Exploring feature set combinations for WSD

Eneko Agirre e.agirre@ehu.es
Oier Lopez de Lacalle jibloleo@si.ehu.es
David Martínez davidm@si.ehu.es

IXA NLP Group
Univ. of the Basque Country
Donostia, 20018

Resumen: Este trabajo explora la división de atributos en grupos para poder mejorar la desambiguación de acepciones (WSD) mediante la combinación de sistemas entrenados en cada uno de estos grupos de atributos. Los resultados conseguidos demuestran que sólo k-NN es capaz de obtener beneficio de la combinación de la división de atributos, y que el voto único no es suficiente para la mejora. Por ello proponemos combinar todo los subsistemas k-NN donde cada vecino da su voto según su rango de vecindad. Para la evaluación hemos utilizado dos conjuntos de datos (Senseval-3 Lexical-Sample y All-words), fijando las mejores opciones de combinación en un tercer conjunto de datos (Senseval-2 Lexical-Sample). Los resultados para la tarea *All-words* de Senseval-3 son los mejores que se han publicado hasta el día de hoy. Los resultados del Lexical-Sample se situan entre los mejores en el estado-del-arte.
Palabras clave: Desambiguación de acepciones de palabra, espacio de atributos, k Nearest Neighbor

Abstract: This paper explores the split of features sets in order to obtain better WSD systems through combinations of classifiers learned over each of the split feature sets. Our results show that only k-NN is able to profit from the combination of split features, and that simple voting is not enough for that. Instead we propose combining all k-NN subsystems where each of the k neighbors casts one vote. We have performed a thorough evaluation on two datasets (Senseval-3 Lexical-Sample and All-words), having set the best combination options in a development dataset (Senseval-2 Lexical-Sample). The results for the All-Words task are the best published up to date. The results for the lexical sample are state-of-the-art.
Keywords: Word Sense Disanbiguation, feature space, combination, k Nearest Neighbors

1 *Introduction*

Many current Natural Language Processing (NLP) systems rely on linguistic knowledge acquired from tagged text via Machine Learning (ML) methods. The problem is modeled using a possibly high number of features, statistical or alternative models are learned on top of the features, and the learned models are then applied to running text.

In Word Sense Disambiguation (WSD) the features used to model the context of the word senses are heterogeneous, ranging from part-of-speech sequences to bag-of-words. The last Senseval exercises show that the more feature types one throws into the algorithm, the better are the results (Agirre and Martínez, 2004). Still, it is not clear which is the best way to profit from those rich feature space, as a large feature spaces tend to have highly redundant and heterogeneous features (see Section 2.1).

One alternative to better model the feature space is to split the features, and thus allow the learning algorithm to better capture the patterns in the data. Obtaining more coherent feature spaces we could in principle avoid the noise created by the redundant information. Learning different systems also leads naturally to combine those systems. One system would do better for some instances than the others, and the combination would produce a robust system.

This possibility already yield the best results reported to date on the Senseval-3 lexical sample dataset for English (Agirre, Lopez de Lacalle, and Martínez, 2005), where splitting the feature space was combined with the inference of latent features. The system used k-Nearest Neighbor (k-NN) trained on different feature spaces, some of them created with SVD, which combined up to 20 different k-NN systems. The results were outstanding, but the use of SVD and the large number of combined systems made it rather complex and slow.

In this work, we wanted to test whether a simpler combination of ML systems trained on split feature spaces would attain such a good performance as the previously cited one. For that we tested three algorithms (k-NN, a Vector Space Model using centroids and a Support Vector Machine) on both lexical sample and all-word datasets.

The paper is structured as follows. Section 2 reviews the feature set, the different splits of the feature set, and the learning methods. Section 3 introduces the combination methods. Section 4 presents the experimental setting and results of the experiments. Section 5 discusses the results and related work. Finally, Section 6 draws the conclusions and the future work.

