Inteligencia Artificial, 21(61), 67-81
doi: 10.4114/intartif.vol21iss61pp67-81

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

# Exploring the impact of word embeddings for disjoint semisupervised Spanish verb sense disambiguation

Cristian Cardellino, Laura Alonso Alemany

Facultad de Matemática, Astronomía, Física y Computación.
Universidad Nacional de Córdoba, Argentina.
ccardellino@unc.edu.ar

Facultad de Matemática, Astronomía, Física y Computación.
Universidad Nacional de Córdoba, Argentina.
alemany@famaf.unc.edu.ar

**Abstract** This work explores the use of word embeddings as features for Spanish verb sense disambiguation (VSD). This type of learning technique is named *disjoint semisupervised learning* [21]: an unsupervised algorithm (i.e. the word embeddings) is trained on unlabeled data separately as a first step, and then its results are used by a supervised classifier.

In this work we primarily focus on two aspects of VSD trained with unsupervised word representations. First, we show how the domain where the word embeddings are trained affects the performance of the supervised task. A specific domain can improve the results if this domain is shared with the domain of the supervised task, even if the word embeddings are trained with smaller corpora. Second, we show that the use of word embeddings can help the model generalize when compared to not using word embeddings. This means embeddings help by decreasing the model tendency to overfit.

**Keywords**: Natural Language Processing, Word Embeddings, Word Sense Disambiguation.

## 1 Introduction

SenSem [1] is one of the few Spanish resources available with examples of disambiguated verbs. As most of the manual resources, it was expensive to create, thus the labelled data is small. On the other hand, the distribution of the labels (senses) is Zipfian [24], thus it suffers from a high imbalance in the dataset, where few classes have most of the occurrences and the others have practically none.

To obtain a machine learning algorithm for verb sense disambiguation the simplest approach is to train a supervised automatic classifier. The traditional approach to automatic word sense disambiguation with machine learning models uses supervised "handcrafted features". The features are taken from the training data itself, specially using bag-of-words-like features [8]. We refer to this as a "purely supervised approach".

However with a purely supervised approach and as consequence of the challenges stated before, we have to deal with a major problem: overfitting of the data. As handcrafted features are so tightly related to the training data, it makes the representation fixed on the domain of the data. When dealing with new examples, if the features (e.g. bags-of-words, ngrams, etc.) were not on the original training data, the examples may have little information to be represented with the available features of the model.

**ISSN: 1137-3601 (print), 1988-3064 (on-line)**

This work explores the use of word embeddings to aid the task of Spanish verb sense disambiguation. It reports the performance of supervised learning algorithms when given word embeddings as features. The latter is also known as *disjoint semisupervised learning* [21].

Word embeddings can be obtained from unlabelled corpora. Hence they are known as unsupervised representations. This is why the use of word embeddings as features of a supervised classifier can be considered a semisupervised method. Since the corpus where word embeddings are obtained from is different from the annotated corpus given to the classifier, this kind of learning is known as disjoint semisupervised learning.

We explore the use of two different kinds of word embeddings: ones pre-trained from a general domain corpus, the Spanish Billion Word Corpus and Embeddings [4], and others trained from a specific domain corpus belonging to the journalistic domain, the same as the SenSem corpus.

We show the use of specific domain corpora to train the word embeddings has an impact on the performance of the supervised classifier, improving the results for Spanish verb sense disambiguation.

On the other hand we compare the performance of word embeddings against the purely supervised approach described above. We show that although word embeddings may produce some loss in performance of the classifier they decrease the model's tendency to overfit.

This paper is structured in the following way: Section 2 does a general review of the previous work for English and Spanish word and verb sense disambiguation. It also reviews on different unsupervised word representations and mentions the work done in word sense disambiguation using word embeddings. Section 3 describes the resources we worked with in this paper. Section 4 describes the experimental setting of this work and lists the features for the supervised approach and the way the unsupervised representations are used to represent instances. It also describes the metrics we used in the analysis of results to measure the performance and the tendency of an algorithm to overfit. Section 5 shows the results obtained from the experimentation and does a general analysis on what we find out. Finally, Section 6 finalizes the work with general remarks of the results, the conclusions regarding them, and establish the future work to be done.

## 2 Previous Work

For general word sense disambiguation (WSD) the state of the art at the time of writing this work is *It Makes Sense* (IMS) [23]. IMS presents a flexible framework which allows users to integrate different preprocessing tools, additional features, and different classifiers. In the original work, the authors using a simple implementation, with a linear support vector machines classifier with multiple knowledge-based features, achieved state-of-the-art results on several SensEval and SemEval tasks. IMS provides an extensible and flexible platform for researchers interested in using a WSD component. Users can choose different tools to perform preprocessing, such as trying out various features in the feature extraction step, and applying different machine learning methods or toolkits in the classification step. IMS consists of three independent modules: preprocessing, feature and instance extraction, and classification. Knowledge sources are generated from input texts in the preprocessing step. With these knowledge sources, instances together with their features are extracted in the instance and feature extraction step. Then IMS trains one classification model for each word type. The model is used to classify test instances of the corresponding word type.

McCarthy and Carroll [13] worked on disambiguation of nouns, verbs and adjectives using selectional preferences acquired from automatically preprocessed and parsed text. The selectional preferences are acquired for grammatical relations (subject, direct objects, and adjective-noun) involving nouns and grammatically related adjectives or verbs. They use Wordnet synsets to define the sense inventory. Their method exploits hyponym links given for nouns (e.g. *cheese* is an hyponym of *food*), troponym links for verbs (e.g., *limp* is a troponym of *walk*), and the "similar-to" relationship given for adjectives (e.g., one sense of *cheap* is similar to *flimsy*). From the paper, it is not clear whether selectional preferences impact positevely in verb sense disambiguation (VSD).

Ye and Baldwin [22], use Selectional Preferences extracted with a Semantic Role Labeler for VSD. Their VSD framework is based upon three components: extraction of disambiguating features, selection of the best disambiguating feature with respect to unknown data and the tuning of the machine learner's parameters. For their study they use a Maximum Entropy algorithm [2]. The VSD features they used include selectional preferences and syntactic features, e.g, bag of words, bag of PoS tags, bag of chunks; parsed tree based features using different levels of the tree as source of information; and non-parse trees based syntactic features, e.g., voice of the verb, quotatives, etc. They show improved performance of their system when selectional preferences are taken into account.

