	.393	.564
res	.489	.469	.479	.230	.145	.178	1.00	.307	.470	1.00	.393	.564
lin	.605	.574	.589	.284	.167	.210	1.00	.307	.470	1.00	.393	.564
jcn	.670	.646	.658	.423	.354	.386	1.00	.307	.470	1.00	.393	.564
lesk	.630	.628	.629	.421	.416	.419	.707	.706	.707	.679	.675	.677
vector	.628	.626	.627	.442	.438	.440	.690	.690	.690	.650	.649	.649
random	.404	.404	.404	.206	.206	.206	.503	.503	.503	.590	.590	.590
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 69: SemCor results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.542	.542	.542	.448	.448	.448	.846	.846	.846	.843	.843	.843
lch	.543	.543	.543	.450	.450	.450	.846	.846	.846	.843	.843	.843
wup	.528	.528	.528	.405	.405	.405	.846	.846	.846	.843	.843	.843
res	.502	.502	.502	.374	.374	.374	.846	.846	.846	.843	.843	.843
lin	.614	.614	.614	.425	.425	.425	.846	.846	.846	.843	.843	.843
jcn	.675	.675	.675	.452	.452	.452	.846	.846	.846	.843	.843	.843
lesk	.631	.631	.631	.423	.423	.423	.707	.707	.707	.679	.679	.679
vector	.628	.628	.628	.443	.443	.443	.691	.691	.691	.649	.649	.649
random	.402	.402	.402	.205	.205	.205	.499	.499	.499	.590	.590	.590
sense1	.798	.798	.798	.616	.616	.616	.846	.846	.846	.843	.843	.843

Table 70: SemCor results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

A.3.2 SENSEVAL-2 Tables

measure	P	R	F	Att(%)	Time
path	.360	.114	.173	31.74	00:00:42
lch	.361	.115	.174	31.74	00:01:48
wup	.339	.107	.163	31.74	00:01:17
res	.320	.087	.137	27.12	00:00:42
lin	.425	.089	.147	2.82	00:00:54
jcn	.509	.140	.220	27.56	00:01:11
lesk	.458	.435	.447	94.99	00:04:31
vector	.446	.445	.446	99.78	00:05:12
random	.246	.246	.246	100.00	00:00:09
sense1	.588	.588	.588	100.00	00:00:09

Table 71: SENSEVAL-2 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,796. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.268	.085	.129	31.78	00:00:42
lch	.272	.086	.131	31.78	00:01:08
wup	.259	.082	.125	31.78	00:01:00
res	.226	.060	.095	26.46	00:00:43
lin	.232	.047	.078	2.08	00:00:53
jcn	.165	.044	.069	26.60	00:00:59
lesk	.253	.242	.247	95.74	00:03:07
vector	.292	.291	.291	99.87	00:04:20
random	.210	.210	.210	100.00	00:00:08
sense1	.000	.000	.000	100.00	00:00:09

Table 72: SENSEVAL-2 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 752. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.629	.280	.388	44.56	00:02:57
lch	.630	.281	.388	44.56	00:04:53
wup	.617	.275	.380	44.56	00:03:56
res	.635	.261	.370	41.002	00:03:00
lin	.732	.262	.386	35.80	00:03:23
jcn	.732	.301	.426	41.11	00:04:13
lesk	.563	.539	.551	95.84	00:06:28
vector	.547	.546	.546	99.82	00:06:18
random	.411	.411	.411	100.00	00:00:09
sense1	.665	.665	.665	100.00	00:00:10

Table 73: SENSEVAL-2 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.610	.610	.610	100.00	00:00:42
lch	.610	.610	.610	100.00	00:01:07
wup	.604	.604	.604	100.00	00:01:00
res	.607	.607	.607	100.00	00:00:43
lin	.638	.638	.638	100.00	00:00:53
jcn	.647	.647	.647	100.00	00:00:57
lesk	.574	.574	.574	100.00	00:03:04
vector	.554	.554	.554	100.00	00:04:10
random	.413	.413	.413	100.00	00:00:08
sense1	.667	.667	.667	100.00	00:00:08

Table 74: SENSEVAL-2 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.351	.143	.203	4.76	00:03:47
lch	.355	.145	.206	4.76	00:04:33
wup	.337	.138	.195	4.76	00:04:34
res	.328	.115	.171	35.13	00:03:50
lin	.414	.114	.179	27.56	00:03:59
jcn	.470	.167	.247	35.52	00:04:22
lesk	.459	.448	.453	97.55	00:06:35
vector	.438	.437	.438	99.78	00:06:22
random	.239	.239	.239	100.00	00:00:08
sense1	.585	.585	.585	100.00	00:00:09

Table 75: SENSEVAL-2 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,796. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.272	.110	.157	4.56	00:01:13
lch	.279	.113	.161	4.56	00:01:40
wup	.262	.106	.151	4.56	00:01:49
res	.230	.078	.117	34.18	00:01:13
lin	.230	.061	.097	26.60	00:01:23
jcn	.208	.072	.107	34.57	00:01:29
lesk	.278	.271	.274	97.74	00:04:51
vector	.296	.295	.295	99.87	00:05:00
random	.206	.206	.206	100.00	00:00:08
sense1	.000	.000	.000	100.00	00:00:09

Table 76: SENSEVAL-2 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 752. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.590	.309	.405	52.35	00:03:59
lch	.593	.310	.407	52.35	00:04:27
wup	.582	.305	.400	52.35	00:04:27
res	.604	.288	.390	47.65	00:03:53
lin	.688	.285	.403	41.37	00:03:55
jcn	.681	.326	.441	47.83	00:04:16
lesk	.568	.557	.562	97.96	00:06:26
vector	.547	.546	.547	99.82	00:06:04
random	.395	.395	.395	100.00	00:00:09
sense1	.665	.665	.665	100.00	00:00:09

Table 77: SENSEVAL-2 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.591	.591	.591	100.00	00:01:17
lch	.592	.592	.592	100.00	00:01:44
wup	.585	.585	.585	100.00	00:01:55
res	.592	.592	.592	100.00	00:01:17
lin	.627	.627	.627	100.00	00:01:26
jcn	.631	.631	.631	100.00	00:01:36
lesk	.578	.578	.578	100.00	00:04:53
vector	.555	.555	.555	100.00	00:05:06
random	.408	.408	.408	100.00	00:00:08
sense1	.667	.667	.667	100.00	00:00:08

Table 78: SENSEVAL-2 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.352	.167	.227	47.49	00:03:27
lch	.355	.169	.229	47.49	00:04:20
wup	.335	.159	.216	47.49	00:04:00
res	.321	.131	.186	4.76	00:03:33
lin	.389	.127	.191	32.63	00:03:46
jcn	.474	.200	.282	42.32	00:04:06
lesk	.468	.462	.465	98.83	00:06:02
vector	.461	.460	.461	99.78	00:06:42
random	.266	.266	.266	100.00	00:00:09
sense1	.588	.588	.588	100.00	00:00:10

Table 79: SENSEVAL-2 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,796. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.250	.117	.159	46.81	00:03:22
lch	.253	.118	.161	46.81	00:04:13
wup	.244	.114	.156	46.81	00:03:58
res	.207	.081	.117	39.23	00:03:24
lin	.207	.066	.101	32.18	00:03:43
jcn	.201	.085	.120	42.29	00:04:02
lesk	.278	.275	.277	99.07	00:06:01
vector	.304	.303	.303	99.87	00:06:35
random	.203	.203	.203	100.00	00:00:08
sense1	.000	.000	.000	100.00	00:00:09

Table 80: SENSEVAL-2 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 752. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.568	.329	.416	57.88	00:03:22
lch	.570	.330	.418	57.88	00:04:17
wup	.557	.322	.408	57.88	00:03:50
res	.575	.300	.395	52.21	00:03:25
lin	.651	.297	.408	45.58	00:03:36
jcn	.663	.356	.463	53.67	00:03:52
lesk	.574	.568	.571	99.03	00:06:03
vector	.566	.565	.566	99.82	00:06:27
random	.405	.405	.405	100.00	00:00:08
sense1	.667	.667	.667	100.00	00:00:09

Table 81: SENSEVAL-2 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.575	.575	.575	100.00	00:01:25
lch	.577	.577	.577	100.00	00:02:12
wup	.569	.569	.569	100.00	00:02:00
res	.576	.576	.576	100.00	00:01:23
lin	.615	.615	.615	100.00	00:01:46
jcn	.628	.628	.628	100.00	00:01:54
lesk	.575	.575	.575	100.00	00:04:56
vector	.566	.566	.566	100.00	00:06:04
random	.407	.407	.407	100.00	00:00:08
sense1	.667	.667	.667	100.00	00:00:09

Table 82: SENSEVAL-2 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.346	.182	.239	52.67	00:04:04
lch	.349	.184	.241	52.67	00:05:01
wup	.319	.168	.220	52.67	00:04:50
res	.321	.146	.201	45.43	00:04:02
lin	.398	.151	.219	37.92	00:04:16
jcn	.475	.228	.308	47.94	00:04:43
lesk	.474	.469	.471	98.94	00:07:28
vector	.459	.458	.459	99.78	00:07:44
random	.239	.239	.239	100.00	00:00:08
sense1	.588	.588	.588	100.00	00:00:09

Table 83: SENSEVAL-2 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,796. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.239	.126	.165	52.93	00:01:52
lch	.244	.129	.169	52.93	00:02:43
wup	.231	.122	.160	52.93	00:02:46
res	.205	.092	.127	44.81	00:01:53
lin	.211	.081	.117	38.43	00:02:10
jcn	.203	.100	.134	49.07	00:02:22
lesk	.290	.287	.288	99.20	00:06:24
vector	.296	.295	.295	99.87	00:07:04
random	.202	.202	.202	100.00	00:00:08
sense1	.000	.000	.000	100.00	00:00:09

Table 84: SENSEVAL-2 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 752. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.551	.342	.422	62.12	00:03:59
lch	.551	.342	.422	62.12	00:04:50
wup	.531	.330	.407	62.12	00:04:44
res	.558	.313	.401	56.02	00:04:01
lin	.632	.314	.420	49.69	00:04:08
jcn	.649	.378	.478	58.23	00:04:30
lesk	.579	.573	.576	99.12	00:07:15
vector	.564	.563	.564	99.82	00:07:33
random	.404	.404	.404	100.00	00:00:08
sense1	.667	.667	.667	100.00	00:00:09

