A comparative study of classifier combination applied to NLP tasks

Fernando Enríquez, Fermín L. Cruz, F. Javier Ortega, Carlos G. Vallejo, José A. Troyano *

Universidad de Sevilla, Escuela Técnica Superior de Ingeniería Informática, Avenida Reina Mercedes, s/n 41012 Sevilla, Spain

Keywords:
Natural Language Processing
Text analysis
Classifier combination
Part-of-speech tagging

A B S T R A C T

The paper is devoted to a comparative study of classifier combination methods, which have been successfully applied to multiple tasks including Natural Language Processing (NLP) tasks. There is variety of classifier combination techniques and the major difficulty is to choose one that is the best fit for a particular task. In our study we explored the performance of a number of combination methods such as voting, Bayesian merging, behavior knowledge space, bagging, stacking, feature sub-spacing and cascading, for the part-of-speech tagging task using nine corpora in five languages. The results show that some methods that, currently, are not very popular could demonstrate much better performance. In addition, we learned how the corpus size and quality influence the combination methods performance. We also provide the results of applying the classifier combination methods to the other NLP tasks, such as name entity recognition and chunking. We believe that our study is the most exhaustive comparison made with combination methods applied to NLP tasks so far.

1. Introduction

In Natural Language Processing (NLP), there are many lines of research based on the classification of words, grammatical constructions and text documents in a number of predefined categories. In recent years there have been many classification algorithms developed using Machine Learning techniques based on multiple theories and approaches. These features give them distinguishing characteristics that make them better suited for some tasks than for others preventing the appearance of the "perfect" algorithm for any problem.

Machine Learning is an area where these algorithms have played a major role and the combination of classifiers has been studied with great interest. In addition to investigating the results given by combination methods, a great effort in demonstrating the theoretical advantages of their use over the application of a single classifier has been made. The underlying idea is simply to get the most out of the different views that different classifiers provide facing the same problem. NLP researchers use these combination methods for their own classification tasks, generating a succession of papers that since the late nineties continue providing improvements to their initial results.

* Corresponding author.
E-mail addresses: fenros@us.es (F. Enríquez), fcruz@us.es (F.L. Cruz), javierortega@us.es (F. Javier Ortega), vallejo@lsi.us.es (C.G. Vallejo), troyano@us.es (J.A. Troyano).

1.1. Classifier combination

What does it take for the combination to be successful? What does classifier combination offer? What are the reasons to believe that the combination will improve the results of one classifier? Following the studies of Hansen and Salamon [1] there are two basic requirements necessary and sufficient for the combination to succeed in improving the individual classifiers:

- Diversity: Classifiers must make different mistakes when faced with new data to classify. This is to provide different views of the same problem, which is achieved in many different ways.
- Accuracy: Classifiers involved should provide a lower error rate than a random classifier.

When justifying the confidence in the combination as a method capable of improving classification results, Dietterich [2] suggests three reasons:

- Statistical: Selecting one of the classifiers can lead us to a mistake as it may not be the best classifier for our problem. Furthermore, we do not have infinite resources and the limited number of examples of the training database can make a particular classification algorithm suffer more than others.
- Computational: Even if we had unlimited training data, classifiers could fall in a local maximum or other phenomena that avoids them from reaching their goal.
- Representational: Finally, the search space explored by a particular classifier algorithm may not contain the objective function we are trying to approximate. This can occur for all the classifiers at our disposal.

Also one more reason to combine is the ability to avoid or mitigate overfitting according to [3].

Kuncheva [4] also shows that classifier combination can be addressed following different approaches. These will depend on the point where the diversity is generated in the whole process. This way she distinguishes four types of combination separated in levels:

- Combination level: At this level we have to decide which combination algorithm to use for merging the categories proposed by the base classifiers.
- Classifier level: Here we focus on the type of base classifiers we can use among all the available algorithms developed so far.
- Feature level: When an example is represented by a feature vector in the database we can also focus on deciding which features to use.
- Data level: At this last level we can use different datasets to generate different classifiers. This includes all methods that generate several versions of a dataset by sampling or any other technique.

Finally, another important element to consider is the type of outputs provided by the base classifiers for each element. According to Xu [5], there are three levels of outputs:

- Abstract level: The classifier provides the final category or class among the list of possible categories.
- Rank level: The classifier provides a list of categories sorted according to the confidence calculated by the classification algorithm.
- Measurement level: The classifier provides a list of categories along with their confidence values.

1.2. Classifier combination in NLP

Since the nineties, researchers in the field of Natural Language Processing began to apply more frequently combining techniques to improve their classification results. Papers reflected these improvements although the combination techniques used to be rather simple. Among the mostly used methods in these early works were the averages of the results obtained by the base classifiers. Then they started to discover the potential of using linear combinations, Bayes or more complex classification algorithms using meta-learning.

