Smoothing and Word Sense Disambiguation

Eneko Agirre and David Martinez

IXA NLP Group
University of the Basque Country
649 pk. 20.080 Donostia, Spain
{eneko,davidm}@si.ehu.es

Abstract. This paper presents an algorithm to apply the smoothing techniques described in [1] to three different Machine Learning (ML) methods for Word Sense Disambiguation (WSD). The method to obtain better estimations for the features is explained step by step, and applied to n-way ambiguities. The results obtained in the Senseval-2 framework show that the method can help improve the precision of some weak learners, and in combination attain the best results so far in this setting.

1 Introduction

Many current Natural Language Processing (NLP) systems rely on linguistic knowledge acquired from tagged text via Machine Learning (ML) methods. Statistical or alternative models are learned, and then applied to running text. The main problem faced by such systems is the sparse data problem, due to the small amount of training examples. Focusing on Word Sense Disambiguation (WSD), only a handful of occurrences with sense tags are available per word. For example, if we take the word *channel*, we see that it occurs 5 times in SemCor [2], the only all-words sense-tagged corpus publicly available: the first sense has four occurrences, the second a single occurrence, and the other 5 senses are not represented. For a few words, more extensive training data exists. Senseval-2 [3] provides 145 occurrences of *channel*, but still some of the senses are represented by only 3 or 5 occurrences.

It has to be noted that both in NLP and WSD, most of the events occur rarely, even when large quantities of training data are available. Besides, fine-grained analysis of the context requires that it is represented with many features, some of them rare, but which can be very informative. Therefore, the estimation of rare-occurring features might be crucial to have high performances.

Smoothing is the technique that tries to estimate the probability distribution that approximates the one we expect to find in held-out data. In WSD, if all occurrences of a feature for a given word occur in the same sense, Maximum Likelihood Estimation (MLE) would give a 0 probability to the other senses of the word given the feature, which is a severe underestimation. We will denote these cases as X/0, where X is the frequency of the majority sense, and zero is the frequency of the other senses.

For instance, if the word *Jerry* occurs in the context of *art* only once in the training data with a given sense, does it mean that the probability of other senses of *art* occurring in the context of *Jerry* is 0? We will see in Section 4.3 that this is not the case, and that

the other senses are nearly as probable. Our smoothing study will show for this feature of the word *art* that the smoothed ratio should be closer to 1/1.

In this paper, we follow the smoothing method proposed by Yarowsky in his PhD dissertation [1], and present a detailed algorithm of its implementation for the WSD problem, defining some of the parameters used, alongside the account of its use by three different ML algorithms: Decision Lists (DL), Naive Bayes (NB), and Vector Space Model (VSM). The impact of several smoothing strategies is also presented, and the results indicate that the smoothing method explored in this paper is able to make both statistically motivated methods (DL and NB) perform at very high precisions, comparable and in some cases superior to the best results attained in the Senseval-2 competition. We also show that a simple combination of the methods and a fourth system based on Support Vector Machines (SVM) attains the best result for the Senseval-2 competition reported so far.

An independent but related motivation for this work is the possibility to use smoothing techniques in bootstrapping approaches. Bootstrapping techniques such as [4] have shown that if we have good seeds, it could be possible to devise a method that could perform with quality similar to that of supervised systems. Smoothing techniques could help to detect rare but strong features which could be used as seeds for each of the target word senses.

The paper is organized as follows. Section 2 presents the experimental setting. Section 3 introduces smoothing of feature types and Section 4 presents the detailed algorithm with examples. Section 5 presents the results and comparison with other systems, and, finally, the last section draws some conclusions.

2 Experimental setting

In this section we describe the target task and corpus used for evaluation, the type of features that represent the context of the target word, and the ML algorithms applied to the task.

