

Probabilistic Word Sense Disambiguation

Judita Preiss

University of Cambridge, Computer Laboratory, Cambridge CB3 0FD, UK

`Judita.Preiss@cl.cam.ac.uk`

Abstract

We present a theoretically-motivated method for creating probabilistic WSD systems. The method works by composing multiple probabilistic components: such modularity is made possible by an application of Bayesian statistics and Lidstone's smoothing method. We show that a probabilistic WSD system created along these lines is a strong competitor to state-of-the-art WSD systems.

Key words: Word sense disambiguation, probability distribution, system combination

1 Introduction

Word sense disambiguation (WSD) is the task of automatically assigning a sense to a word from a given inventory of senses. In this paper, we present a theoretically-motivated method for creating probabilistic WSD systems, and illustrate it with a system we created using these techniques. We show that probabilistic WSD is a strong competitor to state-of-the-art WSD systems. The main feature of our system is that it is composed of multiple probabilistic components: such modularity is made possible by an application of Bayesian statistics and Lidstone's smoothing method.

It has been observed that sense assignments are usually more accurate when multiple information sources are combined (Stevenson and Wilks, 2001). For example, it is useful to know both the part of speech of the word *bank* (this word has both noun and verb senses) and the co-occurrence information with the word *money*. However, nobody has yet found a provably-optimal method for combining information sources. For example, Stevenson and Wilks (1999) initially used majority voting to combine their information sources, whereas Yarowsky (2000) used hierarchical decision lists. We use Bayesian statistics to combine information sources, generating a probability distribution on senses, which we now motivate.

Many systems generate a single (forced) sense assignment for each word, and it is not clear that this is always optimal: even in a human sense annotation there are words for which the annotators cannot choose just one sense. For example, the word *peculiar* can be assigned either of the two senses *specific* or *characteristic* in the following sentence (from the English all-words task in SENSEVAL-2):

The art of change-ringing is peculiar to the English . . .

In this case, generating a probability distribution over the available senses may be more useful. Such a distribution can also be used in other applications of WSD: for example, the back-off estimates in subcategorization acquisition can be adjusted according to each word's distribution of senses in the corpus (Korhonen and Preiss, 2003). A forced choice, single sense assignment can be easily obtained from a probability distribution, so there is no loss of information in producing a probability distribution.

Each of our information sources produces a probability distribution on senses of the target word, which is smoothed according to our confidence in the given information source. An overall probability distribution on senses is given by a repeated application of Bayes Rule to all the information sources' probability distributions. Bayesian statistics provide a theoretically sound method for combining probabilistic information as opposed to an ad hoc solution such as weighted linear interpolation of probability distributions.

Section 2 describes the origins of the information sources used, and the modifications required to make them probabilistic. The vital topic of smoothing is discussed in Section 3, and Section 4 presents the theoretical background for our method of combining modules. The results obtained on the SENSEVAL-2 English all words task are discussed in Section 5, and the paper closes with a summary of its contributions and avenues for future work (Section 6).

2 Information Sources

Our probabilistic WSD system is implemented within a modular framework. It is composed of a number of information sources which are probabilistic WSD modules, each producing a probability distribution. The main advantage of this design is its flexibility: it is possible to add or remove modules very easily. A modular setup allows a systematic exploration of combinations of modules to optimize performance.

Our modules are based on parts of known WSD algorithms, which we re-implement. Each module is modified to be self-contained and to produce a

probability distribution.¹ We draw from the work of Yarowsky (2000), Mihailescu and Moldovan (2002) and Pedersen (2002), and we also implement a number of baseline modules.

2.1 Unsupervised Modules

We use two types of modules in our WSD system: unsupervised and supervised. In this work, an unsupervised module is a module which our system does not need to train.

2.1.1 Frequency Module

It has been observed that it is quite difficult for WSD systems to beat the most frequent sense baseline (a system selecting for each word its most frequent sense). We reason that the most frequent sense provides a good default value for words which do not obviously (from another module) have another sense, and thus a most frequent sense module forms a part of our system. As a side effect, with this module in place, it is trivial to evaluate the performance of the baseline on each of the data sets.

Usually a most frequent sense baseline returns the most frequent sense (derived from some other corpus) given the word’s (known or guessed) part of speech. So if we know that in our text the word *dog* is being used as a noun, the most frequent sense baseline will return the domestic dog sense of the word (rather than e.g., the unpleasant woman sense).

We require all of our modules to produce a probability distribution. WordNet, the dictionary we employ in this work, contains two types of information we could use for this purpose: a manual ranking of senses created on the basis of their perceived frequency in English, and the frequency of occurrence of each sense within the genre balanced SEMCOR corpus. Each information type can be converted into a probability distribution.

The manual ranking of senses could be used in conjunction with Zipf’s law to yield a distribution on senses (given the part of speech). Zipf’s law (Zipf, 1949) tells us that frequency f_r of the r th ranked sense is given by a power

¹ A probability distribution is created over WordNet senses, as well as an extra sense ‘none’, not present in WordNet. Without the extra sense, each word present in WordNet would have to be assigned a WordNet sense. However, WordNet only covers nouns, verbs, adjectives and adverbs. So a word like *no* could never take the ‘negation’ sense (which does not appear in WordNet) rather than the less frequent ‘number’ sense (which is listed in WordNet).

law (i.e. resulting in a zeta distribution over senses):

$$f_r = ar^{-b}$$

for some positive a, b , such that

$$\sum_{i \in \{\text{senses}\}} f_{\{r:r \text{ rank of sense } i\}} = 1$$

The f_r would be normalized to produce a probability distribution over senses. However, some words are exceptions to Zipf’s law; the SEMCOR frequency of their topmost senses is quite close (e.g., *bear* which has 15 senses, and the top three verb senses have frequencies 25, 17, and 13). In this case, the probability distribution arising from Zipf’s frequencies would be very different to the observed one, and so this method is insufficient.

Making use purely of the SEMCOR frequency counts is also not sufficient. A probability distribution in this case would be obtained by normalising the SEMCOR frequencies within each PoS. However, it is possible for multiple senses to have identical SEMCOR frequency (often zero). The frequency module would then not distinguish between such senses.