In [6] different Boolean formulations of queries on a database of documents are combined for the information retrieval task. Although the combination methods differ from the more generic ones applicable in other domains, the results evaluated on the TREC (one of the most important competitions dedicated to this task) resources were promising and served as a starting point for later works. A year later, in [7] three different systems (knowledge-based, example-based and a lexical transfer system) were also combined to solve the task of machine translation (MT). The conclusion, so well exposed in the title of this work, was that three heads are better than one.

In [8] these ideas were applied to the task of document classification, and more particularly to document filtering. Using different types of classifiers (nearest neighbors, Rocchio, Linear Discriminant Analysis and neural networks) several experiments were made with different combination methods (average and regression techniques). The conclusion was that simple average systems offered better results, although the authors expected that future work could achieve improvements using Bayesian inference models.

In the other major branch of NLP, speech processing, researchers also began to pay more attention to combination methods following the publication of papers like [9]. In this particular work a post-processing stage in automatic speech recognition is performed, using voting techniques to reduce the error rate of the different systems used.

All these works generated confidence in the application of combination techniques to NLP tasks, but was in 1998 with the publication of [10,11], when a larger number of researchers started to develop their work in this direction.

These two papers evolved in parallel and both faced a basic task in NLP like part-of-speech tagging (POS). Although van Halteren was experimenting since 1992 with classifier combination using voting techniques, it was with [10] when he gathered more attention. The results obtained with the LOB (Lancaster-Oslo/Bergen) corpus by applying several techniques were definitely outlined. The methods applied varied from simple majority voting or total voting precision to stacking using second-level classifiers, through a pair voting system that provided very good results. Decision trees and example based learning algorithms were also tested as meta-classifiers. On the other hand, Brill and Wu showed in [11] similar conclusions after applying simple voting techniques and meta-learning methods. Again, after several experiments using the Penn Treebank corpus, the option of using a second level of learning obtained the best results.

Other works have employed combination methods for improving performance in POS tagging [12–14], and also in many other tasks such as word sense disambiguation [15–17], named entity recognition [18–20], different types of parsing [21,22], document classification [23,24], information extraction [25,26], and opinion extraction [27,28] among others.

However, the coverage of the combination methods applied to NLP tasks has been quite limited, showing a clear tendency to use voting and stacking techniques against other methods. To prove this we conducted a bibliographical analysis in which we selected a large number of papers that use the combination of classifiers in NLP tasks. We extracted information about the base classifiers used, combining techniques applied and many other data. This study shows, as seen in Fig. 1, that base classifiers usage is very diverse while voting and stacking are chosen much more frequently than any other combination method. The reasons may be the simplicity of the first and the good results announced for the second, which has proven to be capable of achieving significant improvements in several tasks where it has been applied.

Furthermore, if we take the improvements achieved in these papers using combination techniques and calculate the minimum, maximum and average mean, we get the values shown in Table 1. Even though the resources used are different as well as the measures, that may be percentage of accuracy (in most cases), $F_{\beta=1}$, etc., the information shown is important from our point of view. In this summary we see that the improvements are larger for those tasks in which the base classifiers provide worse results because of their difficulty. When the margin for improvement is bigger the combination seems to get the best out of the situation. This occurs even in high demanding scenarios like POS tagging, where base classifiers provide very good results, but combination is still able to surpass those values. Also if we focus on the combination methods used, we see that the mean obtained by other methods different than voting and stacking is better than the obtained by these most used techniques. So we may conclude that there could be very good methods that are not well known by the NLP community.

For this reason we decided to conduct a comparative study of the most relevant combination methods to a level of depth not present in the literature as far as we know. We collected a set of corpora with different sizes, tagsets, languages and sources. We have also made experiments with specific scenarios to show additional aspects that can be taken into account when assessing a particular method. There are many important issues about these