2.1 Corpus

The experiments have been performed using the Senseval-2 English Lexical-Sample data [3]. This will allow us to compare our results with the systems in the competition and with other recent works that have focused on this dataset. The corpus consists on 73 target words (nouns, verbs, and adjectives), with 4,328 testing instances, and approximately twice as much training. We used the training corpus with cross-validation to estimate the C parameter for the SVM algorithm, and to obtain the smoothed frequencies for the features (see below). For the set of experiments in the last section, the systems were trained on the training part, and tested on the testing part.

A peculiarity of this hand-tagged corpus is that the examples for a given target word include multiword senses, phrasal verbs, and proper nouns. A separate preprocess is carried out in order to detect those cases with 96.7% recall.

2.2 Features

The feature types can be grouped in three main sets:
Local collocations: bigrams and trigrams formed with the words around the target. These features are constituted by lemmas, word-forms, or PoS tags [1]. Other local features are those formed with the previous/posterior lemma/word-form in the context.
Syntactic dependencies: syntactic dependencies were extracted using heuristic patterns, and regular expressions defined with the PoS tags around the target [2]. The following relations were used: object, subject, noun-modifier, preposition, and sibling.
Bag-of-words features: we extract the lemmas of the content words in the whole context, and in a $\pm$4-word window around the target. We also obtain salient bigrams in the context, with the methods and the software described in [6].

2.3 ML methods

Given an occurrence of a word, the ML methods below return a weight for each sense $(weight(s_k))$. The sense with maximum weight will be selected. The occurrences are represented by the features in the context (f_i).

The **Decision List (DL)** algorithm is described in [1]. In this algorithm the sense s_k with the highest weighted feature f is selected, as shown below. In order to avoid 0 probabilities in the divisor, we can use smoothing or discard the feature altogether.

$$weight(s_k) = \arg\max_f \ \log(\frac{P(s_k|f)}{\sum_{j \neq k} P(s_j|f)}) \tag{1}$$

The **Naive Bayes (NB)** method is based on the conditional probability of each sense s_k given the features f_i in the context. It requires smoothing in order to prevent the whole productory to return zero because of a single feature.

$$weight(s_k) = P(s_k) \ \prod_{i=1}^{m} P(f_i|s_k) \tag{2}$$

For the **Vector Space Model (VSM)** method, we represent each occurrence context as a vector, where each feature will have a 1 or 0 value to indicate the occurrence/absence of the feature. For each sense in training, one centroid vector is obtained ($\boldsymbol{C_{s_k}}$). These centroids are compared with the vectors that represent testing examples ($\boldsymbol{f}$), by means of the cosine similarity function. The closest centroid assigns its sense to the testing example. No smoothing is required to apply this algorithm, but it is possible to use smoothed values instead of 1s and 0s.

$$weight(s_k) = cos(\boldsymbol{C_{s_k}}, \boldsymbol{f}) = \frac{\boldsymbol{C_{s_k}} \cdot \boldsymbol{f}}{|\boldsymbol{C_{s_k}}||\boldsymbol{f}|} \tag{3}$$

Regarding **Support Vector Machines (SVM)** we utilized SVM-Light, a public distribution of SVM by [7]. We estimated the soft margin (C) using a greedy process in

[1] The PoS tagging was performed with the fnTBL toolkit [5].
[2] This software was kindly provided by David Yarowsky's group, from the Johns Hopkins University.

cross-validation on the training data. The weight for each sense is given by the distance to the hyperplane that supports the classes, that is, the sense s_k versus the rest of senses.

3 Feature-type smoothing

We have already seen in the introduction that estimating X/0 features with MLE would yield a probability $P(s|f) = 1$ for the majority sense and a probability $P(s|f) = 0$ for the minority senses, which is an underestimation. Features with X/0 counts are usual when the training data is sparse, and these values must be smoothed before they are fed to some learning algorithms, such as DL or NB, as they lead to undetermined values in their formulations.

Other distributions, such as X/1, X/2, ... can also be estimated using smoothing techniques. [1] argues that the probability of the second majority sense in X/1 distributions would be overestimated by MLE. For intermediate cases, such as X/2, X/3, etc. it is not clear whether the effort of modeling would be worth pursuing. For higher frequencies, using the raw frequency could be good enough. In this work we focused in X/0 and X/1 distributions.