We would therefore like to take into account both the frequency counts and the manual ranking. We achieve this by smoothing the frequency counts of the word w with n senses s_1, \dots, s_n by the inverted rank (ir) as follows:

$$f(s_i) = o(s_i) + ir(s_i)$$

where $o(s_i)$ is the observed SEMCOR frequency count and $ir(s_i) = n + 1 - r(s_i)$, with $r(s_i)$ being the rank of the sense s_i . This is illustrated in Table 1: the word *abbey* appeared once in its most frequent sense and neither of its other senses were seen. After smoothing, no two frequencies are the same.

Additionally, in the case of the *frequency module*, it only makes sense to consider the probability of a sense s_i given w ’s part of speech ($\text{pos}(w)$). Thus, the probability of the sense s_i of the word w is given by:

$$\mathcal{P}(s_i | \text{pos}(w) = \text{pos}(s_i)) = \frac{f(s_i)}{\sum_{\{j: \text{pos}(s_j) = \text{pos}(s_i)\}} f(s_j)}$$

where $f(s_i)$ is the smoothed SEMCOR frequency of the sense s_i . In our example, this corresponds to the sense frequencies being divided by $4 + 2 + 1 = 7$.

Sense	Rank	Frequency
Original Entry:		
abbey%1:06:02::	1	1
abbey%1:06:01::	2	0
abbey%1:06:00::	3	0
Modified Entry:		
abbey%1:06:02::	1	$1 + 3 = 4$
abbey%1:06:01::	2	$0 + 2 = 2$
abbey%1:06:00::	3	$0 + 1 = 1$

Table 1
WordNet frequency distribution

2.1.2 Basic Part of Speech Module

WSD systems often make use of the part of speech (PoS) information of words; even the *frequency* module requires information about each word’s PoS to work effectively. This is usually obtained from a tagger, which is run as a pre-processing stage of a WSD system. Often, the tagger is used as a filter: all senses not compatible with the chosen PoS are entirely removed from consideration. However, this means that if a word is assigned a wrong PoS, there is no scope for recovering from the error. We instead use the HMM tagger due to Elworthy (1994), which produces a probability distribution on CLAWS-II PoS tags.

There are 166 tags in the CLAWS-II tagset; it contains tags such as NN1 (singular common nouns), NP (noun proper, neutral for number), or VB0 (to, base form), VVZ (-s form of a lexical verb). For each word w , the tagger returns a probability distribution on these tags. However, WordNet senses are only divided up into four categories: noun, verb, adjective, and adverb. We therefore sum the probabilities obtained over all the noun PoS tags to obtain a probability of the word w being used as a noun, the probabilities of the verb PoS tags make up the probability of w being used as a verb, and so on. The CLAWS-II tagset also contains tags which do not describe nouns, verbs, adjectives or adverbs, such as II (general preposition). The probabilities of these ‘other’ tags make up the probability of the word w having a non-WordNet PoS (and so being used in the extra ‘none’ sense). The full 166 tag tagset is used in some of the subsequent modules (Sections 2.2.1 and 2.2.3).

We illustrate the module on the sentence *She danced with abandon*, where the PoS tagger assigns a high probability to the NN1 tag of the word *abandon*. The original PoS distribution obtained from the tagger can be seen in Table 2. The simplified PoS probabilities over the WordNet (noun, verb, adjective, adverb

and other) classes are shown in Table 3. In this table, the verb tag probabilities of the word *danced* (VVD and VVN) have been merged into a verb probability. Smoothing (Section 3) is applied to all PoS categories containing at least one sense of the word (for example, to the noun and verb categories for the word *danced*), as well as to the ‘other’ category (to allow for senses not in WordNet).

Word	Tag	Prob	Tag	Prob
She	PPHS1	1	–	–
danced	VVD	0.98	VVN	0.02
with	II	0.05	IW	0.95
abandon	NN1	0.96	VV0	0.04

Table 2
Original PoS distribution

Word	Noun	Verb	Adjective	Adverb	Other
She	–	–	–	–	–
danced	0.00	1.00	0.00	0.00	0.00
with	–	–	–	–	–
abandon	0.96	0.04	0.00	0.00	0.00

Table 3
Simplified PoS distribution

Note that in this module, the probability distribution generated is over parts of speech. If we had instead generated the distribution over senses, this would have had the effect of penalizing senses in the more common parts of speech for the word. For example, consider a word with four noun senses and one verb sense. If the noun part of speech is assigned a probability of 3/4 by the tagger and the verb PoS is given 1/4, our system will assign the probability of 3/4 to all the noun senses, rather than producing a probability distribution on senses and thus assigning each noun sense 3/16 ($< 1/4$).

2.2 Supervised Modules

The *frequency* and the *basic part of speech* modules do not need to be trained (at least, not by the WSD system). The remaining modules all require training data and produce probability distributions on senses of the type

$$\mathcal{P}(\text{observed training data}|\text{sense})$$

The form of the distribution produced is dictated by Bayes Rule, which is used to combine modules together (Section 4). We now present the trained

modules, many of which are based on known successful WSD approaches.

2.2.1 PoS Context Modules

The *part of speech* modules work on the principle that PoS tags of words in the context of the target word are informative. Attention needs to be paid to the size of the tagset for this module: if it is too small, it will not be very informative, if it is too large, it will require too many examples to train. We use the 166 tag CLAWS-II tagset to create five *part of speech* modules: part of speech tag of the words one and two to the left (*pos-1*, *pos-2*) and right (*pos1*, *pos2*) of the target word and the word’s own tag (*pos0*). In the case of the *pos0* module, the probability of the tag being *t* given that the sense of the word *w* is *s_i* is:

$$\mathcal{P}(t|s_i) = \frac{\text{no. of occurrences of } w \text{ in sense } s_i \text{ when the PoS tag of } w \text{ is } t}{\text{no. of occurrences of } w \text{ in sense } s_i}$$

For the *pos1* module we would consider the tag of the word immediately following *w*. We present three frequency distributions from the *pos0* module for the word *shirt* ($\mathcal{P}(\text{pos0} = \text{NN1}|\text{shirt}_i)$, $\mathcal{P}(\text{pos0} = \text{NN2}|\text{shirt}_i)$, and $\mathcal{P}(\text{pos0} = \text{VVD}|\text{shirt}_i)$) in Table 4. In this table, $f(\text{sense} \cap \text{pos})$ denotes the number of occurrences of *w* in the given sense with the given PoS tag, and $f(\text{sense})$ is the number of occurrences of *shirt* in the given sense.