Another work on English VSD is the one by Chen and Palmer [5], presenting a high-performance broad-coverage supervised word sense disambiguation system for English verbs that uses linguistically motivated features and a smoothed maximum entropy machine learning model. Kawahara and Palmer [9] presented a supervised method for verb sense disambiguation based on VerbNet. Contrary to the most common VSD methods, which create a classifier for each verb that reaches a frequency threshold, they created a single classifier to be applied to rare or unseen verbs in a new text. Their classifier also exploits generalized semantic features of a verb and its modifiers in order to better deal with rare or unseen verbs.

Many of the features that are used for English VSD are not available for Spanish VSD because the preprocessing tools and annotated corpora are less developed.

In the SemEval 2007 task for multilevel semantic annotation of Catalan and Spanish [12], Màrquez et al. [11] primarly focused on Noun Sense Disambiguation. They used a three way approach: if the word has more than a threshold number of occurrences, it is classified with a SVM classifier; if the word has less occurrences than the threshold it is assigned the most frequent sense (MFS) in the training corpus; if the word is not present in the training corpus then it is assigned the MFS in WordNet. The SVM classifier features were a bag of words, n-grams of part-of-speech tags and lemmas, and syntactic label and syntactic function of the constituent that has the target noun as head.

Other work in WSD with applications in Spanish is the work of Montoyo et al. [17] where the task of WSD consists in assigning the correct sense to words using an electronic dictionary as the source of word definitions. They present a knowledge-based method and a corpus-based method. In the knowledge-based method the underlying hypothesis is that the higher the similarity between two words, the larger the amount of information shared by two of their concepts. The corpus-based method is based on conditional maximum-entropy models, it was implemented using a supervised learning method that consists of building word-sense classifiers using a semantically annotated corpus. Among the features for the classifier they used word forms, words in a window, part-of-speech tags and grammatical dependencies.

It is noticeable that the features for Spanish WSD are more shallow than the features available for English WSD. In this work we will explore more combinations of features aimed specifically to Spanish VSD.

When it comes to using word embeddings, Turian et al. [19] improve the accuracy of different existing NLP systems by using unsupervised word representations as extra features. In their work, they evaluate three different unsupervised word representations: Brown clusters [3], Collobert and Weston [21] embedding, and HLBL embeddings of words [16]; and try them out on named entity recognition and chunking. Using these representations they effectively show improvement of performance in nearly state-of-the-art baselines. More recent years have seen the introduction to the *skip-gram model* and *continuous bag-of-words model* by Mikolov et al. [14]. These are novel model architectures for computing continuous vector representations of words from very large data sets. In particular, the skip-gram model is able to learn high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. The use of negative sampling [15] improve both the quality of the vectors and the training speed, by sub-sampling of the frequent words.

For WSD with word embeddings there is an evaluation study by Iacobacci et al. [7]. In this, they propose different methods through which word embeddings can be leveraged in a state-of-the-art supervised WSD system architecture, and perform analysis of how different parameters affect performance.

## 3 Resources

### 3.1 Labeled Corpus

SenSem [1] is a manually disambiguated corpus for verbs in both Spanish and Catalan. It contains the 248 most common verbs of Spanish, annotated with senses defined in a provided lexicon, some of them with mappings to the Spanish WordNet Ontology [6].

A version of the SenSem corpus has part-of-speech tags automatically annotated with Freeling [18]. However these tags are annotated on a word based level, thus there is a large proportion of them annotated with the wrong tag (e.g. verbs annotated as nouns). Furthermore, Spanish has some words that are multi-words (i.e. words formed of two ore more different terms) which tag is not the same than those of each of the words compounding the multi-word. E.g. “más_allá_de” is tagged as a multi-word with Part-of-Speech tag “SP”, it is a preposition, however, the words “más” and “allá’ are by themselves adverbs and only “de” is a preposition.

In order to gather information more useful for feature extraction, there were two preprocessing steps of the SenSem corpus. First, an automatic annotation using a statistical dependency parser. In this step the SenSem’s sentences, which are tokenized, are parsed with Freeling’s statistical dependency parser. The sentences are automatically annotated with: lemma, part-of-speech tag, morphosyntactic information and dependency triples. Also, there is multi-word detection and named entity recognition (treated by Freeling as multi-words).

Nevertheless, the automatic annotation is not enough as errors come not only from Freeling but other problems SenSem has as well: sentences without a defined sense, sentences where the verb to disambiguate is not present and sentences truncated before finishing. For this reason, the second step of preprocessing was manual, where each of the automatically annotated sentences, where the main lemma to disambiguate was lost (because of mistagging, not being correctly marked in the original resource, etc.), is found manually. Besides this, all cases that are erroneous in the original corpus (e.g. truncated sentences or sentences without a defined sense) were discarded.

After the preprocessing step, the SenSem corpus was split in train/test. For this all those senses with only one occurrence in the corpus are filtered out and the remaining senses are split with stratified sampling using 80% for training and 20% for testing, where the training and the test corpus have each at least one occurrence of every sense and at most a percentage of samples of each sense similar to the complete set. This was done in order to have feedback in the test set regarding those classes appearing the least number of times, which is different to the circumstances in which this kind of systems work in real environments; thus, the experimental results which follow should be understood as the best we are able to obtain in some of the most favorable conditions. Table 1 shows some of the statistics of the SenSem corpus after the preprocessing of the text and removal of erroneous sentences. We want to focus specially on the average number of instances for the most frequent sense per lemma in comparison to the average number of instances for the second most frequent sense per lemma. It is clear to see the imbalance of the senses in the corpora as the most frequent class has more than 3 times more occurrences than the next.