The smoothing algorithm shown here (which we will call *feature-type smoothing*) follows the ideas of [1]. The main criteria to partition the training data has been to use raw frequencies and feature types (e.g. *prev_N_wf*, feature type that represents the first noun word-form to the left of the target). Raw frequency is the most important parameter when estimating the distribution, and joining features of the same type is a conservative approach to partition the data. Therefore we join all occurrences of the *prev_N_wf* feature type that have the same frequency distribution for the target word, e.g. 1/0. This way, we perform smoothing separately for each word.

We could use the smoothed values calculated in this manner directly, but many data points would still be missing. For instance, when studying *prev_N_wf* in the X/0 frequency case for *art*, we found occurrences of this feature type in held-out data in the 1/0, 2/0 and 3/0 cases, but not the rest (4/0 and higher). In this case it is necessary to use interpolation for the missing data points, and we applied log-linear interpolation. The interpolation also offers additional benefits. Firstly, using the slope of the interpolated line we can detect anomalous data (such as cases where 1/0 gets higher smoothed values than 5/0) as we always expect a positive slope, that is, higher ratios deserve higher smoothed values. Secondly, interpolation can be used to override a minority of data points which contradict the general trend. These points will be illustrated in the examples presented in Section 4.3.

However, when using interpolation, we need at least two or three data points for all feature types. For feature types with few points, we apply a back-off strategy: we join the available data for all words in the same Part of Speech. The rationale for this grouping is that strong features for a noun should be also strong for other nouns. In order to decide whether we have enough data for a feature type or not, we use the number of data points (minimum of three) available for interpolation. In order to check the validity of the interpolation, those cases where we get negative slope are discarded.

Original	Held-out			Accumulated				Interpolated			
X Y	X'	Y'	X'/Y'	X'	Y'	X'/Y'	log(X'/Y')	X"	Y"	X"/Y"	log(X"/Y")
1 0	4	4	1	4	4	1.00	0.00	1	0.91	1.10	0.09
2 0	6	1	6	10	5	2.00	0.69	2	1.18	1.69	0.52
3 0	2	0	∞	12	5	2.4	0.88	3	1.14	2.63	0.96
								4	0.98	4.08	1.40
								...			

Table 1. Smoothing table for the feature *prev_N_wf* and the word *art* (X/0 distribution).

4 Feature-type smoothing algorithm

There are two steps in the application of the smoothing algorithm to the disambiguation task. First, we use the available training data in cross-validation, with an interpolation method, in order to estimate the smoothing tables for each feature type with X/0 or X/1 raw frequency. Second, the interpolated tables are accessed on the disambiguation phase, when the WSD methods require them. Sections 4.1 and 4.2 present the algorithms, and Section 4.3 shows some illustrative examples.

4.1 Building smoothing tables

We build two kinds of smoothing tables. The first kind is the application of the grouping strategy based on feature types and frequency distributions. Two tables are produced: one at the word level, and another at the PoS level, which we will call *smoothed tables*. The second kind is the result of the interpolation method over the two aforementioned tables, which we will call *interpolated tables*. All in all, four tables are produced in two steps for each frequency distribution (X/0 and X/1).

1) Construct smoothing tables for each target word and for each PoS. For each feature type (e.g.: *prev_N_wf*), we identify the instances that have X/0 or X/1 distributions (e.g. *prev_N_wf Aboriginal*) and we count collectively their occurrences per sense. We obtain tables with (X',Y') values for each word, feature type and pair (X,Y); where (X,Y) indicate the values seen for each feature in the training part, and (X',Y') represent the counts for all the instances of the feature type with the same (X,Y) distribution in the held-out part.