Sense id	PoS (<i>t</i>)	$f(\text{sense} \cap \text{pos})$	$f(\text{sense})$	$\mathcal{P}(t s_i)$
shirt%1:06:00::	NN1	8	9	$\frac{8}{9}$
shirt%2:29:00::	NN1	0	1	0
shirt%1:06:00::	NN2	1	9	$\frac{1}{9}$
shirt%2:29:00::	NN2	0	1	0
shirt%1:06:00::	VVD	0	9	0
shirt%2:29:00::	VVD	1	1	1

Table 4
Part of speech distributions for the word *shirt*

2.2.2 Window Module

Words in the context of the target word have been shown to have disambiguating power (Gale et al. (1992)). For example, if the noun *bank* is seen in the context of the word *money*, it is much more likely to have its financial institution noun sense. It is therefore informative to look for certain sense indicative words occurring in a window around the target word. Gale et al. use a ± 50 word context in their disambiguation system.

In the training phase of this module, frequencies of words up to 50 positions to the left and right of each target word (100 words in total) are stored along with the sense of the target word. After removing stoplist words, each of the context words is treated as ‘indicating’ the given sense of the target word.² For each target word w , the training phase results in a number of probability distributions given each sense s_i arising from each of the context words c , as follows:

$$\mathcal{P}(c|s_i) = \frac{\text{no. of occurrences of } c \text{ in the context of } w \text{ in sense } s_i}{\text{no. of occurrences of } w \text{ in sense } s_i}$$

2.2.3 PoS Trigram Module

The *trigram* module is very similar to the *window* module described in the preceding section. It is also based on co-occurrence information, this time looking at a group of three adjacent words. A number of studies (e.g., Choueka and Lusignan (1985)) have shown that humans only require a very small context of two or three words for disambiguation, justifying the small window size. Using trigrams for WSD has been successfully employed by Pedersen (2002). The main defect of the method is its sparse data problem, which is even worse than in the *window* module as we are now concerned with co-occurrences of three words, not just the presence of one word in the vicinity of another.

Using the N-gram Software Package created by Pedersen,³ we implemented a *trigram* module based on part of speech tags. We use PoS tags to abstract away from the more usual word form trigrams, which suffer from sparse data problems. In this module, each sense tagged word w from the training corpus is a part of three trigrams:

Position -1: w is the first word of the trigram.

Position 0: w is the middle word.

Position 1: w is the last word of the trigram.

Table 5 contains an example for the word *Algerian* in the sentence (the relevant PoS tags follow an underscore):

... to_II the_AT Algerian_JJ rebels_NN2 entails_VVZ ...

² Note that removing stoplist words is only done for efficiency reasons – since they do not indicate any particular sense, they would be found to be uninformative in disambiguation.

³ NSP is available from <http://www.d.umn.edu/~tpederse/code.html>

Sense id	Trigram			$f(\text{sense} \cap \text{tri})$	$f(\text{sense})$	$\mathcal{P}(t s_i)$
Position: -1						
algerian%3:01:00::	JJ	NN2	VVZ	1	3	$\frac{1}{3}$
algerian%1:18:00::	JJ	NN2	VVZ	0	0	0
Position: 0						
algerian%3:01:00::	AT	JJ	NN2	1	3	$\frac{1}{3}$
algerian%1:18:00::	AT	JJ	NN2	0	0	0
Position: 1						
algerian%3:01:00::	II	AT	JJ	1	3	$\frac{1}{3}$
algerian%1:18:00::	II	AT	JJ	0	0	0

Table 5

Frequency trigram data for *Algerian*

The probability of a trigram t given the sense s_i of the word w is given by:

$$\mathcal{P}(t|s_i) = \frac{\text{no. of occurrences of } w \text{ in sense } s_i \text{ in trigram } t}{\text{no. of occurrences of } w \text{ in sense } s_i}$$

In our example, all three trigrams, positions -1, 0, and 1, will give rise to higher probabilities for the adjectival sense (indicated by %3 in its senseid) of the word *Algerian*.

2.2.4 Lemma Co-occurrence Modules

Although we do not have enough data to train trigrams on surrounding lemmas, bigrams (two adjacent words) have also been found useful in WSD. Bigrams represent a large part of the local context which was found sufficient by human annotators (Choueka and Lusignan, 1985), and in certain cases may be more informative than the *window* module. For example, the fact that the word *to* occurs within the ± 50 word window of the target word is uninformative, whereas if we know that it immediately precedes the target word, the target word is likely to take its verb senses.⁴

We therefore create a co-occurrence module by extracting lemmas surrounding our target word (we focus at words one and two to the left, *lemma-1*, *lemma-2*, and right, *lemma1*, *lemma2*, of the target word). We use the frequencies of co-occurrence to produce a probability distribution; for *lemma-1* this is a probability of the target word w_t being preceded by the lemma l_p given that

⁴ Note that in the case of the *lemma* modules, the stoplist is not applied.

w_t has sense s_i :

$$\mathcal{P}(l_p|s_i) = \frac{\text{no. of occurrences of } w_t \text{ in sense } s_i \text{ preceded by } l_p}{\text{no. of occurrences of } w_t \text{ in sense } s_i}$$

Prec. Word (w_p)	Sense Id	$f(\text{sense} \cap w_p)$	$f(\text{sense})$	$\mathcal{P}(w_p s_i)$
to	burrow%2:38:00::	1	1	1
to	burrow%1:17:00::	0	1	0
the	burrow%2:38:00::	0	1	0
the	burrow%1:17:00::	1	1	1

Table 6

Lemma co-occurrence data for *burrow*

An example for the word *burrow* and its preceding word can be found in Table 6. We can see that if the preceding word is *to*, the word *burrow* is more likely to have its verb sense (denoted by the %2 in its sense id), whereas if the preceding word is *the*, the noun sense (%1) is more likely.