2 *Features and learning methods used*

In this section we introduce the features used, the different feature sets we defined, and the three learning methods.

2.1 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.
Global features: we extract the lemmas of the content words in the whole context, and in a ±4-word window around the target. We also obtain salient bigrams in the context, with the methods and the software described in (Pedersen, 2001).

[1]The PoS tagging was performed with the fnTBL toolkit (Ngai and Florian, 2001).

[2]This software was kindly provided by David Yarowsky's group, from the Johns Hopkins University.

2.2 Splitting the feature sets

There are many ways to split the feature space. Following our criterion we tried to group regarding the kind of the feature type, trying build a coherent set of features. In total, we tried with 6 feature spaces. Initially, we have the set with all the features, being the richest and heteregeneous feature space. Another way for distributing them is to separate bag-of-words feature and rest features (Local collocations, salient bag-of-bigrams, see 2.1, and syntactic dependencies). Finally, we try with the local collocations (introduced 2.1. section), syntactic dependencies (2.1. section) and the bag-of-bigrams obtained from the context as described in (Pedersen, 2001). The figure 1 summarizes the feature sets used.

- **bow**: `bag-of-words (open-class lemmas).`
- **local**: `local collocations.`
- **sx**: `syntactic dependencies.`
- **bob**: `bag-of-bigrams.`
- **notbow**: `all features except` **bow**.
- **ehu**: `all features.`

Figure 1: Summary of feature sets.

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. Each occurrence or instance is represented by the features found in the context (f_i).

$$weight(s_k) = cos(\vec{C}_{s_k}, \vec{f}) = \frac{\vec{C}_{s_k} \cdot \vec{f}}{|\vec{C}_{s_k}||\vec{f}|} \quad (1)$$

The k **Nearest Neighbor** (k-NN) is a memory based learning method, where the neighbors are the k most similar contexts, represented by feature vectors ($\vec{c_i}$), of the test vector ($\vec{f}$). The similarity among instances is measured by the cosine of their vectors (as in eq. (1)). The test instance is labeled with the sense obtaining the maximum sum of the weighted votes of the k most similar contexts.

The vote is weighted depending on its (neighbor) position in the ordered rank, with the closest being first. Eq. (2) formalizes k-NN, where C_i corresponds to the sense label of the i-th closest neighbor.

$$\arg\max_{S_j} = \sum_{i=1}^{k} \begin{cases} \frac{1}{i} & \text{if } C_i = S_j \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

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 ($\vec{C_{s_k}}$). These centroids are compared with the vectors that represent testing examples ($\vec{f}$), by means of the cosine similarity function (eq. (1)). The closest centroid assigns its sense to the testing example.

Regarding **Support Vector Machines** (SVM) we utilized SVM-Light, a public distribution of SVM by (Joachims, 1999). 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 *Combination*

We first present a simple way to combine classifiers using single-voting, which can be used with all ML methods. We then present a special method to combine k-NN systems.

3.1 Single-voting

There are many ways to combine different ML methods. The simplest way is to associate one vote to each of the systems, and count the total number of votes for each class. In our experiments, we assign the same weight to each ML method, regardless of its prior performance. The main advantages of this approach are its simplicity, and the possibility of integrating new systems easily.

We have built different classifiers from the feature splits explained before (cf. 2.2). After obtaining the single classifiers, we have combined them using the single-voting approach. This specific combination will be studied in section 4.1.

3.2 kNN combination

Taking advantage of the characteristics from k-NN method, we exploited the fact that a classifier can be seen as k points casting each one vote, making easy a combination of several k-NN classifier.

For instance, if we have two k-NN classifiers of $k = 5$, c_1 and c_2, then we can combine them into a single classifier equivalent to $k = 10$. The difference with a single k-NN classifier is that the votes are coming from a different feature space, that is, each vote discriminate in another way to the rest. The combined classifier is from a different mappings of a feature spaces before tag the target instance.