We perform this step using 5-fold cross-validation on the training data. We separate in a stratified way[3] the training data in two parts: estimation-fold (4/5 of the data) and target-fold (1/5 of the data), which plays the role of the held-out data. We run the algorithm five times in turn, until each part has been used as target. The algorithm is described in detail in Figure 1 for the X/0 case (the X/1 case is similar). Note that the X count corresponds to the majority sense for the feature, and the Y count to all the rest of minority senses for the feature. For example, we can see in the held-out columns in Table 1 the (X',Y') counts obtained for the feature type *prev_N_wf* and the target word *art* in the Senseval-2 training data for the X/0 cases.

[3] By stratified, we mean that we try to keep the same proportion of word senses in each of the 5 folds.

1. Construct word smoothing tables for X/0 (X0)
- For each fold from training-data (5 folds)
  Build $count(f, w, sense)$ for all senses from the estimation-folds (4 folds)
  For each word w, for each feature f in each occurrence in target-fold (1 fold)
     get $count'(f, w, sense)$ for all senses of w in target-fold
     If distribution of $count'(f, w, sense)$ is of kind X/0 (X0) then
        For each sense
           if $sense = s.\max_s\ count(f, w, s)$
           then   # sense is major sense in estimation-fold
              increment X' in table_word_X0(w,type(f),X)
           else
              increment Y' in table_word_X0(w,type(f),X)

- Normalize all tables: X' is set to X, and Y' := Y'X'/X
  Output (No need to keep X'): $normtable\_word\_X0(w, type(f), X) := Y'$

2. Log linear Interpolation
- Accumulate X' and Y' values
- Map into linear space:
  $logtable\_word\_X0(w, type(f), X) :=$
     $log(acctable\_word\_X0(w, type(f), X).X'/acctable\_word\_X0(w, type(f), X).Y')$
- Do linear interpolation of $logtable$: $sourcepoint(w, type(f)) = a_0$,
     $gradient(w, type(f)) = a_1$
- For each X from 1 to ∞
     $interpolatedtable\_word\_X0(w, type(f), X) := X/(e^{a_0 + a_1 X})$

Fig. 1. Construction of smoothing tables for X/0 features for words. The X/1 and PoS tables are built similarly.

2) Create interpolation curves. From the smoothing tables, we interpolate curves for feature types that have at least 3 points. The process is described in detail in the second part of Figure 1. We first accumulate the counts in the smoothed table from the previous step. The "Accumulated" columns in Table 1 show these values, as well as the X/Y ratio and its logarithm. The Y value is then normalized, and mapped into the logarithmic space. We apply a common linear interpolation algorithm called *least square method* [8], which yields the starting point and slopes for each interpolation table. If we get a negative slope, we discard this interpolation result. Otherwise, we can apply it to any X, and after mapping again into the original space we get the interpolated values of Y, which we denote Y". Table 1 shows the Y" values, the X"/Y" ratios, and the log values we finally obtain for the *prev_N_wf* example for *art* for $X = 1..4$ and $Y = 0$ ("Interpolated" columns). The X"/Y"ratios indicate that for X values lower than 4, the feature type is not reliable, but for $X >= 4$ and $Y = 0$, this feature type can be used with high confidence for *art*.

4.2 Using the smoothed values

The process to use the smoothed values in testing is described in Figure 2. There we see that when we find X/0 or X/1 distributions, the algorithm resorts to the *obtain_smoothed_value* function to access the smoothing tables. The four tables constructed in the previous section are all partial, i.e. in some cases there is no data available for some of the senses. The tables are consulted in a fixed order: we first check the interpolated table for the target word; if it is not available for the feature type, we access

```
Given an occurrence of a word w in testing, for each feature f in the context:
   Get count(f, w, sense) for all senses from all training (all 5 folds)
   If counts are not X/1 or X/0 then
      For each sense:
         count'(f, w, sense) := count(f, w, sense)

   Elseif count is X/Y (where Y is 1 or 0) then
      If Y' = obtain_smoothed_value(X, Y)
      Then
         For each sense
            If sense = s.max_s count(f, w, s) then                    #(MAJOR SENSE)
               count'(f, w, sense) = X
            Elsif sense = 2nd_sense then    #(ONLY IF Y=1, WHERE A MINORITY SENSE
                                                                       OCCURS ONCE)
               count'(f, w, sense) := Y'             #(SECOND SENSE GETS MORE CREDIT)
            Else
               count'(f, w, sense) := Y'/|othersenses| # (DISTRIBUTE WEIGHT UNIFORMLY
                                                              AMONG MINOR SENSES)

      Else                              # (THERE IS NO SMOOTHING DATA FOR THIS X/Y)
         DISCARD                                          #(THIS IS POSSIBLE FOR DL)
         For each sense
            If sense = s.max_s count(f, w, s) then                  # (MAJOR SENSE)
               count'(f, w, sense) := X
            Elsif sense = 2nd_sense then    #(ONLY IF Y=1, WHERE A MINORITY SENSE
                                                                       OCCURS ONCE)
               count'(f, w, sense) := 1            # (SECOND SENSE GETS MORE CREDIT)
```