2.2.5 Head Module

Information can also be gained from the distance to the nearest phrasal head and its type (we only consider noun and verb phrases in this work). We use the Briscoe and Carroll (2002) RASP parser⁵ to obtain grammatical relations (GRs) between words in each sentence. The GRs produced include subject, object, modifier, etc. See Figure 1 for an example of the sentence

The school grounds are large.

The head of the noun phrase “*The school grounds*” can be discovered to be *grounds* because it does not modify another word in an *ncmod* relation (unlike the word *school*), and is also modified by a determiner (the *detmod*) relation.

```
(nsubj are grounds _)
(xcomp _ are large)
(ncmod _ grounds school)
(detmod _ grounds The)
```

Fig. 1. GRs for *The school grounds are large*

The usefulness of the *head* module is easiest to observe if the following word is the head of a phrase. An example for the word *Anglican* followed by a noun

⁵ The RASP parser is available from <http://www.cogs.susx.ac.uk/lab/nlp/rasp/>

phrase head is shown in Table 7. The probability that word w in sense s_i is followed by a head of an NP is given by:

$$\mathcal{P}(\text{NP}|s_i) = \frac{\text{no. of occurrences of } w \text{ in sense } s_i \text{ followed by an NP head}}{\text{no. of occurrences of } w \text{ in sense } s_i}$$

Sense Id	$f(\text{sense} \cap \text{NP})$	$f(\text{sense})$	$\mathcal{P}(\text{NP} s_i)$
anglican%3:01:00::	0	0	0
anglican%1:18:00::	6	6	1

Table 7

Head data for *Anglican* + NP head

We can see that the adjective sense (sense id containing %3) is more likely than the noun sense; this is intuitively plausible since the head of a noun phrase must be a noun, and we are more likely to have an adjective preceding a noun. This module trains probabilities based on the word one and two to the left and right of the target word being a head of a noun or a verb phrase.

2.2.6 Grammatical Relation Module

Our last module exploits noun–verb relations obtained from the RASP parser, along the lines of Hindle (1990) and Resnik (1992). In the training phase, the GRs of each noun target word are examined and information about it being a subject, direct or indirect object are stored together with the relevant verb. An example is shown in Table 8, for the word *attitude*. The table shows the definition of two of the senses of the word along with the WordNet example sentences. The last column shows the information which we would acquire from the sentence.

Sense	Definition	Example sentence	GR
1:04:00	a theatrical pose created for effect	the actor struck just the right attitude	struck dobj
1:09:00	a complex mental state involving beliefs and feelings and values and dispositions to act in certain ways	he had the attitude that work was fun	have dobj

Table 8

Grammatical relation for *attitude*

This module is smoothed, but unlike Resnik’s work, we do not group together ‘similar’ verbs. Resnik observed that in its basic form, this approach suffered from the sparse data problem. This is to be expected, the amount of sense tagged training data is limited, and we are reducing it to the sentences that our parser can parse as well as only focusing on the words filling the grammatical

roles we are interested in. He therefore grouped together similar verbs (verbs in the same WordNet class) and used the training data for each verb in a group as training data for the whole group. Since this is not the only module used in our approach, we do not carry out this generalization and the module returns a uniform distribution on senses when there is no training data for a particular word.

3 Smoothing

It is necessary to smooth the probability distributions produced by all the modules – if we only observed a word once in the training corpus, this will give probability one to all the observed patterns of that occurrence. This would mean that we could never assign a different sense to this word. However, we do not want to discard the one instance. In the case of infrequent words, this may be the only context the word is ever used in. We therefore need to smooth the frequency counts obtained to allow all possible senses, but to still favour the observed patterns.

The chosen method of smoothing needs to satisfy the following requirements:

- (1) The smoothing value for each module should reflect the confidence we have in that module.
- (2) It needs to be possible to make smoothing word specific: it is possible that a module is very good at resolving particular words, but not others.
- (3) The less confident we are in a module, the more the output probability distribution should resemble a uniform distribution, so as not to affect the combination of the modules (see Section 4).
- (4) The probability distribution generated by trained modules for words not seen in the training corpus should be uniform.

The points above illustrate that we are intentionally not smoothing with a Zipfian (zeta) distribution; a Zipfian distribution would favour the more frequent senses (for which there is an explicit *frequency module*), thus making senses other than the most frequent sense extremely unlikely to be favoured.⁶ The method for combining modules which we employ allows us to add in a module which produces a uniform distribution without the new, uninformative module having any effect on the resulting probability distribution on senses. Therefore the requirement for approximating a uniform distribution in points 3 and 4, when the modules do not provide very reliable information, suggests that the

⁶ Note that the modules produce $\mathcal{P}(\text{context}|s_i)$ probabilities, which (unlike the $\mathcal{P}(s_i|\text{context})$ probability) factor out the frequency bias which is present in the training corpus.

best smoothing method is based on a linear interpolation with a uniform prior.

Smoothing methods based on discounting, which decrease the frequency of data observed in the training phase by a small amount and distribute the discounted frequency between the unseen events (e.g. Good (1953)), would not allow us to approximate a uniform distribution when we are not confident in a particular module. Treating point 2 above, we specialize to a particular word w , and use Lidstone’s smoothing (e.g., Manning and Schütze (1999)):

$$\mathcal{P}(\text{context}|s_i) = \frac{C(\text{context} \cap s_i) + \lambda}{C(s_i) + \lambda N}$$

where w is a word with a sense s_i , and for which there are N possible contexts. The C function represents the frequency function and λ is the smoothing value. This is equivalent to a linear interpolation method:

$$\mathcal{P}(\text{context}|s_i) = \mu \frac{C(\text{context} \cap s_i)}{C(s_i)} + (1 - \mu) \frac{1}{N}$$

where

$$\mu = \frac{C(s_i)}{C(s_i) + \lambda N}$$

This smoothing method is frequently criticized for assigning too much probability mass to unseen events. This occurs when N is much greater than $C(s_i)$. However, we have argued above that our combination method is sensitive to deviations from the uniform distribution, and therefore that an interpolation method involving a uniform prior (as frequency is a separate module) is the most suitable. There are two possible methods for addressing the potential difficulty:

- To decrease N , it may be possible to merge contexts; for example, Resnik (1992) grouped together ‘similar’ verbs in a grammatical relation-like module.
- A larger corpus could be used to increase the frequency $C(s_i)$ of senses; for example, automatically building a sense annotated corpus along the lines of Mihalcea and Moldovan (1999).