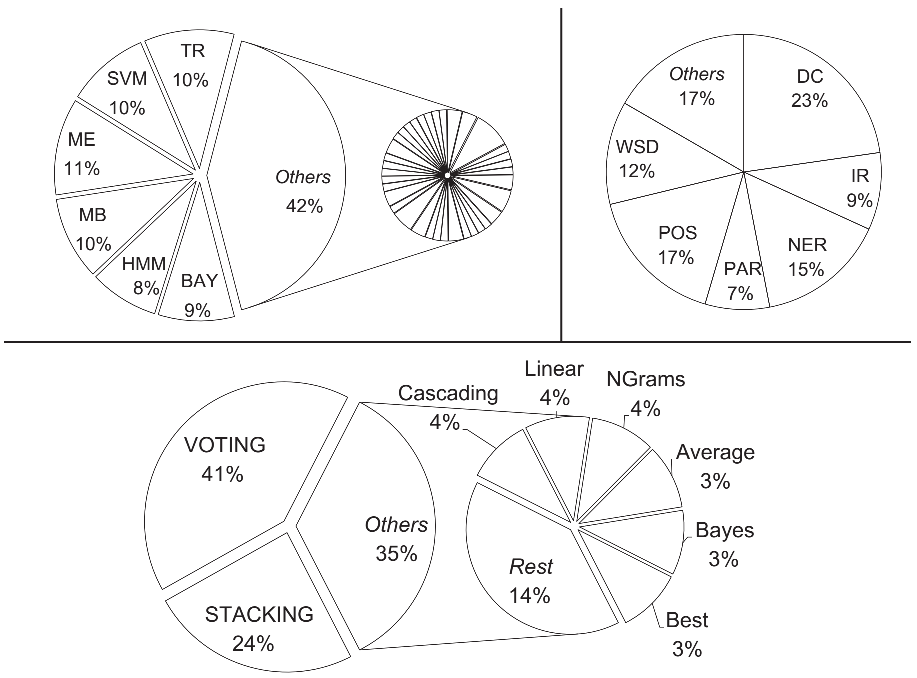

Fig. 1. Distribution of classifiers, tasks and combination methods used. Classification techniques shown are transformation rules (TR), support vector machines (SVM), maximum entropy (ME), memory based (MB), hidden Markov models (HMM) and Bayes (BAY). The NLP tasks are word sense disambiguation (WSD), part-of-speech tagging (POS), parsing (PAR), named entity recognition (NER), information retrieval (IR), and document classification (DC).

Table 1
Summary of results obtained in selected papers. For each NLP task included, the average result obtained by the best base classifier is shown. Also the minimum, maximum and average improvements obtained using combination are calculated. The NLP tasks are document classification (DC), named entity recognition (NER), parsing (PAR), part-of-speech tagging (POS), and word sense disambiguation (WSD). In the bottom, the same information is shown separating the cases in which the most popular combination methods are used (voting or stacking) from the rest.

		Improvement		
	Avg.	Min.	Max.	Avg.
DC	68.72	0.01	8.10	2.02
NER	77.63	1.30	6.41	3.52
PAR	66.06	0.03	2.30	1.12
POS	94.86	−0.58	1.75	0.75
WSD	72.28	1.70	7.00	3.34
Voting	71.17	−0.58	6.20	1.53
Stacking	61.68	0.03	9.02	2.45
Others	63.83	0.02	8.10	2.62

methods like their robustness or adaptability to changes in data or base classifiers.

In the following sections we will show in first place the methodology and general aspects of our experiments in Section 2, followed by the results obtained applying several combination methods to POS tagging in Section 3. Afterwards we will dedicate Section 4 to explain additional experiments focused on special situations that in our opinion give even more importance to the use of these techniques. Finally, we will show more results obtained with other NLP tasks in Section 5 ending up drawing some conclusions considering all the information collected in this work in Section 6.

2. Experimental framework

After an exhaustive bibliographical analysis we have not been able to find any comparative work taking into account a big number of combination methods applied to different corpora. There are important papers that show improvements by using several classifiers to which three or four combination techniques are applied at most. Also the resources in each case are very diverse, making it difficult to know the real potential of each method. A wider experimental framework would give a clearer perspective about this particular way of obtaining better results using the same classification tools employed nowadays. Therefore, we have focused our work on this idea selecting a well known NLP task, many different data collections, some of the most popular classification tools and implementing a large number of combination techniques. Furthermore, we have also made simulations that can provide even more information about how these methods behave when special circumstances appear.

2.1. Methodology

First, experiments will focus primarily on the Part-of-Speech (POS) tagging for two reasons, the availability of classifiers and corpora and the difficulty for the combination to improve the results due to the high level of accuracy obtained by the base classifiers. However, we will also provide results on other NLP tasks to confirm that the observed behavior remains the same.

Second, we will use a total of nine POS corpora and four more to assess other tasks, which is a great variety of data regarding the number of examples, languages, tagsets used, etc. This allows us to check the robustness and consistency of the evaluated methods.

Third, our own implementations have also been developed for the different combination methods trying to respect the basic ideas that support them. This way we avoid the use of *ad hoc* optimizations that can alter the perception of the real potential of every algorithm involved in the study. We have implemented at least one representative from each of the families of combination methods that can be found in the state of the art, except for those requiring measurement level entries because the base classifiers used do not provide them.

Finally, in Section 4 we present a number of situations or scenarios where combination is shown as a promising option, capable of providing better or more robust final classification systems. In some of these cases the possibility to combine usually goes unnoticed because of an apparent lack of diversity. Thus we have experimented with different corpora sizes, altering the quality of the classifiers or exploiting heterogeneous information.