Fig. 2. Application of Feature-type smoothing to DL, NB and VSM.

the interpolated table for the PoS of the target word. Otherwise, we resort to the non-interpolated smoothing table at the word level. Finally we access the non-interpolated smoothing table for the PoS.

In cases were the four tables fail to provide information, we can benefit from additional smoothing techniques. The three ML methods that we have applied have different smoothing requirements, and one of them (NB) does need a generally applicable smoothing technique:

DL: as it only uses the strongest piece of evidence, it can discard X/0 features. It does not require X/1 smoothing either.
NB: It needs to estimate all single probabilities, i.e. all features for all senses, therefore it needs smoothing in X/0, X/1 and even X/2 and larger values of Y. The reason is that in the case of polisemy degrees larger than 2, the rare senses might not occur for the target feature and could lead to infinite values in Equation (2).
VSM: it has no requirement for smoothing.

In order to check the impact of the various smoothing possibilities we have devised 6 smoothing algorithms to be applied with the 3 ML methods (DL, NB, and VSM). We want to note that not requiring smoothing does not mean that the method does not profit from the smoothing technique (as we shall see in the evaluation). For the baseline smoothing strategy we chose both “no smoothing”, and “fixed smoothing”; we also tried a simple but competitive method from [9], denoted as “Ng smoothing” (methods

X Y	*prev_N_wf*			*win_cont_lem_context*			*win_2gram_context*		
	X' Y'	X" Y"		X' Y'	X" Y"		X' Y'	X" Y"	
1 0	4 4	1	0.91	517 1187	1	2.24	63 150	1	2.31
2 0	6 1	2	1.18	82 125	2	4.45	8 4	2	4.37
3 0	2 0	3	1.14	13 22	3	6.62	2 1	3	6.48
...									

Table 2. Smoothed values (interpolation per word) for the feature types *prev_N_wf*, *win_cont_lem_context* and *win_2gram_context* with the target word *art*.

to be described below). The other three possibilities consist on applying the Feature-Type method as in Figure 2, with two variants: use "Ng smoothing" for back-off (E), or in a combined fashion (F):

(A) No smoothing: Use raw frequencies directly.
(B) Fixed smoothing: Assign 0.1 raw frequency to each sense with a 0 value.
(Ng) Ng smoothing: This method is based on the global distribution of the senses in the training data. For each feature, each of the senses of the target word that has no occurrences in the training data gets the ratio between the probability of the sense occurring in the training data and the total number of examples: *Prob(sense)/Number_of_examples*.
(Ft) Feature-type smoothing: The method described in this paper. In the case of DL, note that when no data is available the feature is just discarded. For NB, it is necessary to rely in back-off strategies (see E and F).
(E) Ft with Ng as back-off: When Ft does not provide smoothed values, Ng is applied.
(F) Ft and Ng combined: The smoothed values are obtained by multiplying Ft and Ng values. Thus, in Figure 2, the $count'(f, w, sense)$ values are multiplied by *Prob(sense)/Number_of_examples*.