To satisfy point 1 above, that each module should be smoothed according to our confidence in it, we want to select smoothing values such that the performance of the system is maximized. As the modules perform with different accuracies on individual words, different smoothing values are found for the most frequent words in the test corpus. We split our training corpus (SEMCOR, and the English lexical-sample data) into two parts: $\frac{9}{10}$ for training from which we acquire the frequency counts (such as $C(\text{context} \cap s_i)$); $\frac{1}{10}$, the development corpus, for setting the smoothing values (Chen and Goodman, 1996).

Figure 2 shows the typical shape of a performance against smoothing value graph. As the smoothing value increases, the performance rises sharply reaching a maximum, after which it decreases steadily.⁷ However, it is infeasible to exhaustively search all possible smoothing values. We therefore start with a high smoothing value, which we repeatedly halve until performance on the development corpus starts decreasing. The smoothing value which yielded the highest performance is then used in the testing phase.

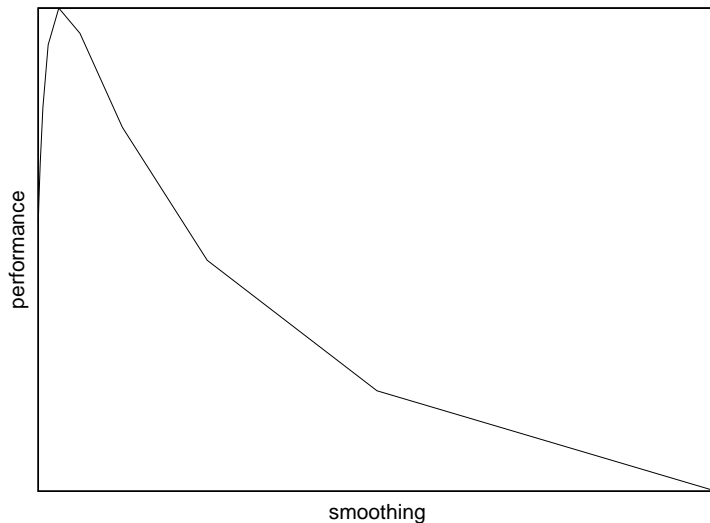


Fig. 2. Performance against smoothing value

We also need smoothing values for words which are not present in the development corpus. These are obtained by finding the optimal smoothing value (the one yielding the highest performance) on all ambiguous words which have not been trained individually. If a word did not appear in the training corpus, its probability distribution becomes the uniform distribution, thus not affecting the choice of sense (see Section 4).⁸

4 Combining Modules

WSD systems based on multiple approaches usually fail to use a theoretically motivated method for their combination. We present a method for combining

⁷ Figure 2 was obtained using the F-measure as an indication of performance. However, we expect the shape of the graph to be the same whether the performance is taken to be precision, recall, or F-measure.

⁸ If a word does not appear in the training corpus at all, all the trained modules will produce a uniform probability distribution and we will be relying on the *frequency* and *basic part of speech* modules to carry out a sense annotation.

our modules based on Bayes Rule:

$$\mathcal{P}(A|B) = \frac{\mathcal{P}(A \cap B)}{\mathcal{P}(B)}$$

As a consequence of Bayes Rule, the probability distributions for individual modules are multiplied together to yield a final probability distribution on senses, which can be later converted into a single sense assignment or used directly in an application.

Suppose we have n modules and the i th module observes context c_{ik_i} around the target word w . The probability that the word w has the sense s_j can be calculated as:

$$\begin{aligned} \mathcal{P}(s_j | c_{1k_1} \cap \dots \cap c_{nk_n}) &= \frac{\mathcal{P}(s_j \cap c_{1k_1} \cap \dots \cap c_{nk_n})}{\mathcal{P}(c_{1k_1} \cap \dots \cap c_{nk_n})} \\ &= \frac{\mathcal{P}(c_{1k_1} \cap \dots \cap c_{nk_n} | s_j) \mathcal{P}(s_j)}{\mathcal{P}(c_{1k_1} \cap \dots \cap c_{nk_n})} && \text{Bayes Rule} \\ &= \frac{\prod_i \mathcal{P}(c_{ik_i} | s_j) \mathcal{P}(s_j)}{\mathcal{P}(c_{1k_1} \cap \dots \cap c_{nk_n})} && \text{Independence} \end{aligned}$$

where $\mathcal{P}(s_j)$ is obtained by combining the *frequency module* with the *basic part of speech module*, i.e.

$$\mathcal{P}(s_j) = \mathcal{P}(s_j | \text{pos}(w) = \text{pos}(s_j)) \mathcal{P}(\text{pos}(s_j) = \text{pos}(w))$$

Note that we assume independence between modules – an assumption which does not hold in general. However, factoring out dependencies in general is extremely difficult as they are usually hidden. In the special case of the frequency information, Bayes Rule helps us remove the frequency bias from the supervised modules, as we are considering the probability that a particular context is employed with the given sense, rather than the probability that the given sense is employed with a particular context. This makes the supervised modules independent from the *frequency module*. Removing dependencies between the supervised modules would require an enormous amount of training data and storage space. For example, to work out the dependencies between the *lemma-1* and the *lemma1* modules we would need to store information about the preceding and following word for every target word. This would not be feasible given the amount of training data and the number of modules we are considering.