2.2. Base classifiers

We have selected some tools that were designed to solve NLP sequence tagging tasks and meet the essential requirements for implementing the combination. These requirements are the diversity to provide different perspectives on the problem, accuracy to be useful for the combination set and efficiency to be applied without excessively penalizing runtime. The following briefly explains the theory in which they are based.

2.2.1. TnT

The Trigrams'n'Tags [29] (TnT)[2] classifier is a statistic tagger developed by Thorsten Brants for the POS tagging task. This is a very efficient tagger easily trainable for any language, tagset or domain. According to the creator it is not optimized for any particular language, but to be trained by a great variety of corpora and to achieve great execution speeds. It is based on second order Markov models using the Viterbi algorithm and has a variety of methods to deal with the unknown words and smoothing problems.

2.2.2. TreeTagger

TreeTagger [30] (TT) is a probabilistic tagger also focused on the task of POS tagging developed by Helmut Schmid, from the University of Stuttgart. It differs from TnT and classifiers based on *n*-grams in general by the use of binary decision trees to find the probabilities of different possible sequences of labels (transition probabilities). The aim is to avoid problems in these methods for estimating small probabilities accurately through a limited number of training data. The probability of each label is obtained by following the corresponding path through the tree until you find a leaf. The tool displays by default only the label proposed in each case, which is obviously the one with the maximum likelihood.

2.2.3. MBT

Memory Based Tagger [31] (MBT)[3] is a POS tagger developed by ILK and CNTS groups of the Dutch Universities of Tilburg and Antwerp respectively. It makes use of memory-based learning, which is an adaptation of the nearest neighbors algorithm (k-NN) used in pattern classification. To make use of MBT it is necessary to previously install the TiMBL software, since MBT is an extension of the functionality offered by TiMBL. The tag of a word in a particular context is extrapolated from the similar cases stored in memory. This approach is based on the assumption that reasoning is based on the direct reuse of past experiences rather than the application of induced knowledge, as stated for example by decision trees.

2.2.4. FV

Our contribution to the list of base classifiers used is the implementation of a classifier based on the generation of feature vectors and their subsequent classification by support vector machines (SVMs). This tagger called FV provides a variant in terms of theoretical foundations that underlie the different classifiers as well as being devoid of any optimization exclusively linked to the POS tagging task. This introduces an additional distinctive characteristic that increases the diversity, which is a major factor in the application of combination methods. FV has delegated the application of SVM to *SVM*^{light}[32][4] software, which implements the support vector machines in C. The included features, listed below, are the most commonly used in the literature.

- *Lexical features*: Regular expressions are used to determine if the word has some type of lexical characteristics or follow certain patterns. "Begins with capital letter" or "contains numbers" are examples of these patterns we look for.
- *N*-grams: Unigram, bigram and/or trigrams that form the context of a particular word.
- Prefixes and suffixes: Prefixes and suffixes of size 1, 2, and 3.
- Word length: Feature that reflects the size, in number of characters of a word.
- End of sentence: Feature that includes the term the sentence ends with, usually a period or question mark.

To complete the vector we use a sliding window scheme of a configurable size. The parameters that can be specified are:

- WINDOW: Sliding window size.
- FEATURES: List of features to consider.
- LEXFEAT: Filename with regular expressions associated with the lexical features.
- RS: Threshold used to discard a feature when applying the random subspace method to generate random feature subsets.

Table 2
Summary of corpora used. In this table we show several characteristics of the corpora used in our experiments with POS tagging.

CORPUS	Language	Tags	TRAIN		TEST	
			Sentences	Words	Sentences	Words
Brown	ENG	83	14,101	1,048,112	1566	113,080
CoNLL'00	ENG	34	8936	211,727	2012	47,377
CoNLL'02	DUT	13	15,806	202,931	2895	37,761
CoNLL'07	SPA	15	2949	75,822	563	19,206
	EUS	23	2595	40,032	580	10,096
Floresta	POR	24	8340	195,538	926	17,113
Susanne	ENG	131	5754	141,140	830	15,482
Talp	SPA	11	3492	91,400	389	9071
Treebank	ENG	37	25,117	766,463	1513	46,461

3. Applying combination to POS tagging

Among all the tasks related to the sequential tagging of texts, we selected POS tagging mainly due to the availability of resources and the good results that the base classifiers provide. This allows us to test the combination methods in a high demanding scenario because of the little margin for improvement left by these good initial results. However, we have also tested other tasks that will be discussed later to see if they confirm the results of this section.