Table 85: SENSEVAL-2 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.566	.566	.566	100.00	00:04:56
lch	.568	.568	.568	100.00	00:06:35
wup	.554	.554	.554	100.00	00:06:18
res	.569	.569	.569	100.00	00:05:06
lin	.612	.612	.612	100.00	00:05:14
jcn	.628	.628	.628	100.00	00:06:10
lesk	.580	.580	.580	100.00	00:08:09
vector	.565	.565	.565	100.00	00:08:35
random	.415	.415	.415	100.00	00:00:09
sense1	.667	.667	.667	100.00	00:00:09

Table 86: SENSEVAL-2 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.364	.207	.263	56.79	00:01:58
lch	.364	.207	.263	56.79	00:03:15
wup	.330	.188	.239	56.79	00:02:56
res	.342	.167	.224	48.83	00:01:58
lin	.401	.164	.233	4.81	00:02:32
jcn	.474	.246	.324	51.78	00:02:47
lesk	.472	.468	.470	99.22	00:06:36
vector	.464	.463	.464	99.78	00:08:02
random	.259	.259	.259	100.00	00:00:08
sense1	.588	.588	.588	100.00	00:00:09

Table 87: SENSEVAL-2 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,796. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.256	.146	.186	57.18	00:01:56
lch	.256	.146	.186	57.18	00:03:16
wup	.244	.140	.178	57.18	00:02:54
res	.223	.106	.144	47.74	00:01:57
lin	.220	.092	.129	41.76	00:02:32
jcn	.207	.110	.144	53.32	00:02:42
lesk	.278	.277	.277	99.60	00:07:19
vector	.294	.294	.294	99.87	00:08:43
random	.194	.194	.194	100.00	00:00:08
sense1	.000	.000	.000	100.00	00:00:08

Table 88: SENSEVAL-2 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 752. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.554	.362	.438	65.22	00:01:55
lch	.554	.361	.437	65.22	00:03:06
wup	.529	.345	.418	65.22	00:02:48
res	.559	.328	.414	58.72	00:01:53
lin	.622	.324	.426	52.12	00:02:24
jcn	.645	.394	.489	61.15	00:02:39
lesk	.576	.573	.574	99.38	00:06:31
vector	.568	.567	.568	99.82	00:07:52
random	.407	.407	.407	100.00	00:00:08
sense1	.667	.667	.667	100.00	00:00:08

Table 89: SENSEVAL-2 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.566	.566	.566	100.00	00:05:34
lch	.566	.566	.566	100.00	00:07:02
wup	.550	.550	.550	100.00	00:06:26
res	.568	.568	.568	100.00	00:05:43
lin	.610	.610	.610	100.00	00:06:02
jcn	.625	.625	.625	100.00	00:06:33
lesk	.577	.577	.577	100.00	00:08:12
vector	.569	.569	.569	100.00	00:09:31
random	.412	.412	.412	100.00	00:00:10
sense1	.667	.667	.667	100.00	00:00:09

Table 90: SENSEVAL-2 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.340	.226	.271	66.37	00:03:32
lch	.338	.224	.270	66.37	00:06:04
wup	.315	.209	.251	66.37	00:05:16
res	.299	.176	.222	58.85	00:03:41
lin	.375	.196	.257	52.28	00:04:42
jcn	.464	.291	.357	62.69	00:05:10
lesk	.470	.468	.469	99.55	00:13:16
vector	.488	.487	.487	99.78	00:27:33
random	.247	.247	.247	100.00	00:00:08
sense1	.588	.588	.588	100.00	00:00:08

Table 91: SENSEVAL-2 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,796. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.227	.158	.186	69.81	00:05:31
lch	.229	.160	.188	69.81	00:07:58
wup	.211	.148	.174	69.81	00:07:02
res	.182	.112	.138	61.30	00:05:31
lin	.193	.109	.139	56.38	00:06:23
jcn	.192	.129	.154	67.02	00:07:01
lesk	.284	.283	.283	99.87	00:14:00
vector	.305	.305	.305	99.87	00:28:18
random	.197	.197	.197	100.00	00:00:09
sense1	.000	.000	.000	100.00	00:00:09

Table 92: SENSEVAL-2 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 752. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.515	.376	.435	73.01	00:03:27
lch	.515	.376	.435	73.01	00:05:51
wup	.496	.362	.419	73.01	00:05:13
res	.502	.335	.401	66.68	00:03:26
lin	.572	.350	.435	61.28	00:04:26
jcn	.613	.429	.505	69.96	00:04:51
lesk	.575	.573	.574	99.65	00:12:43
vector	.586	.585	.586	99.82	00:26:08
random	.408	.408	.408	100.00	00:00:08
sense1	.667	.667	.667	100.00	00:00:08

Table 93: SENSEVAL-2 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.546	.546	.546	100.00	00:03:30
lch	.546	.546	.546	100.00	00:06:00
wup	.532	.532	.532	100.00	00:05:15
res	.539	.539	.539	100.00	00:03:31
lin	.593	.593	.593	100.00	00:04:36
jcn	.619	.619	.619	100.00	00:05:09
lesk	.577	.577	.577	100.00	00:13:21
vector	.587	.587	.587	100.00	00:28:01
random	.405	.405	.405	100.00	00:00:09
sense1	.667	.667	.667	100.00	00:00:09

Table 94: SENSEVAL-2 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 2,260. 'Att' is 'Attempted'.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.355	.199	.255	.379	.091	.147	.000	.000	.000	.000	.000	.000
lch	.357	.200	.256	.379	.091	.147	.000	.000	.000	.000	.000	.000
wup	.337	.189	.242	.345	.083	.134	.000	.000	.000	.000	.000	.000
res	.337	.176	.232	.206	.027	.048	.000	.000	.000	.000	.000	.000
lin	.448	.181	.258	.261	.025	.046	.000	.000	.000	.000	.000	.000
jcn	.543	.255	.347	.395	.094	.151	.000	.000	.000	.000	.000	.000
lesk	.498	.482	.490	.302	.287	.294	.575	.529	.551	.487	.450	.468
vector	.477	.476	.476	.305	.304	.304	.595	.595	.595	.412	.409	.411
random	.242	.242	.242	.183	.183	.183	.330	.330	.330	.281	.281	.281
sense1	.634	.634	.634	.422	.422	.422	.661	.661	.661	.696	.696	.696

Table 95: SENSEVAL-2 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.305	.177	.224	.169	.038	.063	.000	.000	.000	.000	.000	.000
lch	.310	.181	.228	.169	.038	.063	.000	.000	.000	.000	.000	.000
wup	.305	.177	.224	.138	.031	.051	.000	.000	.000	.000	.000	.000
res	.264	.144	.186	.056	.007	.012	.000	.000	.000	.000	.000	.000
lin	.256	.107	.151	.115	.010	.019	.000	.000	.000	.000	.000	.000
jcn	.204	.094	.128	.079	.017	.029	.000	.000	.000	.000	.000	.000
lesk	.321	.308	.314	.163	.157	.160	.290	.272	.281	.280	.264	.272
vector	.348	.348	.348	.200	.199	.200	.360	.360	.360	.321	.321	.321
random	.217	.217	.217	.147	.147	.147	.316	.316	.316	.283	.283	.283
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 96: SENSEVAL-2 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.567	.371	.449	.432	.112	.178	1.00	.194	.325	1.00	.372	.542
lch	.569	.372	.450	.432	.112	.178	1.00	.194	.325	1.00	.372	.542
wup	.556	.363	.439	.402	.104	.165	1.00	.194	.325	1.00	.372	.542
res	.570	.360	.441	.329	.049	.085	1.00	.194	.325	1.00	.372	.542
lin	.677	.362	.472	.424	.049	.088	1.00	.194	.325	1.00	.372	.542
jcn	.713	.415	.525	.442	.112	.179	1.00	.194	.325	1.00	.372	.542
lesk	.610	.594	.602	.326	.310	.318	.650	.607	.628	.687	.650	.668
vector	.591	.590	.591	.318	.316	.317	.659	.659	.659	.627	.625	.626
random	.443	.443	.443	.200	.200	.200	.477	.477	.477	.578	.578	.578
sense1	.715	.715	.715	.436	.436	.436	.719	.719	.719	.809	.809	.809

Table 97: SENSEVAL-2 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.603	.603	.603	.420	.420	.420	.727	.727	.727	.809	.809	.809
lch	.604	.604	.604	.420	.420	.420	.727	.727	.727	.809	.809	.809
wup	.595	.595	.595	.411	.411	.411	.727	.727	.727	.809	.809	.809
res	.604	.604	.604	.407	.407	.407	.727	.727	.727	.809	.809	.809
lin	.664	.664	.664	.420	.420	.420	.727	.727	.727	.809	.809	.809
jcn	.682	.682	.682	.422	.422	.422	.727	.727	.727	.809	.809	.809
lesk	.615	.615	.615	.342	.342	.342	.671	.671	.671	.697	.697	.697
vector	.596	.596	.596	.324	.324	.324	.674	.674	.674	.632	.632	.632
random	.438	.438	.438	.212	.212	.212	.487	.487	.487	.574	.574	.574
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 98: SENSEVAL-2 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.351	.247	.290	.350	.119	.177	.000	.000	.000	.000	.000	.000
lch	.357	.250	.294	.350	.119	.177	.000	.000	.000	.000	.000	.000
wup	.348	.244	.287	.301	.102	.152	.000	.000	.000	.000	.000	.000
res	.356	.238	.285	.157	.029	.049	.000	.000	.000	.000	.000	.000
lin	.453	.240	.314	.154	.021	.037	.000	.000	.000	.000	.000	.000
jcn	.517	.305	.383	.331	.110	.165	.000	.000	.000	.000	.000	.000
lesk	.500	.492	.496	.292	.285	.288	.570	.550	.560	.518	.497	.507
vector	.477	.476	.476	.305	.304	.304	.556	.556	.556	.400	.398	.399
random	.258	.258	.258	.177	.177	.177	.273	.273	.273	.263	.263	.263
sense1	.631	.631	.631	.420	.420	.420	.652	.652	.652	.696	.696	.696