In order to carry through the properties of each feature space, we first try weighing each vote by the cosine similarity. We then realized that each feature space has its own scale to measure similarities, and combining in this way we were introducing some kind of bias. We attach more importance to some classifier, distorting the combination. In order to solve the bias problem, we weight each vote by the inverse ratio of its position in the rank[3]. The rank weighting method alleviates bias problem and maintains the importance of some neighbors regarding to others less similar.

We tried with the different classifiers previously built from splitted feature space (explained in 2.2). More specifically, the results of combinations will be explained in section 4.1 where we do the optimization.

4 *Experimental setting and results*

In order to organize the experiments, we used corpora from different editions of senseval[4]. For development and tuning of parameters we use the Senseval-2 Lexical-Sample corpora. For testing we use both Senseval-3 Lexical-Sample and Senseval-3 All-Words datasets. In the latter, we take Semcor as the training corpus. We will present the the setting and results for each of the datasets in turn.

4.1 Senseval-2 Lexical-Sample

The optimization experiments have been performed using the Senseval-2 Lexical-Sample (Kilgarriff, 2001) data. The source corpora was a sample of BNC (Leech, 1992) (mostly) and the WSJ. The chosen sense inventory was a previous version of WordNet 1.7. The corpus consist on 73 words

[3] $(k - r_i + 1)/k$

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

(noun, verbs, adjectives), with 4,328 testing instances, and 8,611 instances for training.

For k-NN methods we saw, in previous experiments, that best k was 5. All the experiments were carried with a fixed k in 5. We have done an exploration for the best combination. The VSM does not need any optimization, it has not any parameter, and finally, we have used SVM in a default mode, without any parameter optimization.

First, we will show the results for single classifiers in the different feature-spaces, and finally, we will describe the results for best combination.

4.1.1 Senseval-2 Lexical Sample single classifiers

Table 1 shows the results, recall and precision, for each method trained and tested in several feature spaces (the extension of the classifier denote the feature space). The results show that the more feature types one throws into the algorithm, the better are the results as shown in (Agirre and Martínez, 2004).

We can see richest feature space, *ehu*, is the best single classifier in every method, except for SVM which attains the same results for *ehu* and *notbow* feature distributions.

The VSM from the *ehu* feature space is the best method. It obtains 63.3 of recall in the Senseval-2 Lexical Sample.

Taking into account each of the feature sets, we note that generally the local collocations features (*local*) discriminate better than bag-of-words features (*bow*). Only in the case of VSM the *bow* features work better than *local* ones. The reason that syntactic dependencies (*sx*) and bag-of-bigrams (*bob*) do not work as well as local collocations and bag-of-words features might be the insufficient amount of feature instances that occur in the given context. But we see that they help when we throw into a more complex feature space (*notbow*).

4.1.2 Senseval-2 Lexical-Sample combination optimization

In this section we report the results for the combination of the single systems in the previous section. As our focus is on the use of feature spaces, we tried all exhaustive combinations of feature sets, but always using a single learning method.

Table 2 shows the best combinations per learning method. For k-NN, *ehu+bow+notbow* yields the best results, both for single-voting and for the k-NN combination. The same combination yields best results for SVM, but *ehu+bow+notbow+bob* is the best in the case of VSM. k-NN is the only one improving over the single methods, and attains the best results overall.

Table 2 also shows that the finer grained feature sets are not the best, and that using all features helps obtain better results.

classifier	**cov** (%)	**rec** (%)
knn.ehu	100	62.4
knn.notbow	100	60.4
knn.local	100	58.5
knn.bow	100	52.4
knn.sx	100	49.1
knn.bob	39.0	23.8 (60.8)
svm.ehu	100	61.0
svm.notbow	100	61.0
svm.local	100	60.9
svm.sx	100	56.1
svm.bow	100	55.8
svm.bob	100	51.9
vsm.ehu	100	63.3
vsm.notbow	100	57.2
vsm.bow	100	55.2
vsm.local	100	51.6
vsm.sx	100	41.6
vsm.bob	39.0	22.1 (56.6)