3.1. Corpora

First of all we will show the corpora used in the experimentation phase, indicating its origin and a brief description of its contents in each case. In Table 2 a summary with the number of sentences and words they contain can be seen, as well as the tagsets used. These tagsets may not coincide with the original ones due to small changes that may have been made because of various reasons: correction of errors in the corpus, to avoid incompatibilities with the implementation of the classifiers, to increase understandability, etc. In cases where there was only one dataset available, a 10% of the examples were extracted to be used as the

[2] http://www.coli.uni-sb.de/~thorsten/tnt.
[3] http://ilk.uvt.nl/mbt/.
[4] http://www.cs.cornell.edu/People/tj/svm_light/.

test corpus leaving the rest for training. We split sentences into a word per line when needed and when a tag can be linked to a group of words we use the IOB format. Therefore, if the tag is *X* we write 'B-*X*' for the first word of the group, 'I-*X*' for the rest of the words of the group (if any) and 'O' for those words that are out of this group or any other group.

- Brown: The Brown[5] corpus is defined as:

  A Standard Corpus of Present-Day Edited American English, for use with Digital Computers.

  It was developed at the Department of Linguistics in the Brown University by Francis and Kucera in 1964, although it has been revised and extended several times. In addition to transforming the content to the IOB format with a word per line for implementation reasons, we joined all the sections that compose the corpus and slightly simplified the original tagset.

- CoNLL 2000: This is the corpus used in the yearly CoNLL meeting held in Lisbon in year 2000,[6] dedicated to the text chunking task, which consists of dividing a text in syntactically correlated parts of words. The original data is formed by three columns separated by spaces, each word being represented by a line from the corpus. The first column contains the word, the second is the POS tag and the third is the corresponding chunking tag. The existence of the POS column allows us to use this corpus for the POS tagging task ignoring the chunking column. However, we also experimented with a particular case of combination, which tries to exploit the possibility of having this heterogeneous information.
- CoNLL 2002: The CoNLL edition of 2002[7] was focused on the Named Entity Recognition (NER) task and provided a corpus in Spanish and another one in Dutch. In the latter there is a column with the POS tag, so it offers the possibility of using this corpus for the POS task, not only for the entity recognition. The data comes from four editions of the Belgian newspaper "De Morgen" of 2000 and the tagging was carried out by members of the University of Antwerp.
- CoNLL 2007: In its 2007 edition, the CoNLL[8] shared task was dependency parsing. For this purpose several corpora were used and a subset of them were released under the terms of the Creative Commons license. In this case we have only used the information regarding the POS tagging in Spanish and Euskera.
- Floresta: The Floresta project[9] created a syntactically analyzed corpus in Portuguese that includes the POS tags associated with the text. These are the tags we have used to apply this corpus for the POS tagging task.
- Susanne: The Susanne corpus[10] was created at the University of Sussex by the team of Geoffrey Sampson. It is based on 64 of the 500 texts of the Brown corpus and has a very extensive tagset. In addition to transforming the text format to a word per line, a slight simplification of the huge initial tagset has been carried out for implementation reasons.
- CLiC-TALP: The CLiC-TALP[11] is a Spanish corpus with one hundred thousand words morphologically analyzed and manually disambiguated. It was developed by the Servei de Tecnologia Lingüística (STeL), which belongs to the Faculty of Philology in the University of Barcelona.
- Penn Treebank: Finally, the Penn Treebank is available through the Linguistic Data Consortium at the University of Pennsylvania and it consists of one million words from material extracted from the Wall Street Journal in 1989. Considering the parsing corpora, they are divided into two large groups: those based on the tagging of different syntactic categories and the ones focused on dependency structures. The Penn Treebank is one of the most popular corpora that belong to the first group.

3.2. Baseline

In order to know the starting point for the different combination methods behind these experiments, the base classifiers have been executed independently using the default parameters, obtaining the results shown in Table 3.

As it can be seen, TnT obtained in most cases the best result, which should result in a bigger influence when considering the different tags proposed by the classifiers during the combination process. It is also important that the rest of the classifiers do not always hold the same order in respect to the best results obtained and that the differences vary in significant quantities when we change from one corpus to another.

3.3. Combination methods

Once the available corpora and base classifiers have been shown, we now focus on the combination methods to be applied. As occurred with the FV classifier, we decided to implement ourselves the combination algorithms to avoid possible optimizations that harm the comparison. Our goal was the highest correspondence between the results and the potential of each original algorithm. This also allows us to obtain a high level of consistency and flexibility that facilitates the creation of more complex meta-learning combination schemes. Below is a brief description of each method. In the results table we will find the improvements achieved in relation to the best result obtained by a base classifier, which will be considered the baseline.

3.3.1. Voting

Voting methods relate the concept of 'elections' to the problem of classification, 'voters' to the classifiers involved and the 'votes' to the tags proposed by each of them. This allows to apply the election methods found in the Social Choice Theory [33], whose goal is the study of collective decision making according to individual preferences.