Table 99: SENSEVAL-2 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.330	.237	.276	.133	.042	.064	.000	.000	.000	.000	.000	.000
lch	.340	.244	.284	.133	.042	.064	.000	.000	.000	.000	.000	.000
wup	.330	.237	.276	.100	.031	.048	.000	.000	.000	.000	.000	.000
res	.272	.187	.222	.059	.010	.018	.000	.000	.000	.000	.000	.000
lin	.268	.147	.190	.056	.007	.012	.000	.000	.000	.000	.000	.000
jcn	.254	.147	.186	.115	.035	.054	.000	.000	.000	.000	.000	.000
lesk	.348	.341	.345	.168	.164	.166	.345	.333	.339	.327	.321	.324
vector	.368	.368	.368	.207	.206	.207	.325	.325	.325	.302	.302	.302
random	.197	.197	.197	.161	.161	.161	.325	.325	.325	.245	.245	.245
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 100: SENSEVAL-2 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.538	.417	.470	.406	.143	.212	1.00	.194	.325	1.00	.372	.542
lch	.542	.420	.473	.406	.143	.212	1.00	.194	.325	1.00	.372	.542
wup	.538	.417	.470	.356	.126	.186	1.00	.194	.325	1.00	.372	.542
res	.558	.417	.478	.252	.051	.085	1.00	.194	.325	1.00	.372	.542
lin	.649	.412	.504	.291	.045	.078	1.00	.194	.325	1.00	.372	.542
jcn	.675	.460	.547	.373	.130	.192	1.00	.194	.325	1.00	.372	.542
lesk	.617	.609	.613	.320	.312	.316	.658	.638	.648	.703	.682	.692
vector	.596	.595	.595	.325	.324	.325	.643	.643	.643	.627	.625	.626
random	.409	.409	.409	.204	.204	.204	.472	.472	.472	.574	.574	.574
sense1	.715	.715	.715	.436	.436	.436	.719	.719	.719	.809	.809	.809

Table 101: SENSEVAL-2 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.571	.571	.571	.401	.401	.401	.727	.727	.727	.809	.809	.809
lch	.574	.574	.574	.401	.401	.401	.727	.727	.727	.809	.809	.809
wup	.569	.569	.569	.383	.383	.383	.727	.727	.727	.809	.809	.809
res	.580	.580	.580	.387	.387	.387	.727	.727	.727	.809	.809	.809
lin	.650	.650	.650	.399	.399	.399	.727	.727	.727	.809	.809	.809
jcn	.661	.661	.661	.393	.393	.393	.727	.727	.727	.809	.809	.809
lesk	.623	.623	.623	.330	.330	.330	.681	.681	.681	.711	.711	.711
vector	.601	.601	.601	.330	.330	.330	.662	.662	.662	.632	.632	.632
random	.441	.441	.441	.196	.196	.196	.487	.487	.487	.552	.552	.552
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 102: SENSEVAL-2 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.349	.279	.310	.359	.154	.215	.000	.000	.000	.000	.000	.000
lch	.354	.282	.314	.359	.154	.215	.000	.000	.000	.000	.000	.000
wup	.345	.275	.306	.306	.131	.183	.000	.000	.000	.000	.000	.000
res	.355	.271	.308	.133	.031	.051	.000	.000	.000	.000	.000	.000
lin	.431	.268	.330	.134	.023	.039	.000	.000	.000	.000	.000	.000
jcn	.513	.353	.418	.365	.154	.216	.000	.000	.000	.000	.000	.000
lesk	.517	.515	.516	.297	.293	.295	.582	.568	.574	.491	.480	.485
vector	.511	.510	.511	.305	.304	.304	.592	.592	.592	.412	.409	.411
random	.287	.287	.287	.160	.160	.160	.363	.363	.363	.275	.275	.275
sense1	.634	.634	.634	.422	.422	.422	.661	.661	.661	.696	.696	.696

Table 103: SENSEVAL-2 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.307	.247	.274	.126	.049	.071	.000	.000	.000	.000	.000	.000
lch	.311	.251	.278	.126	.049	.071	.000	.000	.000	.000	.000	.000
wup	.311	.251	.278	.099	.038	.055	.000	.000	.000	.000	.000	.000
res	.250	.194	.218	.048	.010	.017	.000	.000	.000	.000	.000	.000
lin	.244	.161	.194	.044	.007	.012	.000	.000	.000	.000	.000	.000
jcn	.244	.171	.201	.119	.045	.066	.000	.000	.000	.000	.000	.000
lesk	.366	.365	.365	.155	.154	.155	.330	.325	.327	.327	.321	.324
vector	.385	.385	.385	.204	.203	.203	.351	.351	.351	.283	.283	.283
random	.221	.221	.221	.157	.157	.157	.237	.237	.237	.283	.283	.283
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 104: SENSEVAL-2 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.524	.445	.481	.392	.175	.242	1.00	.194	.325	1.00	.372	.542
lch	.527	.447	.484	.392	.175	.242	1.00	.194	.325	1.00	.372	.542
wup	.521	.442	.478	.339	.151	.209	1.00	.194	.325	1.00	.372	.542
res	.539	.443	.486	.213	.053	.085	1.00	.194	.325	1.00	.372	.542
lin	.617	.438	.512	.253	.047	.079	1.00	.194	.325	1.00	.372	.542
jcn	.658	.500	.569	.404	.179	.248	1.00	.194	.325	1.00	.372	.542
lesk	.629	.627	.628	.324	.320	.322	.665	.652	.659	.684	.671	.678
vector	.622	.622	.622	.325	.324	.325	.671	.671	.671	.634	.632	.633
random	.434	.434	.434	.206	.206	.206	.460	.460	.460	.574	.574	.574
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 105: SENSEVAL-2 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.545	.545	.545	.387	.387	.387	.727	.727	.727	.809	.809	.809
lch	.548	.548	.548	.387	.387	.387	.727	.727	.727	.809	.809	.809
wup	.542	.542	.542	.363	.363	.363	.727	.727	.727	.809	.809	.809
res	.557	.557	.557	.363	.363	.363	.727	.727	.727	.809	.809	.809
lin	.630	.630	.630	.385	.385	.385	.727	.727	.727	.809	.809	.809
jcn	.657	.657	.657	.391	.391	.391	.727	.727	.727	.809	.809	.809
lesk	.629	.629	.629	.326	.326	.326	.667	.667	.667	.686	.686	.686
vector	.623	.623	.623	.326	.326	.326	.671	.671	.671	.635	.635	.635
random	.430	.430	.430	.181	.181	.181	.499	.499	.499	.596	.596	.596
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 106: SENSEVAL-2 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.347	.298	.321	.341	.177	.233	.000	.000	.000	.000	.000	.000
lch	.352	.302	.325	.341	.177	.233	.000	.000	.000	.000	.000	.000
wup	.334	.287	.309	.277	.143	.189	.000	.000	.000	.000	.000	.000
res	.359	.297	.325	.146	.044	.067	.000	.000	.000	.000	.000	.000
lin	.448	.314	.370	.143	.033	.054	.000	.000	.000	.000	.000	.000
jcn	.525	.398	.453	.350	.179	.237	.000	.000	.000	.000	.000	.000
lesk	.514	.512	.513	.305	.301	.303	.602	.592	.597	.509	.497	.503
vector	.512	.512	.512	.292	.291	.292	.595	.595	.595	.412	.409	.411
random	.228	.228	.228	.162	.162	.162	.333	.333	.333	.327	.327	.327
sense1	.634	.634	.634	.422	.422	.422	.661	.661	.661	.696	.696	.696

Table 107: SENSEVAL-2 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.304	.264	.283	.116	.056	.075	.000	.000	.000	.000	.000	.000
lch	.312	.271	.290	.116	.056	.075	.000	.000	.000	.000	.000	.000
wup	.308	.268	.286	.087	.042	.057	.000	.000	.000	.000	.000	.000
res	.249	.211	.228	.071	.021	.032	.000	.000	.000	.000	.000	.000
lin	.258	.191	.219	.059	.014	.023	.000	.000	.000	.000	.000	.000
jcn	.253	.197	.222	.118	.056	.076	.000	.000	.000	.000	.000	.000
lesk	.362	.361	.362	.170	.168	.169	.372	.368	.370	.346	.340	.343
vector	.385	.385	.385	.182	.182	.182	.351	.351	.351	.283	.283	.283
random	.237	.237	.237	.157	.157	.157	.228	.228	.228	.189	.189	.189
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 108: SENSEVAL-2 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.515	.463	.488	.369	.196	.256	1.00	.194	.325	1.00	.372	.542
lch	.516	.464	.489	.369	.196	.256	1.00	.194	.325	1.00	.372	.542
wup	.504	.452	.477	.306	.163	.213	1.00	.194	.325	1.00	.372	.542
res	.531	.464	.495	.208	.065	.099	1.00	.194	.325	1.00	.372	.542
lin	.611	.470	.532	.230	.057	.091	1.00	.194	.325	1.00	.372	.542
jcn	.658	.537	.592	.379	.200	.262	1.00	.194	.325	1.00	.372	.542
lesk	.626	.624	.625	.332	.328	.330	.681	.671	.676	.695	.682	.689
vector	.623	.623	.623	.314	.312	.313	.674	.674	.674	.634	.632	.633
random	.427	.427	.427	.216	.216	.216	.460	.460	.460	.574	.574	.574
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 109: SENSEVAL-2 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.531	.531	.531	.377	.377	.377	.727	.727	.727	.809	.809	.809
lch	.534	.534	.534	.377	.377	.377	.727	.727	.727	.809	.809	.809
wup	.521	.521	.521	.344	.344	.344	.727	.727	.727	.809	.809	.809
res	.546	.546	.546	.356	.356	.356	.727	.727	.727	.809	.809	.809
lin	.627	.627	.627	.381	.381	.381	.727	.727	.727	.809	.809	.809
jcn	.659	.659	.659	.383	.383	.383	.727	.727	.727	.809	.809	.809
lesk	.626	.626	.626	.334	.334	.334	.683	.683	.683	.697	.697	.697
vector	.623	.623	.623	.314	.314	.314	.674	.674	.674	.635	.635	.635
random	.437	.437	.437	.220	.220	.220	.477	.477	.477	.592	.592	.592
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 110: SENSEVAL-2 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.377	.339	.357	.331	.200	.249	.000	.000	.000	.000	.000	.000
lch	.377	.339	.357	.331	.200	.249	.000	.000	.000	.000	.000	.000
wup	.358	.322	.339	.262	.158	.197	.000	.000	.000	.000	.000	.000
res	.393	.342	.365	.134	.048	.070	.000	.000	.000	.000	.000	.000
lin	.463	.340	.392	.131	.037	.058	.000	.000	.000	.000	.000	.000
jcn	.532	.424	.472	.343	.202	.254	.000	.000	.000	.000	.000	.000
lesk	.520	.519	.520	.305	.301	.303	.573	.568	.570	.515	.503	.509
vector	.521	.520	.521	.309	.308	.308	.577	.577	.577	.412	.409	.411
random	.268	.268	.268	.173	.173	.173	.357	.357	.357	.269	.269	.269
sense1	.634	.634	.634	.422	.422	.422	.661	.661	.661	.696	.696	.696