In summary, the original version of the SenSem was automatically parsed with Freeling to gather more data to use for the features contruction, and then was manually revised in order to correct mistagging and discard incorrect sentences (e.g. truncated sentences). After this, it was splitted in train and test datasets.

| **Statistic** | **Value** |
|---|---:|
| Total no. of instances (before filtering) | 23938 |
| Total no. of instances (after filtering) | 20138 |
| Total no. of lemmas (before filtering) | 248 |
| Total no. of lemmas (after filtering) | 208 |
| Total no. of senses (before filtering) | 772 |
| Total no. of senses (after filtering) | 732 |
| Average no. of senses per lemma | 3.52 |
| Average no. of instances per lemma | 96.82 |
| Average no. of instances per sense | 27.51 |
| Average no. of instances for the most frequent sense per lemma | 67.08 |
| Average no. of instances for the second most frequent sense per lemma | 19.67 |

Table 1: SenSem statistics

### 3.2 Word embeddings

This work focuses on the embeddings obtained with the Word2Vec algorithm [14]. The original word embeddings used for the experiments are the pre-trained Spanish Billion Word Corpus and Embeddings (SBWCE) [4]. The SBWCE corpus is a compilation of nearly 1.5 billion words of Spanish from different sources available on the Internet, most of them coming from corpora used for statistical machine translation tasks, as well as corpus from the Wikimedia foundation, making it a heterogeneous domain corpus. The word embeddings pre-trained from this resource were created using Word2Vec's *skip-gram* model. The corpus has over 45 million sentences with more than 3 million unique tokens. Filtering out words with less than 5 occurrences, roughly 1 million unique words are left. The final word vectors dimension is 300.

The general idea of using pre-trained embeddings is their availability. In general terms, embeddings trained on big amounts of data perform relatively well for general tasks, however, we wanted to see the impact of training embeddings specifically for the data available and what effect this has on the results.

SenSem is based on a small fraction of two newspapers from the region of Catalunya in Spain: "El Periódico" and "La Vanguardia". This makes the resource heavily based on senses which have more to do with the journalistic domain. We trained word embeddings based on journalistic sources available on the SBWCE and other newspapers available online, particularly the corpora provided by the two newspapers on which SenSem is based. In comparison to the SBWCE corpus, the corpus we could gather for this task was much smaller, with nearly 71 million words which became 70 million after filtering out all those words with less than 3 occurrences. There was a final list of approximately 240 thousand unique words to generate word embeddings with dimension 50.

## 4 Experimental Setting

In this Section, we explain the general methods used to carry out the experiments of the paper. For the scope of this work, the words *dataset* and *corpus* are interchangeable. The terms *word embeddings* and *word vectors* are also used interchangeably. On the other hand when we use the word **model** we refer to the result of training a *classifier* with a specific *representation.*

### 4.1 Basic layout

The verb sense disambiguation task is done per lemma. This means that we do not train a single classifier for all the different senses, but rather one classifier for each lemma with more than 1 sense available, omitting lemmas with only 1 sense.

In particular, to design the machine learning system we took inspiration from the *It Make Sense* system [23], and use a pipeline similar to theirs. For the purely supervised approach there was a preprocessing step of the labelled corpora where it was analyzed automatically. Then we use the available information to generate features to represent the instances to use as training data. Finally different supervised classifiers were tried in the experimentation phase. For the semisupervised approach we use the word embeddings directly to create a representation of the instances. There was a preliminary exploration using a combination of supervised features and word embeddings. However, after some experiments we decided to stop further experimentation as the preliminary results did not show very good results when measuring performance plus the overfitting of this experiments was much worse than the other possibilities. In the end, only the results using supervised features or unsupervised features alone are the ones we present in this work.

### 4.2 Supervised Features

One of the goals of this work is to asses the impact in VSD of word embeddings, an unsupervised form of feature engineering. To do that, we need to compare our results against a traditional approach using handcrafted features.

To design the features we were based on the work already mentioned in Section 2, specially Ye and Baldwin [22], Màrquez et al. [11], and Montoyo et al. [17].

Features represent the instances of the dataset. Such instances, particularly for VSD, are defined by the word (specifically the verb) to be disambiguated in a sentence. With that word as a focus, the following features are used to represent the instance:

- The main word.
- The main word's lemma.
- The main word's part-of-speech tag: in the case of Spanish part-of-speech tags, only the abbreviated form is used (generally the 2 or 3 first letters).
- In case of Spanish, the morphosyntactic information of the main word is given separately from the part-of-speech tag.
- The bag-of-words of a symmetric 5-word window (i.e. 5 words before and 5 words after the main word): this feature represents the number of occurrences of each words surrounding the main word (without considering it) giving no importance to the position.
- The words, lemmas and part-of-speech tags of the surrounding words in a 5-word window at the corresponding position.
- The bigram and trigram formed by the words before and after the main word.
- The dependency triples formed by the main word, the relation and the words dependant on the main word (inbound dependency triples). And the dependency triple formed by the main word, the relation and the word from which the main word depends or if it is the root word (outbound dependency triple).

### 4.2.1 Feature Hashing

The representation obtained by the previously presented features is highly sparse, as many of the features will appear once or twice in the whole dataset. Moreover the amount of different possible combinations for it will end up with a large amount of features to represent each instance. This becomes expensive to work with and in some cases it is not possible to load the whole data into memory.

An approach to reduce dimensionality of the input vector of a classifier is by applying feature selection. Feature selection relies on the assumption that the features used to represent the data have a lot of redundancy and noise, as the feature crafting may not be perfect for many different reasons (e.g. unfamiliarity with the domain, impossibility to obtain more relevant features automatically, etc.).

However, feature selection adds to the computational cost of training a model. In order to filter features we need to explore the whole dataset. Moreover, removing features decreases the coverage of the model: when disambiguating new examples there is a bigger chance that they will not be represented by the selected features.

*Feature hashing* [20], also known as the hashing trick, is an efficient way of vectorizing features. Unlike feature selection, it is not based on the assumption that some features carry more information than others. It consists in applying a data structure specifically designed to deal with the problem of high dimensionality and sparse representations. In this method, features are vectorized into an array of fixed length by applying a hash function to the features and use the value as an index of the array. The feature count is then stored in the corresponding position of the array.