As representatives of the different existing voting methods we have implemented the two most popular versions of this type of algorithms, namely the plurality voting and the weighted voting (see Fig. 2). It is important to notice that other variants such as the simple majority or unanimity require the reject option to be allowed. This means it must be possible not to choose any candidate,

Table 3
Individual base classifier results. The percentage of accuracy obtained by the four base classifiers is shown for every corpus. We have highlighted in italics the best result for each corpus.

CORPUS	FV	MBT	TnT	TT
Brown	96.18	95.82	*96.55*	95.64
Conll00Pos	96.41	96.80	*97.32*	96.41
Conll02nedPos	95.01	95.79	*96.16*	88.53
Conll07esp	95.35	95.01	*95.98*	95.44
Conll07eus	91.27	90.59	93.73	*94.13*
Floresta	96.52	95.81	*97.02*	96.66
Susanne	92.26	91.16	*93.61*	91.27
Talp	94.59	94.80	*95.82*	95.62
TreebankWSJ	*96.28*	95.67	96.21	95.52

5 http://www.hit.uib.no/icame/brown/bcm.html.
6 http://www.cnts.ua.ac.be/conll2000.
7 http://www.cnts.ua.ac.be/conll2002.
8 http://nextens.uvt.nl/depparse-wiki/SharedTaskWebsite.
9 http://www.linguateca.pt/Floresta/.
10 http://www.grsampson.net/Resources.html.
11 http://clic.fil.ub.es/demos/.

something that is not always feasible. To estimate the weights associated with the different participants in the weighted voting scheme, the hold-out method has been chosen. This method divides the training corpus to use, one half for training and the other for testing, considering the average of the two accuracies as the final estimation.

If we combine the four classifiers using the two variants of voting for all different corpora, the results shown in the two first columns of Table 4 are obtained. They show a better performance of the simple method (VT) versus the weighted (VTw). This is because the number of classifiers is not very large, which causes the best of them to obtain a privileged position (higher ratio) and makes it very difficult for the rest to collect enough votes to change the result. In many cases, it only manages to match the result of the best base classifier without any improvement, but this should not be considered as a defect as it can be very useful in certain situations. For example, when we do not know which of our classifiers is the best and in "conservative" situations where we want to make sure not to worsen the outcome we would obtain using the best base classifier available.

3.3.2. Bayes

Methods based on Bayes' Theorem make use of the rule that shows how to obtain the conditional probability of an event A given B in terms of the conditional probability of the event B given A and the prior probability of A. This type of classifiers have been tested in many papers with very good results, often better than those obtained by more complex algorithms ([34]).

The implementation carried out is a direct application of this rule as shown in Fig. 3. We calculate the probability that given a set of possible tags $(s_1, \ldots, s_k)$ and a number of observations $r = r_1, \ldots, r_N$ representing the proposed tags by the N classifiers, the real tag is s_j, with $j = 1, \ldots, k$. Adapting the original equation we can calculate the support for tag s_j as described in Eq. (1). In the practical implementation we introduce the concept of confusion matrix of a classifier c_p (M_p), which is a $k \times k$ matrix where $M_p[m, n]$ shows the number of words of the training corpus T whose correct tag was s_n, and were assigned by c_p the tag s_m. By Z we denote the number of tags in T and by Z_j the number of s_j tags in T. Thus we take Z_j/Z as an estimate of $P(s_j)$ and $M_p[j, r_p]/Z_j$ as an estimate of $P(r_p|s_j)$.

$$support[j] = P(s_j)\prod_{p=1}^{N} P(r_p|s_j) \tag{1}$$

The application of this method with the four base classifiers and the different corpora yields the results shown in Table 4. Once again, there is a significant improvement in all corpora, so we consider this method as an easy to implement option, but very profitable in terms of the results you get.

```
Voting:
1. Create base classifiers $(c_1,\ldots,c_N)$ using the tagged training corpus $T$.
2. Calculate the weights of the classifiers:
   2.1 If weighted voting is selected, calculate classifier weights $(w_1,\ldots,w_N)$ using part of $T$ for training and the rest as a test corpus to measure the accuracy of each classifier $(a_1,\ldots,a_N)$: For $p = 1$ to $N$, $w_p = a_p / \sum_{q=1}^{N} a_q$
   2.2 Else: For $p = 1$ to $N$, $w_p = 1$
3. For each word $t[i]$ in the untagged test corpus $t$,
   3.1 Initialize the support given by the base classifiers to each possible tag: For each tag $s_j \in (s_1, \ldots, s_k)$, $support[j] = 0$.
   3.2 Considering $r_p[i]$ as the tag assigned to word $t[i]$ by classifier $c_p$, count the "votes" for each tag: For $p = 1$ to $N$, $support[r_p[i]] = support[r_p[i]] + w_p$
   3.3 Select the most voted tag and assign it to the current word: $t[i] = \arg\max_j support[j]$
```

Fig. 2. Voting combination algorithm.