Table 111: SENSEVAL-2 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.351	.314	.332	.099	.056	.071	.000	.000	.000	.000	.000	.000
lch	.351	.314	.332	.099	.056	.071	.000	.000	.000	.000	.000	.000
wup	.347	.311	.328	.074	.042	.054	.000	.000	.000	.000	.000	.000
res	.284	.247	.264	.061	.021	.031	.000	.000	.000	.000	.000	.000
lin	.284	.221	.249	.037	.010	.016	.000	.000	.000	.000	.000	.000
jcn	.275	.224	.247	.102	.056	.072	.000	.000	.000	.000	.000	.000
lesk	.359	.358	.358	.176	.175	.175	.316	.316	.316	.283	.283	.283
vector	.395	.395	.395	.186	.185	.186	.298	.298	.298	.302	.302	.302
random	.221	.221	.221	.161	.161	.161	.211	.211	.211	.189	.189	.189
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 112: SENSEVAL-2 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.531	.491	.510	.365	.224	.278	1.00	.194	.325	1.00	.372	.542
lch	.530	.490	.509	.365	.224	.278	1.00	.194	.325	1.00	.372	.542
wup	.514	.476	.494	.298	.183	.227	1.00	.194	.325	1.00	.372	.542
res	.548	.495	.520	.186	.069	.100	1.00	.194	.325	1.00	.372	.542
lin	.615	.490	.546	.204	.061	.094	1.00	.194	.325	1.00	.372	.542
jcn	.664	.560	.607	.376	.226	.282	1.00	.194	.325	1.00	.372	.542
lesk	.631	.630	.631	.331	.328	.330	.657	.652	.655	.692	.682	.687
vector	.630	.629	.629	.329	.328	.329	.659	.659	.659	.634	.632	.633
random	.434	.434	.434	.208	.208	.208	.475	.475	.475	.563	.563	.563
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 113: SENSEVAL-2 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.538	.538	.538	.361	.361	.361	.727	.727	.727	.809	.809	.809
lch	.538	.538	.538	.361	.361	.361	.727	.727	.727	.809	.809	.809
wup	.524	.524	.524	.320	.320	.320	.727	.727	.727	.809	.809	.809
res	.556	.556	.556	.330	.330	.330	.727	.727	.727	.809	.809	.809
lin	.632	.632	.632	.361	.361	.361	.727	.727	.727	.809	.809	.809
jcn	.659	.659	.659	.369	.369	.369	.727	.727	.727	.809	.809	.809
lesk	.631	.631	.631	.334	.334	.334	.659	.659	.659	.697	.697	.697
vector	.630	.630	.630	.330	.330	.330	.659	.659	.659	.635	.635	.635
random	.432	.432	.432	.218	.218	.218	.492	.492	.492	.567	.567	.567
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 114: SENSEVAL-2 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.366	.351	.358	.291	.249	.268	.000	.000	.000	.000	.000	.000
lch	.363	.349	.356	.291	.249	.268	.000	.000	.000	.000	.000	.000
wup	.356	.342	.348	.237	.204	.219	.000	.000	.000	.000	.000	.000
res	.359	.338	.348	.143	.087	.109	.000	.000	.000	.000	.000	.000
lin	.455	.383	.416	.161	.085	.111	.000	.000	.000	.000	.000	.000
jcn	.549	.485	.515	.315	.268	.290	.000	.000	.000	.000	.000	.000
lesk	.537	.536	.537	.282	.281	.282	.584	.583	.583	.458	.450	.454
vector	.547	.546	.547	.342	.341	.342	.595	.595	.595	.406	.404	.405
random	.239	.239	.239	.175	.175	.175	.336	.336	.336	.310	.310	.310
sense1	.634	.634	.634	.422	.422	.422	.661	.661	.661	.696	.696	.696

Table 115: SENSEVAL-2 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.340	.324	.332	.092	.077	.084	.000	.000	.000	.000	.000	.000
lch	.344	.328	.336	.092	.077	.084	.000	.000	.000	.000	.000	.000
wup	.323	.308	.315	.079	.066	.072	.000	.000	.000	.000	.000	.000
res	.245	.231	.238	.084	.052	.065	.000	.000	.000	.000	.000	.000
lin	.273	.237	.254	.067	.038	.049	.000	.000	.000	.000	.000	.000
jcn	.282	.251	.265	.092	.077	.084	.000	.000	.000	.000	.000	.000
lesk	.388	.388	.388	.158	.157	.158	.342	.342	.342	.245	.245	.245
vector	.398	.398	.398	.204	.203	.203	.307	.307	.307	.321	.321	.321
random	.211	.211	.211	.136	.136	.136	.263	.263	.263	.302	.302	.302
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 116: SENSEVAL-2 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.516	.501	.508	.311	.267	.287	1.00	.194	.325	1.00	.372	.542
lch	.515	.500	.507	.313	.269	.289	1.00	.194	.325	1.00	.372	.542
wup	.508	.494	.501	.258	.222	.239	1.00	.194	.325	1.00	.372	.542
res	.511	.489	.500	.177	.108	.134	1.00	.194	.325	1.00	.372	.542
lin	.595	.523	.557	.202	.108	.141	1.00	.194	.325	1.00	.372	.542
jcn	.663	.604	.632	.338	.289	.311	1.00	.194	.325	1.00	.372	.542
lesk	.644	.643	.644	.312	.310	.311	.666	.664	.665	.661	.653	.657
vector	.649	.648	.648	.359	.358	.358	.674	.674	.674	.634	.632	.633
random	.431	.431	.431	.208	.208	.208	.499	.499	.499	.549	.549	.549
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 117: SENSEVAL-2 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

A.3.3 SENSEVAL-3 Tables

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.518	.518	.518	.316	.316	.316	.727	.727	.727	.809	.809	.809
lch	.516	.516	.516	.318	.318	.318	.727	.727	.727	.809	.809	.809
wup	.509	.509	.509	.271	.271	.271	.727	.727	.727	.809	.809	.809
res	.517	.517	.517	.287	.287	.287	.727	.727	.727	.809	.809	.809
lin	.608	.608	.608	.334	.334	.334	.727	.727	.727	.809	.809	.809
jcn	.661	.661	.661	.338	.338	.338	.727	.727	.727	.809	.809	.809
lesk	.644	.644	.644	.314	.314	.314	.667	.667	.667	.664	.664	.664
vector	.649	.649	.649	.360	.360	.360	.674	.674	.674	.635	.635	.635
random	.419	.419	.419	.250	.250	.250	.439	.439	.439	.585	.585	.585
sense1	.717	.717	.717	.438	.438	.438	.727	.727	.727	.809	.809	.809

Table 118: SENSEVAL-2 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

measure	P	R	F	Att(%)	Time
path	.394	.155	.222	39.27	00:03:10
lch	.394	.155	.222	39.27	00:05:25
wup	.365	.143	.206	39.27	00:04:41
res	.301	.097	.147	32.22	00:03:18
lin	.388	.114	.177	29.50	00:03:53
jcn	.481	.184	.266	38.22	00:04:29
lesk	.421	.401	.411	95.30	00:06:38
vector	.405	.401	.403	98.89	00:08:26
random	.205	.205	.205	100.00	00:00:09
sense1	.602	.602	.602	100.00	00:00:09

Table 119: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,617. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.394	.155	.222	39.27	00:03:10
lch	.394	.155	.222	39.27	00:05:25
wup	.365	.143	.206	39.27	00:04:41
res	.301	.097	.147	32.22	00:03:18
lin	.388	.114	.177	29.50	00:03:53
jcn	.481	.184	.266	38.22	00:04:29
lesk	.421	.401	.411	95.30	00:06:38
vector	.405	.401	.403	98.89	00:08:26
random	.205	.205	.205	100.00	00:00:09
sense1	.602	.602	.602	100.00	00:00:09

Table 120: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,617. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.165	.060	.088	36.45	00:01:08
lch	.165	.060	.088	36.45	00:01:46
wup	.190	.069	.102	36.45	00:01:37
res	.129	.038	.058	29.22	00:01:08
lin	.197	.053	.083	26.81	00:01:25
jcn	.174	.062	.091	35.54	00:01:26
lesk	.220	.212	.216	96.54	00:04:24
vector	.210	.208	.209	99.10	00:07:06
random	.184	.184	.184	100.00	00:00:08
sense1	.000	.000	.000	100.00	00:00:09

Table 121: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 664. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.571	.275	.371	48.12	00:03:47
lch	.572	.275	.372	48.12	00:04:54
wup	.558	.268	.362	48.12	00:04:38
res	.547	.230	.324	41.97	00:03:45
lin	.619	.246	.352	39.70	00:04:12
jcn	.642	.305	.413	47.44	00:04:32
lesk	.510	.489	.499	95.97	00:06:33
vector	.495	.490	.492	99.02	00:08:17
random	.331	.331	.331	100.00	00:00:10
sense1	.657	.657	.657	100.00	00:00:10

Table 122: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.577	.577	.577	100.00	00:01:20
lch	.577	.577	.577	100.00	00:01:58
wup	.568	.568	.568	100.00	00:01:48
res	.568	.568	.568	100.00	00:01:19
lin	.598	.598	.598	100.00	00:01:37
jcn	.609	.609	.609	100.00	00:01:39
lesk	.518	.518	.518	100.00	00:04:31
vector	.497	.497	.497	100.00	00:07:11
random	.321	.321	.321	100.00	00:00:08
sense1	.657	.657	.657	100.00	00:00:09