We illustrate our combination method on two trained modules *lemma-1* and *lemma-2*. Consider the word *admirable*, which has two adjectival senses *estimable* and *pleasing*. The probability combination is presented in Table 9; the

table contains values for the component probabilities $P(s_i)$ obtained from the *frequency module* and $P(l|s_i)$, the probability of lemma l given the sense s_i from the *lemma modules*. The combination is simply implemented by multiplying the probability distributions together and normalizing the result. In our example, the *estimable* sense of *admirable* in the fragment *most admirable american* has the probability:

$$\frac{\frac{5}{7} * \frac{2}{3} * \frac{2}{3} + \frac{5}{7} * \frac{2}{3} * \frac{1}{3} + \frac{2}{7} * \frac{1}{3} * \frac{1}{3} + \frac{2}{7} * \frac{1}{3} * \frac{2}{3}}{\frac{5}{7} * \frac{2}{3} * \frac{2}{3} + \frac{5}{7} * \frac{2}{3} * \frac{1}{3} + \frac{2}{7} * \frac{1}{3} * \frac{1}{3} + \frac{2}{7} * \frac{1}{3} * \frac{2}{3}} = \frac{5}{9}$$

Sense s_i	$P(s_i)$	<i>Lemma-1</i>		<i>Lemma1</i>		$P(s_i l_1 \cap l_2)$
		l_1	$P(l_1 s_i)$	l_2	$P(l_2 s_i)$	
estimable	5/7	most	2/3	american	2/3	5/9
				type	1/3	5/18
pleasing	2/7	most	1/3	american	1/3	1/18
				type	2/3	1/9

Table 9
Combination of modules for the word *admirable*

5 Results

5.1 SENSEVAL Evaluation

Until the creation of the SENSEVAL evaluation exercise in 1998, there was no consensus on evaluation methods for WSD, which made comparing published system performance results impossible. Firstly, systems differed in the number of words they were designed to attempt (e.g., Bruce and Wiebe (1994) evaluated their system on the single word *interest*, whereas the system of Stevenson (1999) was evaluated on all labelled words in the SEMCOR corpus). Secondly, systems used different evaluation corpora which can lead to large performance differences. Thirdly, systems were based on different sense inventories.⁹ To enable the WSD community to compare systems directly, the SENSEVAL WSD evaluation exercise was created (Kilgarriff and Palmer, 2000).

We evaluate the system on the English all words task from SENSEVAL-2 (Palmer et al., 2002), see Section 5.2 for a discussion of why the English

⁹ A more fine-grained sense inventory will mean that the WSD task is more difficult and therefore will lead to lower performance.

lexical sample task is not suitable. In the English all words task in SENSEVAL-2, nouns, verbs, adjectives and adverbs in three given texts (taken from the Wall Street Journal) were manually annotated to create a gold standard. This resulted in 1082 distinct words in the texts, corresponding to 2473 instances to be labelled¹⁰ with senses by participating systems. Twenty one systems sense-annotated the given texts, and submitted their answers for a calculation of precision and recall against the gold standard. The results for these systems are presented in Table 10. The table contains a tick in the second column, if the given system incorporates modules trained on annotated data. It also includes results for the correct file (which is a copy of the gold standard) and for two baselines (most frequent sense and random sense).¹¹

The most frequent sense baseline in the table is a ‘perfect’ baseline: it has access to perfect part of speech information and the correct lemmas (and so knows about multiwords).¹² Although this baseline will be higher than the most frequent sense baseline found by a genuine system employing a real tagger and multiword detecting component, it is useful since it is independent of any participating system.

5.2 Amount of training data required

We have investigated the effect of increasing the amount of training data on performance of our WSD system. We used the *line* corpus (Leacock et al., 1993), which contains 4148 instances of the word *line* each annotated with one of six senses. The *line* corpus was divided into three corpora: a test corpus, a development corpus (used for acquiring smoothing values), and a training corpus. With 18 training corpus increments, the average performance (F-measure) was found to rise sharply up to 200 training instances after which the gradient begun to level off. A performance plateau for the six sense word *line* was reached at about 500 instances.

In the SENSEVAL-2 English lexical sample task, the number of instances tagged for each word was $75 + 15n$ where n is the number of senses a word has within the chosen part of speech (Kilgarriff, 2002), (Palmer et al., 2002). Two thirds of the total annotated data for each word was used for training. The average number of training instances in the English lexical sample task training corpus

¹⁰ This includes 86 instances labelled with the “U” tag. This label was assigned when the correct sense of a word did not appear in WordNet 1.7 pre-release.

¹¹ Precision for the two baselines was only evaluated on words not labelled with “U”. If these were included, the precision of the most frequent sense would be 64.62% and that of the random choice would be 34.26%.

¹² The perfect part of speech and lemma information was obtained directly from the gold standard.

System	T	Precision	Recall	F-measure
BCU-ehu-dlist-all	✓	57.2%	29.1%	38.5%
CNTS-Antwerp	✓	63.5%	63.5%	63.5%
DIMAP	×	45.1%	45.1%	45.1%
IIT1	×	28.7%	3.3%	6.0%
IIT2	×	32.8%	3.8%	6.8%
IIT3	×	29.4%	3.4%	6.1%
IRST	✓	74.7%	35.7%	48.3%
SMUaw	✓	68.9%	68.9%	68.9%
Sheffield	×	44.5%	20.0%	27.6%
Sinequa-LIA-HMM	✓	61.8%	61.8%	61.8%
Sussex-sel	×	59.8%	14.0%	22.6%
Sussex-sel-ospd	×	56.6%	16.9%	26.1%
Sussex-sel-ospd-ana	×	54.5%	16.9%	25.8%
UCLA-gchao	✓	50.8%	44.4%	47.4%
UCLA-gchao2	✓	48.3%	45.3%	46.7%
UCLA-gchao3	✓	48.1%	45.1%	46.6%
UNED-AW-T	×	57.4%	56.8%	57.1%
UNED-AW-U	×	55.5%	54.9%	55.2%
USM1	×	34.2%	31.6%	32.8%
USM2	×	36.0%	36.0%	36.0%
USM3	×	33.6%	33.6%	33.6%
correct	–	100.0%	100.0%	100.0%
frequency	–	66.9%	64.6%	65.7%
random	–	35.5%	34.2%	34.9%

Table 10
Precision, recall, and F-measure for all systems

is 121, and 93% of the words have fewer than 200 instances. Given that we found that 200 instances were necessary for a six sense word, and that the average polysemy for the English lexical sample task was 9.1 (rather than the 5.4 average for the English all words task), we believe that the English lexical sample is not suitable for evaluating the true training power of our WSD system (without additional training data).