We use this technique as a way to represent data with a limited amount of memory without removing features. This is useful in the case of having examples on which the selected features using the previously shown method are not present.

After some experimentation we did not find any significative improvement of using feature selection over feature hashing, and taking into account the results by Weinberger et. al. [20] we decided to do the supervised experiments using the feature hashing technique.

## 4.3 Unsupervised Features

Word embeddings are straightforward to use. The idea is to represent each instance (i.e. the sentence with the verb to disambiguate) as a concatenation of word vectors. We use the token of the verb to disambiguate as the central vector in the concatenation, and chose a symmetric window of 5 tokens at each side of the central word making the final vector a concatenation of 11 words. In this way, the final representation not only captures the semantics of the words through the embeddings but also through the relative position of each word with respect to the verb embedding.

If the token is not available in the word embeddings model, we try the token with all lowercase characters and capitalized (first character uppercase and the rest lowercase). If neither version of the token is available we use a vector of zeros of the same dimension that the word embeddings. For the case when the central word is near to the beginning or end of the sentence, we pad the amount of words left to complete the whole vector with zeros. E.g., if the verb is located as the third word from the beginning of the sentence, then to complete the right window we use the word vectors for the first and second token of the sentence and pad with three zero valued vectors before the vectors of two tokens.

Following this adjustment, the input vectors when using the SBWCE corpus are of dimension 3300 and the vectors for the journalistic domain are of dimension 550.

## 4.4 Classifiers

For the purely supervised approach we tried two different kinds of classifiers: linear and non-linear. We choose the classifier with the best performance in a purely supervised approach to combine with the unsupervised representation.

A linear classifier algorithm does the classification process by using a linear combination of the features, they seek to split the high-dimensional input space with some hyperplanes. We explored multinomial naive bayes, logistic regression and support vector machines with a linear kernel.

A non-linear classifier uses a more complex function in order to better approximate the problem. The features can be combined in non linear ways. Thus more complex patterns in the data can be found. Plus, some problems are strictily non-linearly separable. We explored a Decision Tree classifier and a multilayer perceptron.

We also use a simple baseline classifier which assigns the most frequent sense to every instance.

### 4.4.1 Neural network's architecture

As the amount of data is small, there is a high risk of the network memorizing the datasets if there are enough neurons available. Thus we cannot work with a very deep neural network without falling into this problem. That is why we explore neural networks with only up to three hidden layers.

After some experimentation, we found that the best results were given by a neural network with 3 hidden layers of size equal to 200 (the size of the layers did not influence the final results as much as the number of layers).

## 4.5 Metrics

Metrics work alongside visualizations to show different views of a result. For this work we mostly use two kind of metrics: one to measure the performance of a model and one to measure the tendency to overfit of a model.

### 4.5.1 Performance

Performance metrics measure how well an experiment does with respect to a test corpus held out from the training corpus.

Word sense disambiguation has a Zipfian [24] distribution over the data. Thus, there are certain metrics which can affect the perception of how good or bad an algorithm is, because they show better results only when the most frequent class shows better results. Accuracy is a classic example of a biased metric, it measures the percentage of correct guesses from an algorithm, and if the most frequent class shows a large proportion of examples over the whole dataset, accuracy shows a good result, even for a simple baseline (e.g the algorithm that assigns every instance to the most frequent class). In this work we rely on other metrics. In particular, precision and recall are good metrics but they are class-based, which in cases like this, with a large amount of lemmas, each having many classes, results become difficult to follow. The F1-score, a harmonic mean between precision and recall, gives us a simpler way to measure the performance, but still deals with having one value for every possible class. In order to digest the values of the metric for all classes, we use an average. However, averages also have a bias, that is why we use two different ones: *macro average*, and *weighted average*.

*Macro average* is defined as the unweighted mean of the values for each class [10]. In this metric the least frequent senses are as important as the most frequent ones, nevertheless, it also means that extreme class imbalance will drastically reduce the final results. Weighted average is calculated by averaging the metric of each class weighted by the number of instances it has. Classes with more instances have more relevance in the final results.

As imbalanced classes is a challenge we have to deal with, it is important to show how this affects the overall performance of the models. The use of both macro and weighted average of F1-score helps to assess whether a model is having a large bias to the most frequent class.

### 4.6 Tendency to overfit

Another important measure in this work is the tendency of a model to overfit the training examples. This can be measured by analyzing the *error due to high variance*. Manning et al. [10] define it as the variation of the prediction of learned classifiers: it measures how inconsistent the predictions are from one another, over different datasets, not whether they are accurate or not. To calculate this, we divide the dataset in different parts, and calculate the amount by which the prediction over a part of the dataset differs from the expected predicted value over the rest of parts of the dataset. Additionally, we calculate this by adding more instances to the dataset, to observe the tendency to overfit for different sizes of the dataset. This is calculated with the following algorithm:

1. Split the whole corpus evenly in $n$ parts.
2. Take the first part and split it again using stratified $k$-fold cross-validation.
3. Take $k - 1$ folds, train a model and test it against the fold that was left out, saving the error (in this case, cross entropy) for both the training data and test data.
4. Repeat the previous step for each $k$ fold.
5. Add another part to the dataset and repeat steps (3) and (4) until all the data is added.
6. Show the mean and variance of the error of the training and test sets for each step of the algorithm.

The resulting curve shows the variance error of a model as the number of training examples increases. Models with higher variance are overfitting the training data.

## 5 Results and Analysis

The next section reports the results obtained through the different experiments. The final objective is to asses the impact of word embeddings in Spanish VSD.

### 5.1 Supervised Classifier Selection

Before delving into semisupervised models using word embeddings, we need to select from the available classifiers that which performs the best in a purely supervised model. If there is none to perform strikingly better than the rest, it is useful to know at least there is no visual significance between the results of different classifiers.