The output of the smoothing algorithm is the list of counts that replace the original frequency counts when computing the probabilities. We tested all possible combinations, but notice that not all smoothing techniques can be used with all the methods (e.g. we cannot use NB with "no smoothing").

4.3 Application of smoothing: an example

We will focus on three feature types and the target word *art* in order to show how the smoothed values are computed. For *art*, the following features have a 1/0 distribution in the training data: *"prev_N_wf Aboriginal"*, *"win_cont_lem_context Jerry"*, and *"win_2gram_context collection owned"*[4]. The majority sense for the three cases is the first sense. If we find one of those features in a test occurrence of *art*, we would like to know whether they are good indicators of the first sense or not.

As all these features occur with frequency 1/0, we have collected all counts for the feature types (e.g. *prev_N_wf*) which also have 1/0 occurrences in the training data.

[4] The first feature indicates that *Aboriginal* was the first noun to the left of *art*. The second that *Jerry* was found in the context window. The third that the bigram *collection owned* was found in the context window.

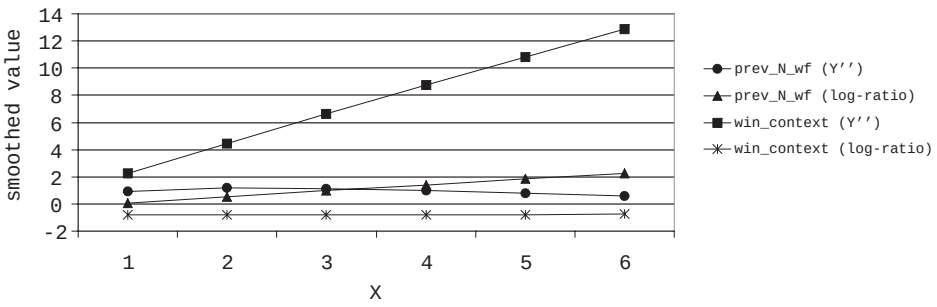

Fig. 3. Interpolation curves for the X/0 case (features *prev_N_wf* and *win_context*) with the target word *art*. The Y" estimation and the $log(X''/Y'')$ values are given for each X value and feature.

Table 1 shows the counts for *prev_N_wf*; the (4,4) values that appear for (X',Y') indicate that the *prev_N_wf* features that have 1/0 distribution in the target-folds contribute 4 examples to the majority sense and 4 to the minority senses when looked up in the estimation-folds.

The data for *prev_N_wf* has at least 3 points, and therefore we use the accumulated frequencies to obtain an interpolation table. We see that the interpolated frequencies for the minority senses stay nearly constant when the X values go up. This would reflect that the probability of the minority senses would go down quickly for higher values of X. In fact, the interpolated table can be used for values of X greater that 3, which had not been attested in the training data.

The same process is followed for the other two feature types: *win_cont_lem_context* and *win_2gram_context*. Table 2 shows the smoothed values (X',Y') and the interpolated values (X",Y") for the three types studied. The values for Y are much higher in the latter two cases, indicating that there is a very low confidence for these features for the word *art*. In contrast, *prev_N_wf* can be a valuable feature if found in 4/0 or greater distributions.

Figure 3 shows this different behavior graphically for *win_cont_lem_context* and *prev_N_wf*. For each feature type, the estimated Y" values and the log-ratio of the majority sense are given: the higher the Y" the lower the confidence in the majority sense, and inversely for the log-ratio. We can see that the curve for the Y" values assigned to *prev_N_wf* get lower credit as X increases, and the log-ratio grows constantly. On the contrary, for *win_cont_lem_context* the values of Y" increase, and that the log-ratio remains below zero, indicating that this feature type is not informative.