Table 1: Results for single classifiers in the Senseval-2 Lexical-Sample. Parenthesis for precision.

classifier	**rec**
knn.ehu+bow+notbow (knn-comb)	**63.5** ↑
vsm.ehu+bow+notbow+bob	63.0 ↓
knn.ehu+bow+notbow (single-vot)	62.2 ↓
svm.ehu+bow+notbow	61.0 =

Table 2: Results for best classifier combinations in the Senseval-2 Lexical-Sample. ↑ means *ehu* single system is worse, = means *ehu* system is equal, and ↓ is ehu system better. Coverage is 100% for all.

4.2 Senseval-3 Lexical Sample

These experiments have been performed using the Senseval-3 English Lexical-Sample data (Mihalcea, Chklovski, and Killgarriff, 2004). The source corpora was the BNC (Leech, 1992). WordNet 1.7.1. (Fellbaum, 1998) was chosen as the sense inventory for nouns and adjectives, while the verb senses came from the *Wordsmyth* dictionary[5]. 57 words (nouns, verbs, and adjectives) were tagged, with 7,860 instances for training and 3,944 for testing.

In this experiment, we have only tried the best parameter settings and the best combinations in Senseval-2 Lexical Sample.

4.2.1 Senseval-3 Lexical Sample single and combined classifiers

Table 3 shows the results of the single and combined classifiers. As expected from the results in the previous section, the combinations for VSM and SVM do not improve over the single results, but do improve over the k-NN system. The best results are for VSM with the combined k-NN following closely.

Compared to more complex system described in (Agirre, Lopez de Lacalle, and Martínez, 2005) we are still 2 points below.

classifier	rec
vsm.ehu	**71.5**
knn.ehu	70.4
svm.ehu	69.2
knn.ehu+bow+notbow (knn-comb)	**71.4**
vsm.ehu+bow+notbow+bob	70.9
knn.ehu+bow+notbow (single-vot)	70.6
svm.ehu+bow+notbow	68.9
Best S3[6]	72.9
Best published[7]	**73.4**

Table 3: Results for best single and combined classifier in the Senseval-3 Lexical-Sample.Coverage is 100% for all.

4.3 Senseval 3 All-Words Task

The test data for this task consisted on 5,000 words of text (Snyder and Palmer, 2004). The data was extracted from two WSJ articles and one excerpt from the BC. The texts represents three different domains: editorial, news story, and fiction. Overall, 2,212 words were tagged with WordNet 1.7.1. senses (2,081 if we do not include multi-words).

We use Semcor (Miller et al., 1993) corpus for training. Semcor consist on subset of the Brown Corpus (BC) plus the novel The Red Badge of the Courage. It contains a number of texts comprising about 200,000 words where all content words have been manually tagged with senses from WordNet 1.6. We have use the mapping from WordNet 1.6 to 1.7.1 (Daude, Padro, and Rigau, 2000). In the case where target word has less than 10 instances in SemCor we have applied the most frequent sense.

We have prepared a clean test set to make our systems results comparable with the official results from Senseval-3 all words task. Taking into account that systems from Senseval-3 did not know the part-of-speech (PoS) of the target word, we have removed test instances where the PoS was wrongly assigned by the two best systems in the competition. We have also removed multiwords. After cleaning the test set comprises 1.819 instances.

As in the previous section, we have used the parameterization from Senseval-2 Lexical-Sample (cf. 4.1.2).

classifier	rec
knn.ehu+bow+notbow (knn-comb)	**68.5**
svm.ehu	67.9
knn.ehu	67.8
knn.ehu+bow+notbow (single-vot)	67.8
GAMBL (best S3AW)	67.8
vsm.ehu	65.5

Table 4: Results for different systems in the Senseval-3 All-Words task. Coverage is 100% for all.