Figure 1 showcases the comparison of the classifiers we mentioned before as part of our experimental setting. Since it was established that the supervised representation to use in the rest of the experiments

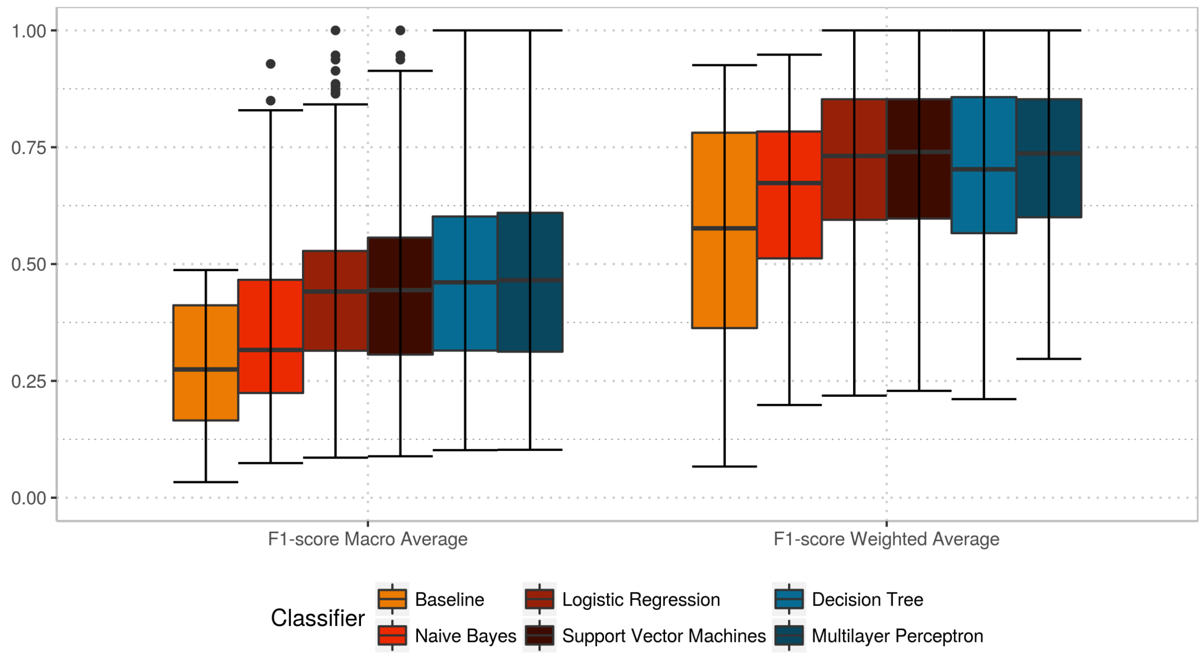

Figure 1: Comparison of classifiers: baseline, linear and non-linear classifiers are displayed from left to right.

of this work is using the hashing trick, the comparison here is only for such representation. The Figure shows a box and whiskers plot. The plot is structured in the following way:

- Each group of boxplots represents a metric: F1-score macro average and F1-score weighted average.
- Each box of different colors inside a group is the classifier: baseline, naive bayes, logistic regression, support vector machines (with linear kernel), decision tree, and multilayer perceptron.
- Linear classifiers are represented by different shades of red and non-linear classifiers are represented by different shades of blue.
- Box and whiskers plots represent the distribution of the values of the metrics through their quartiles. Each value is the performance for a lemma of the corpus. The black thick line in the middle of a boxplot represents the median and the whiskers at the end of each boxplot represents maximum and minimum value (except for eventual outliers represented by black dots outside the boxplot).

The first thing that can be seen in the plot is that all classifiers outperform the most frequent sense baseline classifier. In particular, naive bayes is the one to show the worst results among all classifiers, very near the performance of the baseline classifier, clearly biased by the most frequent sense. On the other hand, the multilayer perceptron classifier shows the best performance. It is specially noticeable the difference in macro average score for non-linear classifiers. Remember that macro average has a bias towards the less frequent classes. From the results there is a strong indication that the problem of VSD is better solved with a non-linear classifier.

The neural network classifier shows the best results for purely supervised word sense disambiguation algorithm. We chose this classifier to integrate with word embeddings.

## 5.2 Word Embeddings Domain

We want to check whether the domain from where the word embeddings are trained has an impact on the final performance of the model. We trained the VSD classifiers of the different lemmas using the two domains of word embeddings we described above: general domain using the whole SBWCE Word Embeddings and specific domain using the Journalistic Corpus Word Embeddings.