Table 123: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.377	.196	.258	51.95	00:01:57
lch	.380	.197	.260	51.95	00:02:50
wup	.340	.177	.233	51.95	00:03:11
res	.295	.127	.177	42.98	00:02:00
lin	.362	.143	.205	39.52	00:02:12
jcn	.445	.224	.299	5.40	00:02:13
lesk	.431	.417	.424	96.85	00:07:06
vector	.412	.407	.409	98.89	00:08:23
random	.206	.206	.206	100.00	00:00:08
sense1	.602	.602	.602	100.00	00:00:08

Table 124: SENSEVAL-3 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,617. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.225	.117	.154	52.26	00:01:59
lch	.225	.117	.154	52.26	00:02:40
wup	.225	.117	.154	52.26	00:02:58
res	.168	.072	.101	43.07	00:02:00
lin	.208	.081	.117	39.16	00:02:10
jcn	.193	.098	.130	5.60	00:02:14
lesk	.221	.215	.218	97.59	00:06:59
vector	.225	.223	.224	99.10	00:08:06
random	.173	.173	.173	100.00	00:00:08
sense1	.000	.000	.000	100.00	00:00:09

Table 125: SENSEVAL-3 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 664. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.532	.314	.395	59.11	00:04:30
lch	.535	.316	.398	59.11	00:05:14
wup	.507	.299	.376	59.11	00:05:24
res	.503	.257	.340	51.16	00:04:36
lin	.561	.270	.364	48.17	00:04:32
jcn	.589	.342	.433	58.03	00:04:54
lesk	.517	.503	.510	97.32	00:08:17
vector	.500	.495	.498	99.02	00:09:11
random	.334	.334	.334	100.00	00:00:08
sense1	.657	.657	.657	100.00	00:00:09

Table 126: SENSEVAL-3 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.559	.559	.559	100.00	00:01:55
lch	.560	.560	.560	100.00	00:02:42
wup	.543	.543	.543	100.00	00:02:59
res	.549	.549	.549	100.00	00:02:00
lin	.579	.579	.579	100.00	00:02:13
jcn	.593	.593	.593	100.00	00:02:18
lesk	.521	.521	.521	100.00	00:07:03
vector	.502	.502	.502	100.00	00:08:05
random	.334	.334	.334	100.00	00:00:08
sense1	.657	.657	.657	100.00	00:00:09

Table 127: SENSEVAL-3 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.380	.234	.290	61.66	00:02:09
lch	.381	.235	.291	61.66	00:03:30
wup	.345	.213	.263	61.66	00:03:12
res	.299	.154	.203	51.45	00:02:10
lin	.368	.174	.237	47.43	00:02:49
jcn	.450	.270	.338	6.11	00:02:44
lesk	.432	.424	.428	98.14	00:07:19
vector	.423	.419	.421	98.89	00:09:41
random	.214	.214	.214	100.00	00:00:08
sense1	.602	.602	.602	100.00	00:00:09

Table 128: SENSEVAL-3 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,617. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.235	.143	.178	6.84	00:02:06
lch	.238	.145	.180	6.84	00:03:27
wup	.233	.142	.176	6.84	00:03:09
res	.183	.093	.124	5.90	00:02:08
lin	.241	.113	.154	46.84	00:02:45
jcn	.203	.120	.151	59.49	00:02:44
lesk	.205	.202	.203	98.64	00:07:14
vector	.223	.221	.222	99.10	00:09:20
random	.191	.191	.191	100.00	00:00:09
sense1	.000	.000	.000	100.00	00:00:09

Table 129: SENSEVAL-3 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 664. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.514	.348	.415	67.73	00:04:50
lch	.515	.349	.416	67.73	00:06:53
wup	.487	.330	.393	67.73	00:06:15
res	.480	.281	.354	58.49	00:04:59
lin	.539	.295	.381	54.83	00:05:49
jcn	.574	.380	.458	66.29	00:06:08
lesk	.516	.507	.512	98.40	00:08:49
vector	.510	.505	.507	99.02	00:10:51
random	.337	.337	.337	100.00	00:00:08
sense1	.657	.657	.657	100.00	00:00:09

Table 130: SENSEVAL-3 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.539	.539	.539	100.00	00:02:09
lch	.539	.539	.539	100.00	00:03:31
wup	.521	.521	.521	100.00	00:03:12
res	.528	.528	.528	100.00	00:02:13
lin	.566	.566	.566	100.00	00:02:50
jcn	.580	.580	.580	100.00	00:02:47
lesk	.519	.519	.519	100.00	00:07:24
vector	.512	.512	.512	100.00	00:09:47
random	.318	.318	.318	100.00	00:00:08
sense1	.657	.657	.657	100.00	00:00:09

Table 131: SENSEVAL-3 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.373	.247	.297	66.17	00:02:44
lch	.374	.247	.298	66.17	00:04:10
wup	.337	.223	.269	66.17	00:04:09
res	.280	.156	.201	55.91	00:02:48
lin	.355	.184	.243	51.95	00:03:21
jcn	.453	.294	.356	64.81	00:03:23
lesk	.425	.417	.421	98.33	00:09:30
vector	.421	.416	.419	98.89	00:11:52
random	.205	.205	.205	100.00	00:00:08
sense1	.602	.602	.602	100.00	00:00:08

Table 132: SENSEVAL-3 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,617. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.223	.148	.178	66.27	00:07:15
lch	.223	.148	.178	66.27	00:09:03
wup	.214	.142	.170	66.27	00:08:35
res	.173	.098	.125	56.48	00:07:23
lin	.235	.125	.163	53.16	00:07:28
jcn	.206	.136	.164	65.66	00:08:07
lesk	.209	.206	.208	98.64	00:11:33
vector	.219	.217	.218	99.10	00:12:32
random	.176	.176	.176	100.00	00:00:09
sense1	.000	.000	.000	100.00	00:00:09

Table 133: SENSEVAL-3 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 664. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.501	.359	.419	71.66	00:05:58
lch	.503	.360	.420	71.66	00:07:18
wup	.473	.339	.395	71.66	00:07:08
res	.456	.285	.351	62.47	00:06:05
lin	.521	.307	.386	58.91	00:06:23
jcn	.570	.402	.471	7.42	00:06:32
lesk	.510	.502	.506	98.55	00:10:47
vector	.507	.502	.504	99.02	00:12:19
random	.337	.337	.337	100.00	00:00:09
sense1	.657	.657	.657	100.00	00:00:09

Table 134: SENSEVAL-3 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 1937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.528	.528	.528	100.00	00:02:45
lch	.528	.528	.528	100.00	00:04:08
wup	.509	.509	.509	100.00	00:04:09
res	.511	.511	.511	100.00	00:02:47
lin	.557	.557	.557	100.00	00:03:25
jcn	.580	.580	.580	100.00	00:03:23
lesk	.512	.512	.512	100.00	00:09:26
vector	.509	.509	.509	100.00	00:11:05
random	.338	.338	.338	100.00	00:00:08
sense1	.657	.657	.657	100.00	00:00:09

Table 135: SENSEVAL-3 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.360	.252	.297	7.07	00:02:56
lch	.365	.255	.300	7.07	00:04:59
wup	.319	.223	.263	7.07	00:04:18
res	.267	.161	.201	6.11	00:03:00
lin	.341	.190	.244	55.84	00:03:51
jcn	.440	.304	.359	68.95	00:03:52
lesk	.422	.415	.418	98.33	00:10:00
vector	.425	.421	.423	98.89	00:13:39
random	.207	.207	.207	100.00	00:00:10
sense1	.602	.602	.602	100.00	00:00:10

Table 136: SENSEVAL-3 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,617. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.208	.146	.172	7.18	00:07:36
lch	.212	.149	.175	7.18	00:09:15
wup	.208	.146	.172	7.18	00:08:46
res	.161	.098	.122	6.69	00:07:37
lin	.217	.123	.157	56.93	00:08:02
jcn	.193	.134	.158	69.43	00:08:32
lesk	.206	.203	.205	98.64	00:11:47
vector	.220	.218	.219	99.10	00:16:43
random	.173	.173	.173	100.00	00:00:09
sense1	.000	.000	.000	100.00	00:00:10

Table 137: SENSEVAL-3 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 664. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.485	.364	.416	75.06	00:02:50
lch	.488	.367	.419	75.06	00:04:43
wup	.451	.339	.387	75.06	00:04:12
res	.437	.288	.347	65.88	00:02:55
lin	.502	.312	.385	62.05	00:03:42
jcn	.555	.409	.471	73.83	00:03:41
lesk	.508	.500	.504	98.55	00:09:59
vector	.510	.505	.508	99.02	00:13:46
random	.323	.323	.323	100.00	00:00:08
sense1	.657	.657	.657	100.00	00:00:09

Table 138: SENSEVAL-3 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.513	.513	.513	100.00	00:02:57
lch	.516	.516	.516	100.00	00:04:50
wup	.490	.490	.490	100.00	00:04:16
res	.494	.494	.494	100.00	00:02:56
lin	.543	.543	.543	100.00	00:03:45
jcn	.566	.566	.566	100.00	00:03:46
lesk	.510	.510	.510	100.00	00:10:05
vector	.512	.512	.512	100.00	00:13:58
random	.329	.329	.329	100.00	00:00:09
sense1	.657	.657	.657	100.00	00:00:09

Table 139: SENSEVAL-3 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.352	.267	.304	75.82	00:04:34
lch	.355	.269	.306	75.82	00:07:54
wup	.304	.231	.262	75.82	00:07:03
res	.275	.186	.222	67.59	00:04:36
lin	.344	.220	.268	64.07	00:06:11
jcn	.430	.323	.369	75.08	00:06:02
lesk	.421	.414	.417	98.52	00:19:16
vector	.440	.435	.438	98.89	00:55:48
random	.208	.208	.208	100.00	00:00:09
sense1	.602	.602	.602	100.00	00:00:09

Table 140: SENSEVAL-3 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector, no forcepos. # tokens = 1,617. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.219	.167	.190	76.20	00:04:24
lch	.221	.169	.191	76.20	00:07:44
wup	.200	.152	.173	76.20	00:06:44
res	.187	.125	.150	66.72	00:04:32
lin	.225	.142	.174	62.80	00:06:00
jcn	.202	.152	.174	75.30	00:05:58
lesk	.214	.211	.212	98.64	00:18:53
vector	.214	.212	.213	99.10	00:54:01
random	.184	.184	.184	100.00	00:00:08
sense1	.000	.000	.000	100.00	00:00:08