5 Results

The main experiment is aimed at studying the performance of four ML methods with the different smoothing approaches (where applicable). The recall achieved on the Senseval-2 dataset is shown in Table 3, the best results per method marked in bold. We separated the results according to the type of smoothing: basic smoothing ("no

	Basic Smoothing		Complex Smoothing			
	A	**B**	**Ng**	**Ft**	**E**	**F**
DL		60.4	60.7	**64.4**	**64.4**	64.3
NB		62.9	63.5		61.8	**63.8**
VSM	65.9	65.6	**66.2**	64.0	64.2	65.2
SVM	**65.8**					

Table 3. ML methods and smoothing techniques: (A) no smoothing, (B) fixed smoothing, (Ng) Ng smoothing, (Ft) Feature-type smoothing, the method presented in this paper, (E) Ft with Ng as back-off, and (F) the combination of Ft and Ng.

Systems	Basic smoothing	Complex smoothing
All methods	65.7	66.2
except SVM	64.9	66.2
except NB	**66.0**	**66.7**
except VSM	64.9	65.7
except DL	65.7	66.3

Table 4. Combination of systems with basic smoothing and complex smoothing. The rows show the recall achieved combining the 4 systems, and discarding one in turn.

smoothing" and "fixed smoothing"), and complex smoothing (techniques that rely on "Feature-type smoothing" and "Ng smoothing"). We can see that the results are different depending on the ML method, but the best results are achieved with complex smoothing for the 3 ML methods studied: DL (Ft and E), NB (F), and VSM (Ng). The best performance is attained by the VSM method, reaching 66.2%, which is one of the highest reported in this dataset. The other methods get more profit from the smoothing techniques, but their performance is clearly lower. McNemar's test [5] shows that the difference between the results of the best "basic smoothing" technique and the best "complex smoothing" technique is significant for DL and NB, but not for VSM.

All in all, we see that the performance of the statistically-based (DL, NB) methods improves significantly, making them comparable to the best single methods. In the next experiment, we tested a simple way to combine the output of the 4 systems: one system, one vote. The combination was tested on 2 types of systems: those that relied on "complex smoothing", and those that not. For each algorithm, the best smoothing technique for each type was chosen; e.g. the VSM algorithm would use the (A) approach for "simple smoothing", and (Ng) for "complex smoothing" (see Table 3). The performance of these systems is given in Table 4. The table also shows the results achieved discarding one system in turn.

The results show that we get an improvement over the best system (VSM) of 0.5% when combining it with DL and SVM. The table also illustrates that smoothing accounts for all the improvement, as the combination of methods with simple smoothing only reaches 66.0% in the best case, for 66.7% of the "complex smoothing" (difference statistically significant according to McNemar's test with 95% confidence interval).

[5] McNemar's significance test has been applied with a 95% confidence interval.

Method	Group	Smoothing	Recall	
Combination	IXA	Complex (best)	66.7	
Combination	**JHU**		**66.5**	⇒ **Best result to date**
VSM	IXA	Ng	66.2	
Combination	IXA	Basic (best)	66.0	
SVM	IXA		65.8	
SVM	**NUS**		**65.4**	⇒ **2nd best result to date**
DL	IXA	Ft	64.4	
Combination	**JHU-S2**		**64.2**	⇒ **Senseval-2 winner**
NB	IXA	E	63.8	
NB	NUS	"Add one"	62.7	

Table 5. Comparison with the best systems in the Senseval-2 competition and the recent literature.

As a reference, Table 5 shows the results reported for different groups and algorithms in the Senseval-2 competition and in more recent works. Our algorithms are identified by the "IXA" letters. "JHU - S2", corresponds to the Johns Hopkins University system in Senseval-2, which was the best performing system. "JHU" indicates the systems from the Johns Hopkins University implemented after Senseval-2 [10, 11]. Finally, "NUS" (National University of Singapore) stands for the systems presented in [12]. The Table is sorted by recall.

We can see that our systems achieve high performance, and that the combination of systems is able to beat the best results. However, we chose the best smoothing algorithm for the methods using the testing data (instead of using cross-validation on training, which would require to construct the smoothing tables for each fold). This fact makes the combined system not directly comparable. In any case, it seems clear that the system benefits from smoothing, and obtains results similar to the best figures reported to date.