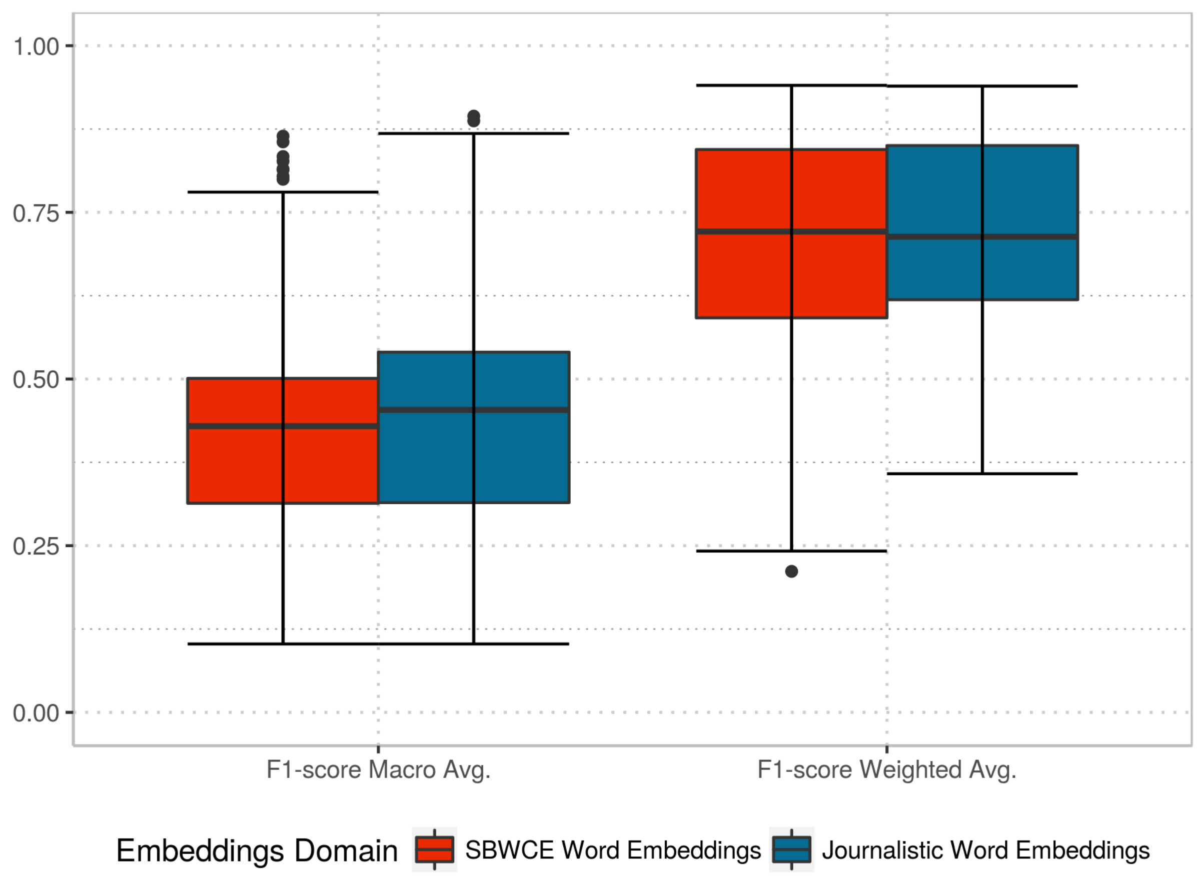

Figure 2: Performance per lemma on the test corpus for VSD integrating general and specific domain word embeddings.

Figure 2 shows the performance on the test corpus for each lemma using two different domains to train word embeddings: a general one (SBWCE) and a specific one (journalistic corpus). The figure shows a box and whiskers plot, where each group of boxplots shows F1-score macro and weighted average. Each boxplot of a different color shows the performance for different training domains of word embeddings: general domain (SBWCE) and specific domain (Journalistic).

From the figure, there is a strong visual indication for the journalistic word embeddins having a better performance than the general word embeddings. This is shown as the median of the performance is higher for the first ones. Besides a better median, the maximum values are also higher. In the case of the F1-score weighted average, there is a better performance for the lowest values, and the difference for the median in favor of the general corpus is marginal. Recall the macro average is good to measure the performance for the minority classes, thus showing that the journalistic word embeddings model better the less frequent senses.

## 5.3 Performance comparison of supervised and semisupervised methods

We compare the performance of the multilayer perceptron with three layers trained with unsupervised representations (word embeddings of the journalistic domain) with models trained with handcrafted features.

Figure 3 shows this comparison with F1-score macro and weighted average. It can be seen that

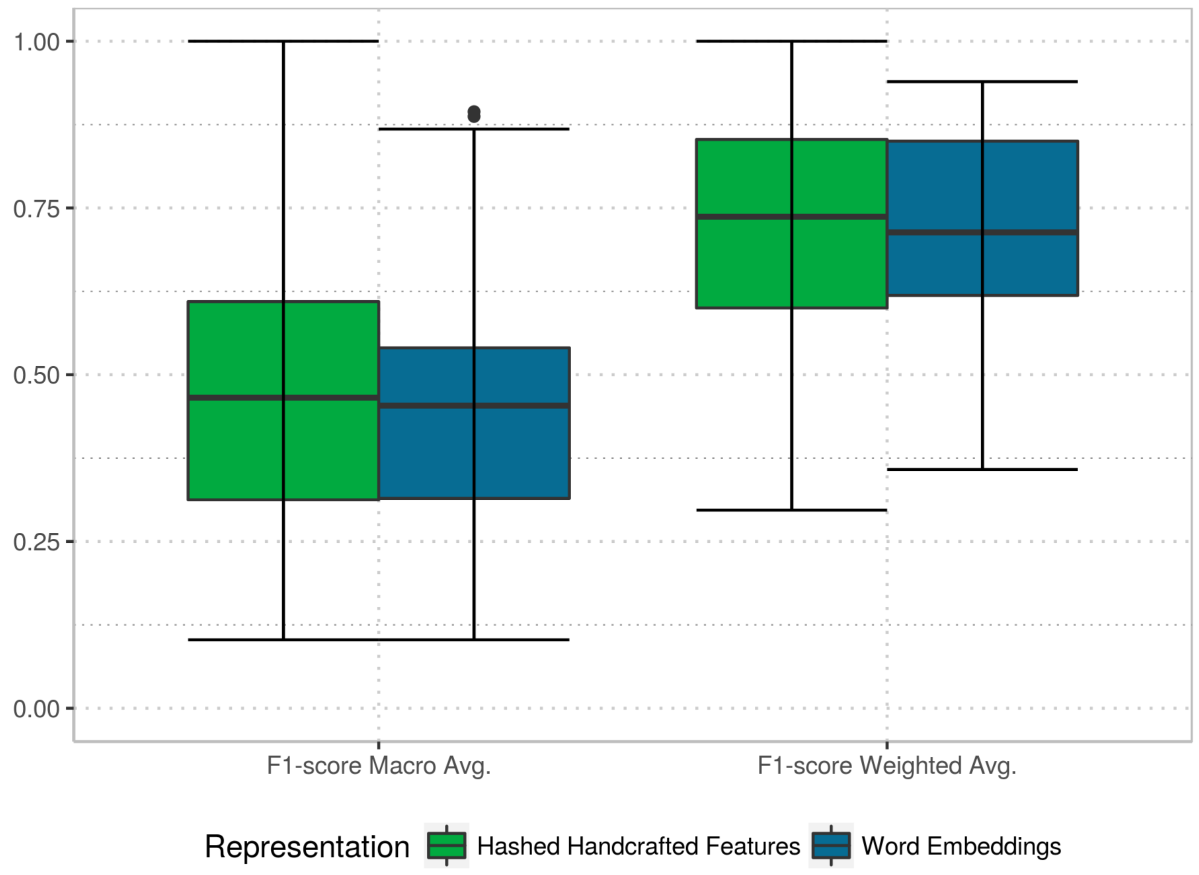

Figure 3: Performance per lemma on the test corpus for hashed handcrafted features and integrating word embeddings of specific domain.

hashed handcrafted features clearly outperform unsupervised representations, specially in F1-score macro average, which means it represents better those senses with low occurrence count. This can be due to the fact that word embeddings serve as a way of smoothing features by reducing their dimensionality to a lower, less specific representation. This may result in underrepresentation of some cases, specifically, unfrequent classes. Besides, the representation which is reduced is already less complex than what handcrafted features provide as it only considers words, leaving out all the rest.