Table 141: SENSEVAL-3 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector, no forcepos. # tokens = 664. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.475	.380	.422	8.02	00:07:13
lch	.478	.383	.425	8.02	00:10:15
wup	.438	.351	.389	8.02	00:09:17
res	.428	.309	.359	72.28	00:07:20
lin	.488	.337	.398	68.97	00:08:38
jcn	.540	.428	.478	79.19	00:08:44
lesk	.507	.500	.504	98.71	00:19:34
vector	.525	.519	.522	99.02	00:55:29
random	.343	.343	.343	100.00	00:00:08
sense1	.657	.657	.657	100.00	00:00:09

Table 142: SENSEVAL-3 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

measure	P	R	F	Att(%)	Time
path	.498	.498	.498	100.00	00:04:31
lch	.500	.500	.500	100.00	00:07:52
wup	.469	.469	.469	100.00	00:06:49
res	.472	.472	.472	100.00	00:04:39
lin	.519	.519	.519	100.00	00:06:06
jcn	.551	.551	.551	100.00	00:05:58
lesk	.509	.509	.509	100.00	00:19:22
vector	.526	.526	.526	100.00	00:53:33
random	.320	.320	.320	100.00	00:00:09
sense1	.657	.657	.657	100.00	00:00:09

Table 143: SENSEVAL-3 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector, no forcepos. # tokens = 1,937. 'Att' is 'Attempted'.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.367	.191	.251	.432	.175	.249	.000	.000	.000	.000	.000	.000
lch	.367	.191	.251	.432	.175	.249	.000	.000	.000	.000	.000	.000
wup	.358	.187	.246	.375	.152	.216	.000	.000	.000	.000	.000	.000
res	.376	.185	.248	.147	.038	.061	.000	.000	.000	.000	.000	.000
lin	.489	.219	.303	.184	.044	.072	.000	.000	.000	.000	.000	.000
jcn	.548	.272	.364	.390	.158	.225	.000	.000	.000	.000	.000	.000
lesk	.455	.442	.449	.366	.345	.355	.464	.433	.448	.000	.000	.000
vector	.435	.434	.434	.354	.346	.350	.452	.448	.450	.000	.000	.000
random	.235	.235	.235	.147	.147	.147	.270	.270	.270	.000	.000	.000
sense1	.660	.660	.660	.528	.528	.528	.627	.627	.627	.000	.000	.000

Table 144: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.195	.095	.127	.137	.052	.075	.000	.000	.000	.000	.000	.000
lch	.195	.095	.127	.137	.052	.075	.000	.000	.000	.000	.000	.000
wup	.229	.111	.150	.153	.058	.084	.000	.000	.000	.000	.000	.000
res	.159	.074	.101	.086	.021	.034	.000	.000	.000	.000	.000	.000
lin	.260	.111	.156	.108	.024	.040	.000	.000	.000	.000	.000	.000
jcn	.241	.111	.152	.113	.043	.062	.000	.000	.000	.000	.000	.000
lesk	.296	.292	.294	.158	.150	.154	.233	.223	.228	.000	.000	.000
vector	.272	.272	.272	.143	.141	.142	.277	.277	.277	.000	.000	.000
random	.189	.189	.189	.144	.144	.144	.309	.309	.309	.000	.000	.000
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 145: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.551	.334	.416	.492	.216	.300	1.00	.217	.357	1.00	1.00	1.00
lch	.553	.335	.417	.492	.216	.300	1.00	.217	.357	1.00	1.00	1.00
wup	.555	.336	.419	.448	.196	.273	1.00	.217	.357	1.00	1.00	1.00
res	.573	.337	.425	.308	.090	.140	1.00	.217	.357	1.00	1.00	1.00
lin	.665	.368	.473	.348	.096	.150	1.00	.217	.357	1.00	1.00	1.00
jcn	.690	.407	.512	.470	.206	.286	1.00	.217	.357	1.00	1.00	1.00
lesk	.562	.550	.556	.400	.377	.388	.587	.556	.571	1.00	1.00	1.00
vector	.545	.544	.544	.388	.380	.384	.572	.568	.570	1.00	1.00	1.00
random	.389	.389	.389	.196	.196	.196	.447	.447	.447	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 146: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.592	.592	.592	.494	.494	.494	.708	.708	.708	1.00	1.00	1.00
lch	.592	.592	.592	.494	.494	.494	.708	.708	.708	1.00	1.00	1.00
wup	.588	.588	.588	.474	.474	.474	.708	.708	.708	1.00	1.00	1.00
res	.604	.604	.604	.453	.453	.453	.708	.708	.708	1.00	1.00	1.00
lin	.658	.658	.658	.467	.467	.467	.708	.708	.708	1.00	1.00	1.00
jcn	.671	.671	.671	.483	.483	.483	.708	.708	.708	1.00	1.00	1.00
lesk	.569	.569	.569	.412	.412	.412	.596	.596	.596	1.00	1.00	1.00
vector	.545	.545	.545	.394	.394	.394	.575	.575	.575	1.00	1.00	1.00
random	.372	.372	.372	.206	.206	.206	.413	.413	.413	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 147: SENSEVAL-3 results with wntagged format, window=3, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.343	.235	.279	.425	.230	.298	.000	.000	.000	.000	.000	.000
lch	.347	.237	.282	.425	.230	.298	.000	.000	.000	.000	.000	.000
wup	.333	.228	.270	.351	.190	.247	.000	.000	.000	.000	.000	.000
res	.365	.237	.288	.155	.055	.081	.000	.000	.000	.000	.000	.000
lin	.448	.268	.336	.188	.061	.092	.000	.000	.000	.000	.000	.000
jcn	.503	.327	.397	.369	.199	.259	.000	.000	.000	.000	.000	.000
lesk	.466	.459	.463	.370	.354	.362	.485	.464	.475	.000	.000	.000
vector	.447	.447	.447	.343	.335	.339	.484	.480	.482	.000	.000	.000
random	.221	.221	.221	.161	.161	.161	.282	.282	.282	.000	.000	.000
sense1	.660	.660	.660	.528	.528	.528	.627	.627	.627	.000	.000	.000

Table 148: SENSEVAL-3 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.291	.206	.241	.160	.086	.112	.000	.000	.000	.000	.000	.000
lch	.291	.206	.241	.160	.086	.112	.000	.000	.000	.000	.000	.000
wup	.297	.210	.246	.154	.083	.108	.000	.000	.000	.000	.000	.000
res	.217	.148	.176	.100	.037	.054	.000	.000	.000	.000	.000	.000
lin	.285	.177	.218	.101	.034	.050	.000	.000	.000	.000	.000	.000
jcn	.278	.185	.222	.115	.061	.080	.000	.000	.000	.000	.000	.000
lesk	.285	.284	.285	.159	.153	.156	.264	.255	.259	.000	.000	.000
vector	.292	.292	.292	.150	.147	.148	.309	.309	.309	.000	.000	.000
random	.173	.173	.173	.122	.122	.122	.351	.351	.351	.000	.000	.000
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 149: SENSEVAL-3 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.513	.380	.437	.468	.266	.339	1.00	.217	.357	1.00	1.00	1.00
lch	.519	.385	.442	.468	.266	.339	1.00	.217	.357	1.00	1.00	1.00
wup	.507	.376	.431	.407	.231	.295	1.00	.217	.357	1.00	1.00	1.00
res	.536	.383	.447	.279	.107	.155	1.00	.217	.357	1.00	1.00	1.00
lin	.606	.408	.488	.314	.111	.164	1.00	.217	.357	1.00	1.00	1.00
jcn	.634	.455	.530	.436	.248	.316	1.00	.217	.357	1.00	1.00	1.00
lesk	.572	.564	.568	.401	.384	.392	.601	.581	.591	1.00	1.00	1.00
vector	.555	.554	.555	.378	.370	.374	.597	.593	.595	1.00	1.00	1.00
random	.382	.382	.382	.192	.192	.192	.494	.494	.494	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 150: SENSEVAL-3 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.555	.555	.555	.488	.488	.488	.708	.708	.708	1.00	1.00	1.00
lch	.558	.558	.558	.488	.488	.488	.708	.708	.708	1.00	1.00	1.00
wup	.550	.550	.550	.453	.453	.453	.708	.708	.708	1.00	1.00	1.00
res	.579	.579	.579	.434	.434	.434	.708	.708	.708	1.00	1.00	1.00
lin	.629	.629	.629	.453	.453	.453	.708	.708	.708	1.00	1.00	1.00
jcn	.647	.647	.647	.467	.467	.467	.708	.708	.708	1.00	1.00	1.00
lesk	.576	.576	.576	.409	.409	.409	.606	.606	.606	1.00	1.00	1.00
vector	.555	.555	.555	.384	.384	.384	.599	.599	.599	1.00	1.00	1.00
random	.388	.388	.388	.193	.193	.193	.475	.475	.475	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 151: SENSEVAL-3 results with wntagged format, window=4, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.347	.275	.307	.424	.280	.337	.000	.000	.000	.000	.000	.000
lch	.347	.275	.307	.426	.282	.339	.000	.000	.000	.000	.000	.000
wup	.326	.258	.288	.370	.245	.295	.000	.000	.000	.000	.000	.000
res	.375	.285	.324	.158	.070	.097	.000	.000	.000	.000	.000	.000
lin	.458	.322	.378	.199	.081	.115	.000	.000	.000	.000	.000	.000
jcn	.507	.386	.439	.377	.248	.299	.000	.000	.000	.000	.000	.000
lesk	.464	.461	.462	.369	.358	.363	.502	.488	.495	.000	.000	.000
vector	.447	.447	.447	.373	.364	.369	.484	.480	.482	.000	.000	.000
random	.229	.229	.229	.153	.153	.153	.329	.329	.329	.000	.000	.000
sense1	.660	.660	.660	.528	.528	.528	.627	.627	.627	.000	.000	.000