3.3.3. Behavior knowledge space

Methods based on memorization try to recall the behavior demonstrated by the classifiers with the training examples and then make decisions about the new examples based on the past. As a representative of this type of algorithms, the method Behavior Knowledge Space [35] has been implemented following the guidelines shown in Fig. 4.

Its objective is to estimate the probability $P(s_j|r)$ for each tag $s_j \in (s_1, \ldots, s_k)$ and every possible combination of results $r \in \Omega^N$ of the base classifiers. In practice the classifiers are executed with all the examples $T[i]$ of the training corpus T, generating the vector $x = [x_1, \ldots, x_N]$ for each one of them. Those vectors are stored in an indexed table together with the number of times those vectors appeared associated with every possible tag. To classify a new element, we calculate the vector x for that element and access the indexed table recovering the most used tag among the elements that generated the same vector. The ties and empty cells are problems usually solved by selecting at random a class or using a voting scheme to select the final tag. Each table entry $BKS(x)$ has three fields:

- $n(x)(k)$: Number of times where the combination of classifier results x is associated with tag k.
- $S(x)$: Total number of occurrences of that combination x in the training data.
- $R(x)$: Most representative tag associated with combination x.

Using this information the confidence of each possible tag is $Belief(s_j) = n(x)(j)/S(x)$. Finally the result will be R_x if $S(x) > 0$ and $Belief(R_x) \geqslant \alpha$. In other case the algorithm should abstain as it could not surpass the given confidence threshold α.

Once more this method was run with the base classifiers leading to the values shown in Table 4. The improvements are again widespread in all corpora and even larger than those obtained by previous methods, reaching a maximum of 1.36 of accuracy. This maximum value was obtained using the Susanne corpus, which is characterized by its difficulty due to its big tagset containing more than one hundred possible tags.

3.3.4. Bagging

A method that stands among those that generate variability in the data is bagging [36]. It combines different versions of the original corpus created by sampling with replacement. We have slightly modified the bagging algorithm to make it able to combine several base classifiers as the other combination methods do (see Fig. 5). Therefore, what our version does is to generate variability creating a different 'bag' in each iteration as usual and making all the base classifiers share it providing more than one tag to combine with the rest of iterations. In this case we have experimented with TnT, TreeTagger and MBT, excluding FV for being the classifier that needs more time, which gains importance in an iterative process like bagging as this greatly harms the efficiency of the system. The execution scheme is shown in Fig. 6, where we can see how the different samples of the training corpus T are used by the N classifiers to create N tagged versions of the test corpus t.

In another experiment that we call BAGw, the variability is not generated using the corpus. What is done in this case is the sampling of the database created with all the outputs provided by the base classifiers using the Weka [37] tool. The execution scheme coincides with that of stacking, using two learning levels as will be discussed later and selecting bagging as the second level classifier. In fact, it is a cascading scheme (a method that will also be seen later), because it starts running the base classifiers to generate the database and then Weka uses one of its classification algorithms and applies bagging to it.

Table 4
Combination results. The accuracy percentage improvement obtained by every combination method is shown for each corpus. This improvement is always in relation to the best result achieved by the base classifiers. The methods are voting (VT), weighted voting (VTw), Bayes (BAY), behavior knowledge space (BKS), bagging (BAG), bagging using Weka (BAGw), bagging with stacking instead of voting (BAG-SG), and stacking (SG). The last two columns calculate the average mean of the percentage accuracies of the base classifiers (AVG1) and combination methods (AVG2) for each corpus.

CORPUS	VT	VTw	BAY	BKS	BAG	BAGw	BAG-SG	SG	AVG1	AVG2
Brown	0.49	0.24	0.39	0.63	0.22	0.32	0.25	0.64	96.05	96.95
Conll00Pos	0.45	0.00	0.37	0.65	0.17	0.55	0.28	0.71	96.74	97.72
Conll02nedPos	0.68	0.00	0.52	0.67	0.24	0.51	0.37	0.66	93.49	96.27
Conll07espPos	0.98	0.00	0.88	1.10	0.15	0.90	0.64	1.34	95.45	96.73
Conll07eusPos	0.16	0.00	0.43	0.51	0.25	0.22	0.22	0.49	92.43	94.42
Floresta	0.63	0.00	0.55	0.72	0.26	0.36	0.23	0.78	96.50	97.46
Susanne	0.71	0.00	0.67	1.36	0.25	0.81	0.30	1.26	92.08	94.28
Talp	0.76	0.33	0.96	1.08	0.49	0.75	0.62	1.10	95.21	96.58
TreebankwsjPos	0.45	0.12	0.27	0.47	0.14	0.35	0.22	0.59	95.92	96.58
Mean	0.59	0.08	0.56	0.80	0.24	0.53	0.35	0.84	94.92	96.36

```
Bayes:
1. Create base classifiers $(c_1, \ldots, c_N)$ using the tagged training corpus $T$.
2. Build the confusion matrix for each classifier using partitions of the training corpus $T$.
3. For each word $t[i]$ in the untagged test corpus $t$,
   3.1 Initialize the support given by the base classifiers to each possible tag: For each tag $s_j \in (s_1, \ldots, s_k)$, $support[j] = 0$.
   3.2 Apply the Bayes equation: For each tag $s_j \in (s_1, \ldots, s_k)$ calculate the $support$ function considering $r_p[i]$ as the tag assigned to word $t[i]$ by classifier $c_p$ and $Z_j$ being the number of occurrences of tag $s_j$ in $T$: $support[j] = (1/Z_j^{N-1}) \prod_{p=1}^{N} M_p[j, r_p[i]]$
   3.3 Select the most supported tag and assign it to the current word: $t[i] = \arg\max_j support[j]$
```