We can hypothesize that handcrafted features perform better than the semisupervised model because the features better represent the domain of the model, since they are taken exclusively from the training data itself, unlike the journalistic word embeddings, which are taken from a more diverse corpus, even if it is journalistic. Supervised features may have a better performance because they fit more closely to the data.

## 5.4 Tendency to overfit of the models

The results of the previous section showed that supervised features perform better than unsupervised features for the VSD task. The results of the experiments in this section give a hint on what is happening underneath the results shown in the previous section.

These results show the learning curve calculated with the metric described in Section 4.6. The results report the learning curve of a model as the number of examples increases. It shows the mean and error due to variance of both the training and test sets on each iteration.

Figure 4 shows the learning curve for different sizes of the training data over the different representa-

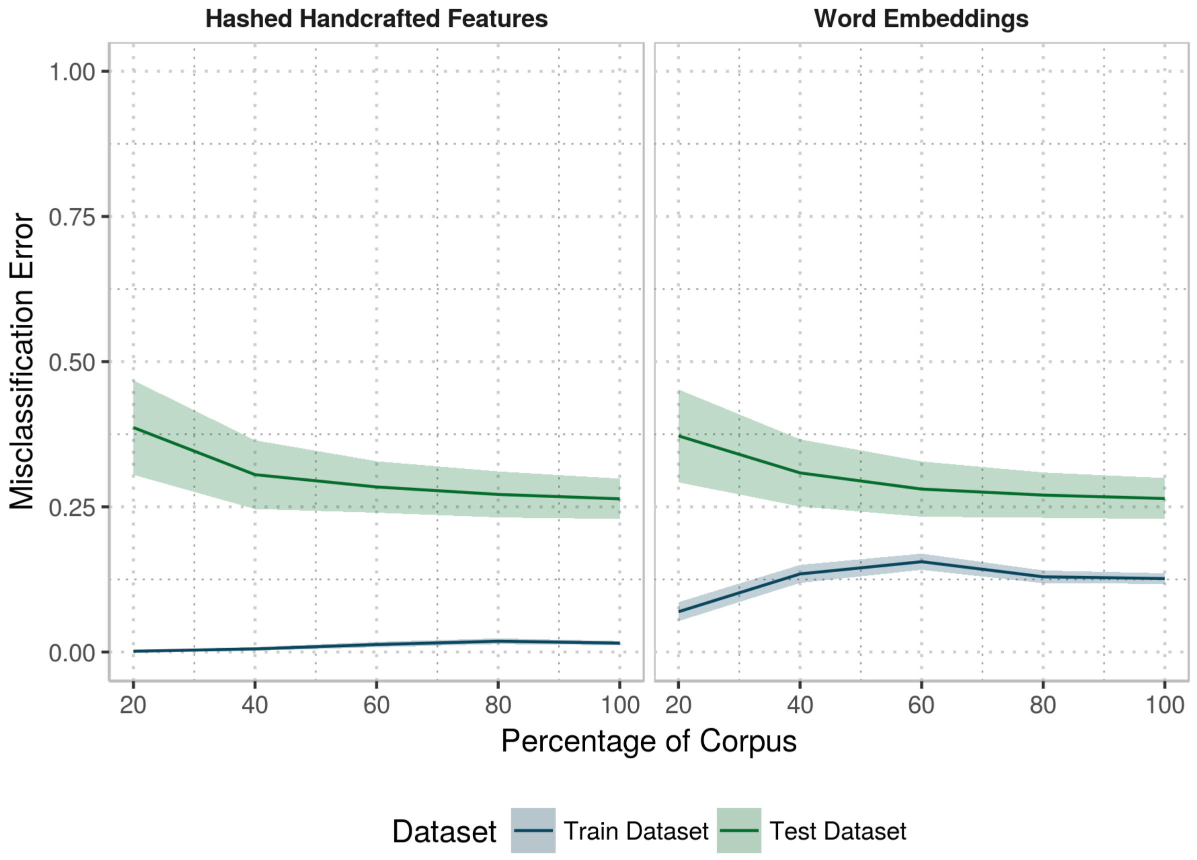

Figure 4: Learning curve for different sizes of traning corpus for supervised and unsupervised representations

tions used as input features: supervised handcrafted features and unsupervised word embeddings. The structure of the learning curve plot is as follows:

- The plot is divided in two columns, each represents an input to the model: supervised handcrafted features and unsupervised word embeddings of the specific domain (journalistic).
- The x-coordinate shows the size of the training data, as a percentage of the total training data available, starting from 20% of the corpus (the corpus was splitted in 5 parts according to what is established in the description of the learning curve metric in Section 4.6).
- The y-coordinate shows the misclassification error of a model.
- There are two colors representing the datasets: train and test.
- The solid darker lines represent the mean of misclassification error trough the different splits of the datasets over all the models.
- The ribbons, which have a lighter color, represent the standard error of the mean of the misclassification error.

It is possible to see in the plot the difference between representations when the tendency to overfit is measured. Word embeddings show less difference between the training data and the test data regarding misclassification, even if the classifier is non linear, which generally have more tendency to overfit data.

The word embeddings are helping to not overfit as much as the supervised representation does. Still, it is important to note that the misclassification error in test data is still the same for one representation or the other, thus the model is not sacrificing training performance in order to gain test performance, but we recall the models are small and neural networks models work better the more information they have. In order to gather more information, more examples are needed and to do so the expansion of a model's examples via unsupervised corpora is needed. And to do so we need models which generalize better to new examples.