Table 4 shows that the k-NN combination outperforms all the other systems, including the best system in Senseval-3 All-Words task. This confirms that a richer feature-space with an appropriate combination is able to yield good results.

5 *Results and related work*

We have shown that a simple method like a k-NN classifier can work as good as more complex, and a priori more powerful methods. Splitting the feature-space and then combining them into a single classifier obtains the best results up to date in the Senseval-3 All-Words task.

The good results are due to the potential for k-NN classifiers to be combined. Rather than using "the one classifier one vote" paradigm, each classifier suggest the k closest instances (and their word senses) from their feature space and after that, the merged classifier sums them to decide the word sense.

We think that the reason of the good performance of the combination is that each of

[5]http://www.wordsmyth.net/
[6](Decadt et al., 2004)
[7](Agirre, Lopez de Lacalle, and Martínez, 2005)

the changes in the feature space helps finding regularities in the data, which k-NN (single classifier) could not find. When we combine each of the simpler k-NN systems, we are looking for the word sense that is closer to the target instance in different feature spaces. In other words, we are discriminating word senses in different feature spaces.

The results for Senseval-3 Lexical-Sample, through state-of-the-art, show that our method could be improved incorporating the more complex setting of (Agirre, Lopez de Lacalle, and Martínez, 2005), see below.

The combination of classifiers is not a new area, but it is still active. There are many works exploring the different ways to combine classifiers, but very little deal with combining k-NN classifiers. We cite some works that confirm our findings. Related to WSD task, for instance, the JHU-English system (Yarowsky et al., 2001; Florian et al., 2002), which used a voting scheme, obtained the best performance at English lexical sample task in Senseval-2. The main conclusions of their study are that the feature space has significally greater impact than the algorithm choice, and that the different algorithms help to construct significantly a more robust WSD system.

In (Agirre, Lopez de Lacalle, and Martínez, 2005), a previous work, we not only split the original feature set, but also created a low dimensional space using Singular Value Decomposition (SVD), which allowed for better results at the cost of more computation.

(Kohomban and Lee, 2005) show in a different WSD task that building separate k-NN classifiers from different subsets of features and combining them works better than constructing a single classifier with the entire feature set.

In (Gliozzo, Giuliano, and Strapparava, 2005), instead of splitting the feature space and then combining the classifiers, they use specialized kernels to model the similarity for each kind of features, but they also incorporate the use of SVD.

Another approach is the combination to disambiguate all the words in the context, as in (Stevenson and Wilks, 2001). In this work, they integrate the answers of three partial taggers based on different knowledge sources in a feature-vector representation for each sense. The vector is completed with information about the sense, and simple collocations extracted from the context. A memory based learning algorithm is then applied to classify new examples.

Not related with the WSD task, (Bay, 1999) describes MFS (Multiple Feature Subset), a combination of NN (nearest neighbor) algorithms that classifies using simple voting from NN classifiers, each having access only to a random subset of feature. As we do, the author built less accurate single classifiers (each classifier making independent errors) and joined them into a unique classifier. The author tried to build less correlated feature subsets in order to make independent errors. The experiments were done in several datasets from the UCI Repository. Related to this, (Dietterich, 1997) claims that spliting features only works when the feature space is higly redundant.

In order to compare our method of splitting the feature space with the proposal from (Bay, 1999), we performed some additional experiments on the Senseval-3 Lexical-Sample dataset. Table 5 shows the results of applying a random split in three feature sets, both using single voting over 1NN and our method for 5NN. We also tried using 1NN instead of 5NN, but the results are lower than combined system (last row).

classifier	**rec**
Random, 3 splits 1NN	65.7
Random, 3 splits 5NN	70.0
knn.ehu+bow+notbow 1NN	67.1
knn.ehu+bow+notbow 5NN	71.4

Table 5: Results for additional experiments on Senseval-3 Lexical-Sample.