We present the results of our probabilistic modular WSD system in Table 11. The table contains the precision (number of correct answers divided by the number of words attempted) and recall (number of correct answers divided by the total number of test instances), as well as the F-measure:

$$\text{F-measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Results are shown for two types of evaluation: the forced choice evaluation, in which only one sense tag is assigned to sense tagged words (the sense tag corresponding to the highest probability), and the log odds ratio evaluation (Dagan and Itai, 1994), in which all the senses (with probability p) satisfying

$$\log \left(\frac{\text{highest probability}}{p} \right) \geq T$$

(where T is a threshold value) are returned and their probabilities are normalized. The log odds ratio allows us to return a number of senses if their probability is close to that of the most probable sense. We investigated the value of T , by varying it between 0.002 and 0.4 in steps of 0.002. The optimal value was found to be 0.1, which generated the results in Table 11. However, the results did not deviate much from those given with any of the tested thresholds (the lowest F-measure was only 0.05% lower than the maximum, and occurred when senses with probability up to 0.002 lower than the highest probability were getting included).

Evaluation	Precision	Recall	F-measure
Forced	64.1%	63.0%	63.6%
Log odds	64.1%	63.1%	63.6%
Baseline	61.0%	60.0%	60.5%

Table 11
Results for the English all words task

Given the evaluation method in SENSEVAL, it is not reasonable to score the full probability distribution. Even when the system is very confident of the correct sense assignment, smoothing will ensure that some probability mass will be assigned to the remaining senses. In some applications this may be considered a benefit, but with the current evaluation method it degrades performance.

Table 11 also includes the performance of the baseline, which is lower than the perfect baseline presented in Table 10. This is to be expected, as the

baseline performance is dependent on the performance of the tagger, correct identification of multiwords¹³ and assignments of the “U” tag.

These results can be directly compared with the English all words task results in Table 10, to give our system a ranking among current state-of-the-art systems. The full system containing 27 modules ranks second (when comparing F-measure). With an F-measure of 63.6%, it comes after the SMUaw system.

5.3.1 Choosing the right modules

We investigated the performance of our system with a lower number of modules; there are a number of successful systems using a subset of the information sources we exploit in our full system. As an exhaustive search is not possible (27 modules give rise to 2^{27} possible module combinations), we must restrict their number in a different way. We can judge the usefulness of a module using the smoothing value produced for it – the lower the smoothing value, the more accurate (and therefore useful) a module is. For example, the smoothing value for the *GRs* module is 267, and so this module is close to the uniform distribution.

For the optimization, the program was always invoked with the *frequency* and *basic part of speech* modules. We selected the top 7 modules:¹⁴ *head-1*, *trigrams0*, *lemma-3*, *lemma-2*, *trigrams-1*, *pos-1*, and *pos-3*. A graph depicting the F-measure when various module combinations are invoked is shown in Figure 3. The performance of the baseline (*frequency* and *basic pos*) is shown in the column with no modules on the *x*-axis, any additional modules can be found on the *x*-axis.

The figure shows that it is possible for our system to improve on its performance (reaching the F-measure of 65.3%). The increase in performance when modules are dropped may seem surprising, however there are two possible reasons for this:

- (1) Some modules encode very similar information (e.g., *pos1* and *head1* will both tell us if the following word is a noun), which may be getting over-emphasized (when wrong) when the modules are invoked together so removing them from the combination may result in an increase in performance.

¹³ We do not preprocess multiwords, as these did not appear to be consistently annotated in the gold standard.

¹⁴ The number of modules was chosen so that the whole optimization did not run for too long. Note that the WSD system program was not written with an emphasis on efficiency.

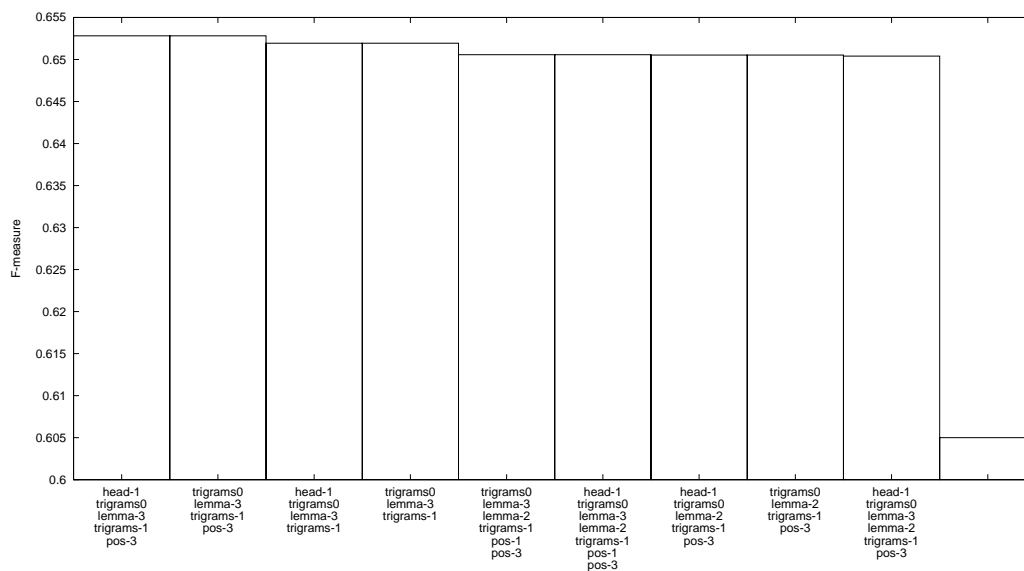


Fig. 3. Combination of modules for English all words task

- (2) The smoothing value may be high for some modules, making the modules approximate the uniform distribution, but the modules may still be having a deleterious effect.