Table 152: SENSEVAL-3 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.311	.247	.275	.166	.107	.130	.000	.000	.000	.000	.000	.000
lch	.311	.247	.275	.171	.110	.134	.000	.000	.000	.000	.000	.000
wup	.295	.235	.261	.175	.113	.138	.000	.000	.000	.000	.000	.000
res	.245	.189	.213	.107	.049	.067	.000	.000	.000	.000	.000	.000
lin	.331	.239	.278	.125	.052	.073	.000	.000	.000	.000	.000	.000
jcn	.293	.222	.253	.123	.080	.097	.000	.000	.000	.000	.000	.000
lesk	.255	.255	.255	.144	.141	.142	.283	.277	.280	.000	.000	.000
vector	.272	.272	.272	.171	.168	.170	.277	.277	.277	.000	.000	.000
random	.230	.230	.230	.147	.147	.147	.245	.245	.245	.000	.000	.000
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 153: SENSEVAL-3 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.495	.415	.452	.460	.313	.373	1.00	.217	.357	1.00	1.00	1.00
lch	.497	.416	.453	.462	.314	.374	1.00	.217	.357	1.00	1.00	1.00
wup	.478	.400	.436	.415	.282	.336	1.00	.217	.357	1.00	1.00	1.00
res	.522	.423	.467	.263	.122	.167	1.00	.217	.357	1.00	1.00	1.00
lin	.589	.447	.508	.307	.132	.185	1.00	.217	.357	1.00	1.00	1.00
jcn	.620	.501	.554	.435	.295	.352	1.00	.217	.357	1.00	1.00	1.00
lesk	.569	.566	.567	.397	.387	.392	.613	.599	.606	1.00	1.00	1.00
vector	.555	.554	.555	.405	.396	.401	.597	.593	.595	1.00	1.00	1.00
random	.396	.396	.396	.204	.204	.204	.447	.447	.447	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 154: SENSEVAL-3 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.525	.525	.525	.473	.473	.473	.708	.708	.708	1.00	1.00	1.00
lch	.525	.525	.525	.474	.474	.474	.708	.708	.708	1.00	1.00	1.00
wup	.511	.511	.511	.442	.442	.442	.708	.708	.708	1.00	1.00	1.00
res	.554	.554	.554	.408	.408	.408	.708	.708	.708	1.00	1.00	1.00
lin	.615	.615	.615	.434	.434	.434	.708	.708	.708	1.00	1.00	1.00
jcn	.630	.630	.630	.453	.453	.453	.708	.708	.708	1.00	1.00	1.00
lesk	.571	.571	.571	.403	.403	.403	.615	.615	.615	1.00	1.00	1.00
vector	.555	.555	.555	.410	.410	.410	.599	.599	.599	1.00	1.00	1.00
random	.362	.362	.362	.203	.203	.203	.429	.429	.429	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 155: SENSEVAL-3 results with wntagged format, window=5, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.331	.277	.301	.426	.309	.358	.000	.000	.000	.000	.000	.000
lch	.329	.275	.300	.430	.312	.362	.000	.000	.000	.000	.000	.000
wup	.317	.265	.289	.363	.263	.305	.000	.000	.000	.000	.000	.000
res	.357	.287	.318	.147	.075	.099	.000	.000	.000	.000	.000	.000
lin	.445	.334	.382	.197	.092	.125	.000	.000	.000	.000	.000	.000
jcn	.503	.407	.450	.392	.283	.329	.000	.000	.000	.000	.000	.000
lesk	.467	.465	.466	.348	.338	.343	.500	.488	.494	.000	.000	.000
vector	.451	.451	.451	.370	.361	.366	.464	.460	.462	.000	.000	.000
random	.226	.226	.226	.133	.133	.133	.329	.329	.329	.000	.000	.000
sense1	.660	.660	.660	.528	.528	.528	.627	.627	.627	.000	.000	.000

Table 156: SENSEVAL-3 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.283	.239	.259	.170	.122	.142	.000	.000	.000	.000	.000	.000
lch	.278	.235	.254	.174	.125	.146	.000	.000	.000	.000	.000	.000
wup	.259	.218	.237	.174	.125	.146	.000	.000	.000	.000	.000	.000
res	.225	.185	.203	.114	.061	.080	.000	.000	.000	.000	.000	.000
lin	.323	.255	.285	.130	.064	.086	.000	.000	.000	.000	.000	.000
jcn	.289	.239	.261	.136	.098	.114	.000	.000	.000	.000	.000	.000
lesk	.263	.263	.263	.144	.141	.142	.293	.287	.290	.000	.000	.000
vector	.284	.284	.284	.156	.153	.154	.266	.266	.266	.000	.000	.000
random	.185	.185	.185	.150	.150	.150	.245	.245	.245	.000	.000	.000
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 157: SENSEVAL-3 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.477	.415	.444	.461	.344	.394	1.00	.217	.357	1.00	1.00	1.00
lch	.477	.415	.444	.465	.346	.397	1.00	.217	.357	1.00	1.00	1.00
wup	.462	.403	.430	.409	.305	.349	1.00	.217	.357	1.00	1.00	1.00
res	.503	.424	.460	.248	.132	.172	1.00	.217	.357	1.00	1.00	1.00
lin	.575	.459	.511	.300	.147	.198	1.00	.217	.357	1.00	1.00	1.00
jcn	.612	.518	.561	.446	.331	.380	1.00	.217	.357	1.00	1.00	1.00
lesk	.571	.569	.570	.379	.369	.374	.611	.599	.605	1.00	1.00	1.00
vector	.557	.557	.557	.401	.392	.397	.581	.578	.579	1.00	1.00	1.00
random	.402	.402	.402	.184	.184	.184	.478	.478	.478	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 158: SENSEVAL-3 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.505	.505	.505	.467	.467	.467	.708	.708	.708	1.00	1.00	1.00
lch	.503	.503	.503	.470	.470	.470	.708	.708	.708	1.00	1.00	1.00
wup	.495	.495	.495	.428	.428	.428	.708	.708	.708	1.00	1.00	1.00
res	.535	.535	.535	.385	.385	.385	.708	.708	.708	1.00	1.00	1.00
lin	.605	.605	.605	.423	.423	.423	.708	.708	.708	1.00	1.00	1.00
jcn	.628	.628	.628	.458	.458	.458	.708	.708	.708	1.00	1.00	1.00
lesk	.572	.572	.572	.385	.385	.385	.612	.612	.612	1.00	1.00	1.00
vector	.558	.558	.558	.406	.406	.406	.584	.584	.584	1.00	1.00	1.00
random	.377	.377	.377	.218	.218	.218	.472	.472	.472	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 159: SENSEVAL-3 results with wntagged format, window=6, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.313	.275	.293	.419	.325	.366	.000	.000	.000	.000	.000	.000
lch	.317	.279	.297	.423	.328	.369	.000	.000	.000	.000	.000	.000
wup	.303	.267	.284	.338	.262	.295	.000	.000	.000	.000	.000	.000
res	.352	.296	.322	.132	.075	.096	.000	.000	.000	.000	.000	.000
lin	.439	.348	.388	.178	.092	.121	.000	.000	.000	.000	.000	.000
jcn	.499	.428	.461	.369	.285	.322	.000	.000	.000	.000	.000	.000
lesk	.472	.471	.471	.340	.331	.335	.488	.476	.482	.000	.000	.000
vector	.460	.459	.460	.378	.369	.373	.448	.444	.446	.000	.000	.000
random	.247	.247	.247	.142	.142	.142	.262	.262	.262	.000	.000	.000
sense1	.660	.660	.660	.528	.528	.528	.627	.627	.627	.000	.000	.000

Table 160: SENSEVAL-3 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.263	.235	.248	.161	.122	.139	.000	.000	.000	.000	.000	.000
lch	.263	.235	.248	.169	.128	.146	.000	.000	.000	.000	.000	.000
wup	.263	.235	.248	.161	.122	.139	.000	.000	.000	.000	.000	.000
res	.214	.185	.199	.104	.061	.077	.000	.000	.000	.000	.000	.000
lin	.305	.251	.275	.118	.064	.083	.000	.000	.000	.000	.000	.000
jcn	.278	.243	.259	.120	.092	.104	.000	.000	.000	.000	.000	.000
lesk	.280	.280	.280	.131	.128	.130	.272	.266	.269	.000	.000	.000
vector	.296	.296	.296	.153	.150	.151	.255	.255	.255	.000	.000	.000
random	.165	.165	.165	.153	.153	.153	.266	.266	.266	.000	.000	.000
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 161: SENSEVAL-3 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.458	.415	.435	.449	.356	.397	1.00	.217	.357	1.00	1.00	1.00
lch	.461	.419	.439	.453	.359	.400	1.00	.217	.357	1.00	1.00	1.00
wup	.446	.405	.425	.379	.300	.335	1.00	.217	.357	1.00	1.00	1.00
res	.492	.430	.459	.225	.132	.167	1.00	.217	.357	1.00	1.00	1.00
lin	.567	.471	.514	.275	.147	.192	1.00	.217	.357	1.00	1.00	1.00
jcn	.607	.536	.569	.418	.330	.369	1.00	.217	.357	1.00	1.00	1.00
lesk	.575	.574	.575	.371	.362	.366	.601	.590	.596	1.00	1.00	1.00
vector	.564	.563	.564	.408	.399	.404	.569	.565	.567	1.00	1.00	1.00
random	.387	.387	.387	.174	.174	.174	.457	.457	.457	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 162: SENSEVAL-3 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.482	.482	.482	.456	.456	.456	.708	.708	.708	1.00	1.00	1.00
lch	.485	.485	.485	.459	.459	.459	.708	.708	.708	1.00	1.00	1.00
wup	.475	.475	.475	.402	.402	.402	.708	.708	.708	1.00	1.00	1.00
res	.521	.521	.521	.356	.356	.356	.708	.708	.708	1.00	1.00	1.00
lin	.592	.592	.592	.401	.401	.401	.708	.708	.708	1.00	1.00	1.00
jcn	.618	.618	.618	.433	.433	.433	.708	.708	.708	1.00	1.00	1.00
lesk	.577	.577	.577	.378	.378	.378	.602	.602	.602	1.00	1.00	1.00
vector	.564	.564	.564	.413	.413	.413	.571	.571	.571	1.00	1.00	1.00
random	.378	.378	.378	.199	.199	.199	.463	.463	.463	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 163: SENSEVAL-3 results with wntagged format, window=7, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.321	.296	.308	.389	.338	.362	.000	.000	.000	.000	.000	.000
lch	.318	.294	.305	.398	.346	.370	.000	.000	.000	.000	.000	.000
wup	.302	.279	.291	.306	.266	.285	.000	.000	.000	.000	.000	.000
res	.369	.337	.352	.138	.093	.111	.000	.000	.000	.000	.000	.000
lin	.446	.393	.418	.186	.116	.143	.000	.000	.000	.000	.000	.000
jcn	.492	.448	.469	.359	.311	.333	.000	.000	.000	.000	.000	.000
lesk	.482	.480	.481	.328	.319	.323	.484	.476	.480	.000	.000	.000
vector	.475	.475	.475	.392	.383	.387	.464	.460	.462	.000	.000	.000
random	.226	.226	.226	.147	.147	.147	.317	.317	.317	.000	.000	.000
sense1	.660	.660	.660	.528	.528	.528	.627	.627	.627	.000	.000	.000