## 6 Conclusions

This paper explored the problem of Spanish verb sense disambiugation (VSD) using word embeddings as a semisupervised method in comparison to supervised feature engineering. The resources chosen to do it were the SenSem corpus of disambiguated verbs and the SBWCE embeddings. As SenSem is a manually annotated resource it is small. Then if we want to train a verb sense disambiguator from it a supervised model is the simplest way.

Before starting we setup a supervised baseline by comparing the performance of both supervised and unsupervised features using this baseline. We explored different classifiers and end up selecting a neural network with three layers as our baseline classifier since it showed the best results in our experiments.

But supervised models have a problem: overfitting. On new examples the model may have little information to represent them. One of the causes to overfit in purely supervised models is given by the very nature of such models. They obtain representations from the same annotated data the classifiers are learning from. We aim to overcome this shortcoming by using features which generalize better, not tied to a particular dataset. This is what word embeddings are for: to give a smoother representation of the data.

For semisupervised models using word embeddings, first we showed the performance of the unsupervised representation depends on the domain from where the unlabeled data to train the embeddings is taken. For a specific domain the results improve for the same task.

Finally we compared supervised and unsupervised representations as input for a supervised classifier (in this case the neural network). The comparison was in two different aspects: the performance, measured by the F1-score macro and weighted average; and the tendency to overfit, measured by the learning curve.

Regarding performance, handcrafted features show better results than unsupervised features, however the reason behind this is that supervised features fit more closely to the dataset, as the features come directly from there. The comparison of the learning curves of both representations shows that word embeddings have a more similar performance between training and test datasets.

Unsupervised features representations, particularly those trained from the same domain as the supervised dataset, show promising results. However, the performance is still under what we can achieve using purely supervised representations. And although there can be many reasons for this phenomenon, according to what we see, it most likely has to do with the adaptation of the supervised features to the supervised data, in contrast to the more smooth representation of word embeddings.

We saw using non-linear classifiers such as neural networks could have a great impact in minimizing the error and even maximixing the performance of the test data. But the cost is the generalization of such models. Unsupervised features help in that aspect by giving a smoother representation helping the neural network to avoid the tendency to overfit.

There is another challenge with supervised approaches: coverage of the model. Coverage can be understood as the unseen examples that are part of the labelled classes and the model can reach. These examples add information to the model but as they are not in the annotated corpus. These examples can only be gathered from unlabeled data which needs to be classified. If the model is only affected by the information it already has (i.e. the features extracted from the supervised data), then it is difficult to use it to classify these examples and add their information to the model, effectively expanding the coverage. Word embeddings hold information about the supervised data, used for the model to classify it, and about unsupervised data not present in the model yet. Thus, this information could eventually help a model trained from unsupervised features generalize better and be able to gather information from new examples expanding its coverage.

In future work we will delve on joint semisupervised methods which add data from unlabeled sources and use that to expand the model. More future work of includes using other unsupervised representations, like the ones listed by Turian et al. [19]: Collobert and Weston [21] and Brown clusters [3]. Another line of work would be doing a more thorough error analysis on the the different word embeddings domain, seeing if a better preprocessing of the data can provide better results.

## References

[1] L. Alonso, J.A. Capilla, I. Castellón, A. Fernández, and G. Vázquez. The sensem project: Syntactico-semantic annotation of sentences in spanish. In N. Nicolov et al., editor, *Selected papers from RANLP 2005*, pages 89–98. John Benjamins, 2007.

[2] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Comput. Linguist.*, 1996.

[3] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December 1992.

[4] Cristian Cardellino. Spanish Billion Words Corpus and Embeddings. `http://crscardellino.me/SBWCE/`, March 2016.

[5] Jinying Chen and Martha S Palmer. Improving English verb sense disambiguation performance with linguistically motivated features and clear sense distinction boundaries. *Language Resources and Evaluation*, 2009.

[6] Ana Fernández-montraveta, Ana Fernández-montraveta, Gloria Vázquez, and Christiane" Fellbaum. The spanish version of wordnet 3.0. *TEXT RESOURCES AND LEXICAL KNOWLEDGE*, pages 175–182, 2008.

[7] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany, August 2016. Association for Computational Linguistics.

[8] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: The state of the art. *Comput. Linguist.*, 24(1):2–40, March 1998.

[9] Daisuke Kawahara and Martha Palmer. Single Classifier Approach for Verb Sense Disambiguation based on Generalized Features. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May 26-312014. European Language Resources Association (ELRA).

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[11] Lluís Màrquez, Lluís Padró, Mihai Surdeanu, and Luís Villarejo. UPC: Experiments with Joint Learning within SemEval Task 9. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, 2007.

[12] Llúıs Màrquez, Luis Villarejo, M. A. Martí, and Mariona Taulé. SemEval-2007 Task 09: Multilevel Semantic Annotation of Catalan and Spanish. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, 2007.

[13] Diana McCarthy and John Carroll. Disambiguating Nouns, Verbs, and Adjectives Using Automatically Acquired Selectional Preferences. *Comput. Linguist.*, 29(4):639–654, December 2003.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. January 2013.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. October 2013.

[16] Andriy Mnih and Geoffrey E. Hinton. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc., 2009.

[17] Andrés Montoyo, Manuel Palomar, German Rigau, and Armando Suárez. Combining knowledge- and corpus-based word-sense-disambiguation methods. *CoRR*, 2011.

[18] Lluís Padró and Evgeny Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May 2012. ELRA.

[19] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[20] Kilian Weinberger, Anirban Dasgupta, Josh Attenberg, John Langford, and Alex Smola. Feature hashing for large scale multitask learning. 02 2009.

[21] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1168–1175, New York, NY, USA, 2008. ACM.

[22] Patrick Ye and Timothy Baldwin. Verb sense disambiguation using selectional preferences extracted with a state-of-the-art semantic role labeler. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 139–148, Sydney, Australia, November 2006.

[23] Zhi Zhong and Hwee Tou Ng. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 78–83, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[24] George Zipf. *Human Behavior and the Principle of Least Effort.* Addison–Wesley, Cambridge, MA, 1949.