6 Conclusion

We have presented a method for creating a theoretically motivated probabilistic WSD system, based on Bayes' Rule and Lidstone's smoothing, and shown that it is a strong competitor to state-of-the-art WSD systems. In its default form, our WSD system ranks second in the SENSEVAL-2 English all words task. The system does not rely on any externally set thresholds or weights, and can therefore be easily ported to other corpora.

The success of our probabilistic WSD system has demonstrated the effectiveness of probabilistic methods in WSD; this trend can also be observed in other areas of NLP such as statistical parsing, or Ge and Charniak's anaphora resolution algorithm. To create our WSD system, we adapted a number of existing WSD approaches to function as probabilistic modules, and we have given a precise description of these probabilistic modules for others to use in future systems.

Our implementation of Lidstone's smoothing provides a uniform mechanism for weighting modules based on their accuracy, removing the need for an additional confidence weighting scheme; and smoothing values are found for individual words where possible, giving an extra degree of specialization.

We believe that it is possible for the probabilistic combination WSD system to exceed the performance of the SMUaw system; the SMUaw system extracts patterns similar to those in our modules, but it uses a much larger training corpus. In addition to SEMCOR, the system trains on WordNet examples and GENCOR, an automatically created corpus containing about 160,000 tagged words (Mihalcea and Moldovan, 1999). It is therefore possible that our system would exceed the performance of the SMUaw system given the same training corpora, as the SMUaw system doesn't appear to have a method for recovering from mistakes in patterns.

Acknowledgements

This work was supported by UK EPSRC project GR/N36462/93 'Robust Accurate Statistical Parsing'. I would also like to thank my supervisor, Ted Briscoe, and Joe Hurd for reading through previous drafts of this paper. Additionally, the three anonymous reviewers were very helpful in providing many useful comments and suggestions for improvement.

References

- Briscoe, E. J., Carroll, J., 2002. Robust accurate statistical annotation of general text. In: Proceedings of the 3rd International Conference on Language Resources and Evaluation. pp. 1499–1504.
- Bruce, R., Wiebe, J., 1994. Word-sense disambiguation using decomposable models. In: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics. pp. 139–145.
- Chen, S. F., Goodman, J., 1996. An empirical study of smoothing techniques for language modeling. In: Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics. pp. 310–318.
- Choueka, Y., Lusignan, S., 1985. Disambiguation by short contexts. *Computer and the Humanities* 19, 22–29.
- Dagan, I., Itai, A., 1994. Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics* 20, 563–596.
- Elworthy, D., 1994. Does Baum-Welch re-estimation help taggers? In: Proceedings of the 4th Conference on Applied NLP. pp. 53–58.
- Gale, W., Church, K., Yarowsky, D., 1992. One sense per discourse. In: Proceedings of the 4th DARPA Speech and Natural Language Workshop. pp. 233–237.
- Ge, N., Hale, J., Charniak, E., 1998. A statistical approach to anaphora resolution. In: Proceedings of the Sixth Workshop on Very Large Corpora. pp. 161–170.

- Good, I. J., 1953. The population frequencies of species and the distribution of population parameters. *Biometrika* 40 (3/4), 237–264.
- Hindle, D., 1990. Noun classification from predicate-argument structures. In: *Proceedings of the 28th Annual Meeting of the ACL*. pp. 268–75.
- Kilgarriff, A., 2002. English lexical sample task description. In: *Preiss and Yarowsky (2002)*, pp. 17–20.
- Kilgarriff, A., Palmer, M., 2000. Introduction to the special issue on SENSEVAL. *Computers and the Humanities* 34 (1–2), 1–13.
- Korhonen, A., Preiss, J., 2003. Improving subcategorization acquisition using word sense disambiguation. In: *Proceedings of ACL*. pp. 48–55.
- Leacock, C., Towell, G., Voorhees, E., 1993. Corpus-based statistical sense resolution. In: *Proceedings of the ARPA Workshop on Human Language Technology*. pp. 260–265.
- Manning, C. D., Schütze, H., 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.
- Mihalcea, R., Moldovan, D. I., 1999. An automatic method for generating sense tagged corpora. In: *Proceedings of AAAI-99*. pp. 461–466.
- Mihalcea, R., Moldovan, D. I., 2002. Pattern learning and active feature selection for word sense disambiguation. In: *Preiss and Yarowsky (2002)*, pp. 127–130.
- Palmer, M., Fellbaum, C., Cotton, S., Delfs, L., Dang, H. T., 2002. English tasks: All-words and verb lexical sample. In: *Preiss and Yarowsky (2002)*, pp. 21–24.
- Pedersen, T., 2002. Machine learning with lexical features: The Duluth approach to Senseval-2. In: *Preiss and Yarowsky (2002)*, pp. 139–142.
- Preiss, J., Yarowsky, D. (Eds.), 2002. *Proceedings of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguating Systems*.
- Resnik, P., 1992. Wordnet and distributional analysis: a class-based approach to lexical discovery. In: *Workshop Notes, Statistically-Based NLP Techniques AAAI*. pp. 54–64.
- Stevenson, M., 1999. Multiple knowledge sources for word sense disambiguation. Ph.D. thesis, University of Sheffield.
- Stevenson, M., Wilks, Y., 1999. Combining weak knowledge sources for sense disambiguation. In: *Proceedings of the International Joint Conference for Artificial Intelligence (IJCAI-99)*. pp. 884–889.
- Stevenson, M., Wilks, Y., 2001. The interaction of knowledge sources in word sense disambiguation. *Computational Linguistics* 27 (3), 321–349.
- Yarowsky, D., 2000. Hierarchical decision lists for word sense disambiguation. *Computers and the Humanities* 34 (1/2), 179–186.
- Zipf, G. K., 1949. *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge MA.