Table 164: SENSEVAL-3 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –score poly with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.305	.280	.292	.152	.131	.141	.000	.000	.000	.000	.000	.000
lch	.305	.280	.292	.155	.135	.144	.000	.000	.000	.000	.000	.000
wup	.274	.251	.262	.141	.122	.131	.000	.000	.000	.000	.000	.000
res	.260	.235	.247	.116	.080	.094	.000	.000	.000	.000	.000	.000
lin	.319	.276	.296	.130	.083	.101	.000	.000	.000	.000	.000	.000
jcn	.303	.272	.286	.124	.107	.115	.000	.000	.000	.000	.000	.000
lesk	.280	.280	.280	.134	.131	.133	.315	.309	.312	.000	.000	.000
vector	.288	.288	.288	.156	.153	.154	.223	.223	.223	.000	.000	.000
random	.173	.173	.173	.141	.141	.141	.362	.362	.362	.000	.000	.000
sense1	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 165: SENSEVAL-3 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –s1nc with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.460	.434	.447	.426	.376	.399	1.00	.217	.357	1.00	1.00	1.00
lch	.460	.434	.447	.434	.382	.407	1.00	.217	.357	1.00	1.00	1.00
wup	.447	.422	.434	.353	.312	.331	1.00	.217	.357	1.00	1.00	1.00
res	.499	.465	.481	.215	.147	.175	1.00	.217	.357	1.00	1.00	1.00
lin	.563	.508	.534	.265	.168	.206	1.00	.217	.357	1.00	1.00	1.00
jcn	.598	.554	.575	.407	.357	.380	1.00	.217	.357	1.00	1.00	1.00
lesk	.583	.581	.582	.361	.352	.357	.597	.590	.594	1.00	1.00	1.00
vector	.576	.576	.576	.425	.416	.421	.581	.578	.579	1.00	1.00	1.00
random	.395	.395	.395	.213	.213	.213	.469	.469	.469	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 166: SENSEVAL-3 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –usemono with measure config for lesk and vector and no forcepos.

	Nouns			Verbs			Adjectives			Adverbs		
measure	P	R	F	P	R	F	P	R	F	P	R	F
path	.471	.471	.471	.430	.430	.430	.708	.708	.708	1.00	1.00	1.00
lch	.468	.468	.468	.437	.437	.437	.708	.708	.708	1.00	1.00	1.00
wup	.457	.457	.457	.367	.367	.367	.708	.708	.708	1.00	1.00	1.00
res	.507	.507	.507	.316	.316	.316	.708	.708	.708	1.00	1.00	1.00
lin	.568	.568	.568	.366	.366	.366	.708	.708	.708	1.00	1.00	1.00
jcn	.598	.598	.598	.414	.414	.414	.708	.708	.708	1.00	1.00	1.00
lesk	.584	.584	.584	.369	.369	.369	.596	.596	.596	1.00	1.00	1.00
vector	.577	.577	.577	.430	.430	.430	.584	.584	.584	1.00	1.00	1.00
random	.374	.374	.374	.195	.195	.195	.422	.422	.422	1.00	1.00	1.00
sense1	.725	.725	.725	.545	.545	.545	.708	.708	.708	1.00	1.00	1.00

Table 167: SENSEVAL-3 results with wntagged format, window=15, contextScore=0.0, pairScore=0.0, –backoff with measure config for lesk and vector and no forcepos.

References

[1] Satanjeev Banerjee and Ted Pedersen. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145, Mexico City, February 2002.

[2] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

[3] A. Budanitsky and G. Hirst. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics*, pages 29–34, Pittsburgh, June 2001.

[4] Robert Burchfield. Frequency analysis of english usage: Lexicon and grammar. by w. nelson francis and henry kucera with the assistance of andrew w. mackie. boston: Houghton mifflin. 1982. x + 561. *Journal of English Linguistics*, 18(1):64–70, April 1985.

[5] Davide Buscaldi and Paolo Rosso. Upv-wsd : Combining different wsd methods by means of fuzzy borda voting. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 434–437, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[6] C. Fellbaum, editor. *WordNet: An electronic lexical database*. MIT Press, 1998.

[7] William Gale, Kenneth Ward Church, and David Yarowsky. Estimating upper and lower bounds on the performance of word-sense disambiguation programs. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 249–256, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[8] Weiwei Guo and Mona Diab. Improvements to monolingual english word sense disambiguation. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 64–69, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[9] G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In C. Fellbaum, editor, *WordNet: An electronic lexical database*, pages 305–332. MIT Press, 1998.

[10] Véronique Hoste, Walter Daelemans, Iris Hendrickx, and Antal van den Bosch. Evaluating the results of a memory-based word-expert approach to unrestricted word sense disambiguation. In *Proceedings of the ACL-02 workshop on Word sense disambiguation*, pages 95–101, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[11] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings on International Conference on Research in Computational Linguistics*, pages 19–33, Taiwan, 1997.

[12] A. Kilgarriff. What is word sense disambiguation good for? In *Proceedings of the NLP Pacific Rim Symposium*, Phuket, Tailand, 1997.

[13] C. Leacock, M. Chodorow, and G. Miller. Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics*, 24(1):147–165, March 1998.

[14] M. E. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine code from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM Press, 1986.

[15] Dekang Lin. Automatic retrieval and clustering of similar words. In *COLING-ACL*, pages 768–774, 1998.

[16] O. Madani and M. Connor. Large-scale many-class learning. In *Proceedings of the SIAM International Conference on Data Mining, SDM-08*, pages 24–26.

[17] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA, 1999.

[18] Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. Unsupervised acquisition of predominant word senses. *Computational Linguistics*, 33(4):553–590, 2007.

[19] J. Michelizzi. Semantic relatedness applied to all words sense disambiguation. Master's thesis, University of Minnesota, Duluth, July 2005.

[20] Rada Mihalcea and Ehsanul Faruque. Senselearner: Minimally supervised word sense disambiguation for all words in open text. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 155–158, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[21] Rada F. Mihalcea. Bootstrapping large sense tagged corpora. In *In Proceedings of the 3rd International Conference on Language Resources and Evaluations (LREC), Las Palmas*, 2002.

[22] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[23] George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. Using a semantic concordance for sense identification. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 240–243, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[24] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A semantic concordance. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 303–308, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[25] Roberto Navigli. Semi-automatic extension of large-scale linguistic knowledge bases. In Ingrid Russell and Zdravko Markov, editors, *FLAIRS Conference*, pages 548–553. AAAI Press, 2005.

[26] Roberto Navigli and Mirella Lapata. Graph connectivity measures for unsupervised word sense disambiguation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1683–1688, Hyderabad, India, 2007.

[27] Yann Ollivier and Pierre Senellart. Finding related pages using green measures: An illustration with wikipedia. In *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI 2007)*, 2007.

[28] Trang Dang H. Palmer M., Tou Ng H. *Word Sense Disambiguation: Algorithms and Applications*, volume 33 of *Text, Speech and Language Technology*, chapter Evaluation of WSD systems., pages 75–106. Springer, New York, July 2006.

[29] S. Patwardhan. Incorporating Dictionary and Corpus Information into a Context Vector Measure of Semantic Relatedness. Master's thesis, University of Minnesota, Duluth, August 2003.

[30] S. Patwardhan, S. Banerjee, and T. Pedersen. Using Measures of Semantic Relatedness for Word Sense Disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257, Mexico City, Mexico, February 2003.

[31] T. Pedersen, S. Banerjee, and S. Patwardhan. Maximizing Semantic Relatedness to Perform Word Sense Disambiguation. Research Report UMSI 2005/25, University of Minnesota Supercomputing Institute, March 2005.

[32] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet::similarity - measuring the relatedness of concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 1024–1025, San Jose, 2004.

[33] Judita Preiss. A detailed comparison of wsd systems: an analysis of the system answers for the SENSEVAL-2 english all words task. *Natural Language Engineering*, 12(3):209–228, 2006.

[34] Judita Preiss, Jon Dehdari, Josh King, and Dennis Mehay. Refining the most frequent sense baseline. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 10–18, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[35] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, January/February 1989.

[36] P. Resnik. WordNet and class–based probabilities. In C. Fellbaum, editor, *WordNet: An electronic lexical database*, pages 239–263. MIT Press, 1998.

[37] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, Montreal, August 1995.

[38] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[39] Hansen A. Schwartz and Fernando Gomez. Acquiring knowledge from the web to be used as selectors for noun sense disambiguation. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 105–112, Manchester, England, August 2008. Coling 2008 Organizing Committee.

[40] Hansen A. Schwartz and Fernando Gomez. Using web selectors for the disambiguation of all words. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 28–36, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[41] Ravi Sinha and Rada Mihalcea. Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *ICSC '07: Proceedings of the International Conference on Semantic Computing*, pages 363–369, Washington, DC, USA, 2007. IEEE Computer Society.

[42] D. D. Sleator and D. Temperley. Parsing english with a link grammar. In *Third International Workshop on Parsing Technologies*, 1993.

[43] Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *CIKM '93: Proceedings of the second international conference on Information and knowledge management*, pages 67–74, New York, NY, USA, 1993. ACM.

[44] Yorick Wilks, Dan Fass, Cheng ming Guo, James E. McDonald, Tony Plate, and Brian M. Slator. Providing machine tractable dictionary tools. *Machine Translation*, 5(2):99–154, 1990.

[45] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, Las Cruces, New Mexico, 1994.

[46] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196, Morristown, NJ, USA, 1995. Association for Computational Linguistics.

[47] G. Zipf. *The Psycho-Biology of Language*. Houghton Mifflin, Boston, MA, 1935.