6 *Conclusions and Future Work*

This paper explores the split of features sets in order to obtain better through combinations of classifiers learned over each of the split feature sets. Our results show that k-NN is able to profit from the combination of split features (contrary to VSMand SVM), and that simple voting is not enough for that. Instead we propose combining all k-NN subsystems where each of the k neighbors casts one vote.

The experiments explore different feature spaces by splitting a rich set of features into a smaller and less accurate, but more coherent sets. The comparisons with random splits suggest that a manual split based on the nature of the features is more productive than random splits.

We have performed a thorough evaluation on two datasets (Senseval-3 Lexical-Sample and All-words), having set the best combination options in a development dataset (Senseval-2 Lexical-Sample). The results for the All-Words task are the best published up to date. The results for the lexical sample are state-of-the-art. In (Agirre, Lopez de Lacalle, and Martínez, 2005) we show that the addition of SVD to the combination into a more complex setting allows to obtain the best results to date on that dataset. Our simplification has the advantage of not needing the computationally demanding algebraic operations of SVD, which allows it to be run on the all-words dataset.

For the future we plan to pursue the use of SVD, and explore ways to make it efficient into an all-words setting.

7 Acknowledgements

Oier Lopez de Lacalle has a PhD grant from the Basque Government.

References

Agirre, E. and D. Martínez. 2004. Smoothing and Word Sense Disambiguation. In *Proceedings of EsTAL - España for Natural Language Processing*, Alicante, Spain.

Agirre, Eneko, Oier Lopez de Lacalle, and David Martínez. 2005. Exploring feature spaces with svd and unlabeled data for Word Sense Disambiguation. In *Proceedings of the Conference on Recent Advances on Natural Language Processing (RANLP'05)*, Borovets, Bulgary.

Bay, Stephen D. 1999. Nearest neighbor classification from multiple feature subsets. *Intell. Data Anal.*, 3(3):191–209.

Daude, J., L. Padro, and G. Rigau. 2000. Mapping wordnets using structural information. Hong Kong, PRC.

Decadt, Bart, Véronique Hoste, Walter Daelemans, and Antal van den Bosch. 2004. Gambl, genetic algorithm optimization of memory-based wsd. In R. Mihalcea and P. Edmonds, editors, *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, pages 108–112, Barcelona, Spain, July. ACL.

Dietterich, Thomas G. 1997. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database.* MIT Press.

Florian, Radu, Silviu Cucerzan, Charles Schafer, and David Yarowsky. 2002. Combining classifiers for word sense disambiguation. *Natural Language Engineering*, 4(8):327–341.

Gliozzo, Alfio Massimiliano, Claudio Giuliano, and Carlo Strapparava. 2005. Domain Kernels for Word Sense Disambiguation. *43nd Annual Meeting of the Association for Computational Linguistics. (ACL-05).*

Joachims, T. 1999. Making Large–Scale SVM Learning Practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA. MIT Press.

Kilgarriff, A. 2001. English Lexical Sample Task Description. In *Proceedings of the Second International Workshop on evaluating Word Sense Disambiguation Systems*, Toulouse, France.

Kohomban, Upali S. and Wee S. Lee. 2005. Learning Semantic Classes for Word Sense Disambiguation. In *43nd Annual Meeting of the Association for Computational Linguistics. (ACL-05)*, University of Michigan, Ann Arbor.

Leech, G. 1992. 100 million words of English: the British National Corpus. *Languaje Research*, 28(1):1–13.

Mihalcea, R., T. Chklovski, and Adam Killgariff. 2004. The Senseval-3 English lexical sample task. In *Proceedings of the 3rd ACL workshop on the Evaluation of Systems for the Semantic Analysis of Text (SENSEVAL)*, Barcelona, Spain.

Miller, G.A., C. Leacock, R. Tengi, and R.Bunker. 1993. A Semantic Concordance. In *Proceedings of the ARPA Human Language Technology Workshop. Distributed as* Human Language Technology *by San Mateo, CA: Morgan Kaufmann Publishers.*, pages 303–308, Princeton, NJ.