6 Conclusions

In this work, we have studied the smoothing method proposed in [1], and we present a detailed algorithm for its application to WSD. We have described the parameters used, and we have applied the method on three different ML algorithms: Decision Lists (DL), Naive Bayes (NB), and Vector Space Model (VSM). We also analyzed the impact of several smoothing strategies. The results indicate that the smoothing method explored in this paper is able to make all three methods perform at very high precisions, comparable and in some cases superior to the best result attained in the Senseval-2 competition, which was a combination of several systems. We also show that a simple combination of the methods and a fourth system based on Support Vector Machines (SVM) attains the best result for the Senseval-2 competition reported so far (although only in its more successful configuration, as the system was not "frozen" using cross-validation). At present, this architecture has also been applied in the Senseval-3 competition, with good results, only 0.6% below the best system for English [13].

For the future, we would like to extend this work to X/Y features for Y greater than 1, and try other grouping criteria, e.g. taking into account the class of the word. We

would also like to compare our results to other more general smoothing techniques [14–16].

Finally, we would like to apply the smoothing results to detect good features for bootstrapping, even in the case of low amounts of training data (as it is the case for most of the words in WSD). The DL method, which improves significantly with smoothing, may be well suited for this task, as it relies on one single piece of evidence (feature) to choose the correct sense.

References

1. Yarowsky D.: Three Machine Learning Algorithms for Lexical Ambiguity Resolution. PhD thesis, Department of Computer and Information Sciences. University of Pennsylvania (1995)
2. Miller G.A., Leacock C., Tengi R., Bunker R.: A Semantic Concordance. Proceedings of the ARPA Human Language Technology Workshop. Distributed as *Human Language Technology* by San Mateo, CA: Morgan Kaufmann Publishers. Princeton, NJ (1993) 303–308
3. Edmonds P., Cotton S.: SENSEVAL-2: Overview. Proceedings of the Second International Workshop on evaluating Word Sense Disambiguation Systems. Toulouse, France (2001)
4. Yarowsky D.: Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics. Cambridge, MA (1995) 189–196
5. Ngai G., Florian R.: Transformation-Based Learning in the Fast Lane. Proceedings of the Second Conference of the North American Chapter of the Association for Computational Linguistics. Pittsburgh, PA, USA (2001) 40–47
6. Pedersen T.: A Decision Tree of Bigrams is an Accurate Predictor of Word Sense. Proceedings of the Second Conference of the North American Chapter of the Association for Computational Linguistics. Pittsburgh, PA, USA (2001)
7. Joachims T.: Making Large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning. MIT Press. Cambridge, MA (1999) 169–184
8. Neter J., Wasserman W., Kutner M. H.: Applied Linear Statistical Models. Irwin. Homewood, Illinois (1985)
9. Ng H.T.: Exemplar-Based Word Sense Disambiguation: Some Recent Improvements. Proceedings of the Second Conference on Empirical Methods in Natural Language Processing Somerset, New Jersey (1997) 208–213
10. Cucerzan S., Yarowsky D.: Minimally Supervised Induction of Grammatical Gender. Proceedings of HLT/NAACL 2003. Edmonton, Canada (2003)
11. Florian R., Cucerzan S., Schafer C., Yarowsky D.: Combining Classifiers for Word Sense Disambiguation. Journal of Natural Language Engineering. Cambridge University Press (2002)
12. Lee Y., Ng H.T.: An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-2002). Philadelphia, PA, USA (2002) 41–48.
13. Agirre A., Martinez D.: The Basque Country University system: English and Basque tasks. Proceedings on the 3rd ACL workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL). Barcelona, Spain (2004)
14. Good I.J.: The Population Frequencies of Species and the Estimation of Population Parameters. Biometrika, Vol. 40. (1953) 237–264
15. Jelinek F., Mercer R.: Interpolated estimation of Markov source parameters from sparse data. Pattern Recognition in Practice. Amsterdam : North Holland Publishing Co. (1980) 381–397
16. Chen S.F.: Building Probabilistic Models for Natural Language. Ph.D. Thesis, Harvard University (1996)