Fig. 3. Bayes combination algorithm.

```
Behavior Knowledge Space:
1. Create base classifiers $(c_1, \ldots, c_N)$ using the tagged training corpus $T$.
2. Use cross validation with the base classifiers to tag the training corpus $T$ and build the BKS matrix: For each word $T[i]$ in the tagged training corpus $T$,
   2.1 Calculate the combination of tags proposed by the base classifiers denoted by $x_i$.
   2.2 $BKS_S[x_i] = BKS_S[x_i] + 1$
   2.3 If the true tag of $T[i]$ is $s_j$, then $BKS_n[x_i, s_j] = BKS_n[x_i, s_j] + 1$
   2.4 $BKS_R[x_i] = \arg\max_j BKS_n[x_i, s_j]$
3. Apply the BKS matrix to new data: For each word $t[i]$ in the untagged test corpus $t$,
   3.1 Calculate the combination of tags proposed by the base classifiers denoted by $x_i$.
   3.2 Select the most representative tag associated with $x_i$ in the BKS matrix and assign it to the current word: $t[i] = BKS_R[x_i]$
```

Fig. 4. Behavior knowledge space combination algorithm.

```
Bagging:
1. Create $M$ versions of the tagged training corpus $T$ using sampling with replacement $(T_1, \ldots, T_M)$.
2. Train the $N$ base classifiers with the $M$ versions of $T$ and use the $M \cdot N$ final classifiers to tag the test corpus $t$.
3. Use Voting (or other combination method) to select the final tag among those proposed by the different versions of the base classifiers.
```

Fig. 5. Bagging combination algorithm.

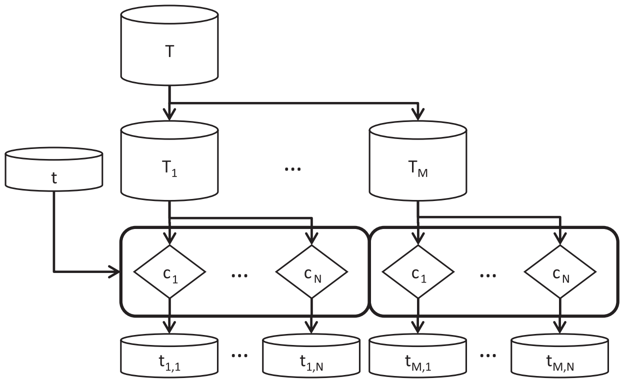

Fig. 6. Execution scheme of “multiple” Bagging (BAG). We generate different versions of the training corpus T using sampling with replacement. These versions $(T_1, \ldots, T_M)$ are used by the base classifiers $(c_1, \ldots, c_N)$ to classify the words in the test corpus t.

The results obtained with 30 iterations (see Table 4), show better results with BAGw than with BAG. The reason may be that sampling the corpus at the sentence level may not create enough variability as the database sampling does. The database is made out of vectors representing words, but independently including their context so the learning method can merge different sentences treating each example in isolation. In the BAG scheme we had to keep the sentences as in the original corpus because the context is also important to decide the tag for a particular word. The improvements shown in the table are calculated as always in relation to the best base classifier. The difference between bagging and the other methods is that FV has been excluded as we said before, so the baseline in this case is the best among TnT, MBT and TreeTagger.

Bagging has been implemented not as an independent method, but as a modifier of the classifiers involved. This option will create different versions of the taggers that can be used as input for any other combination method. In the original bagging algorithm we are supposed to use simple voting, but in our implementation this is a free choice, making it possible to use any other combination method. If we apply stacking rather than voting we get the results shown in Table 4 under the ‘BAG-SG’ column, which are slightly better than the commonly applied method.

3.3.5. Stacking

Stacking is the most popular combination method based on meta-learning. It uses the results of the base classifiers to generate a new database on which to apply a second level learning algorithm. The implementation that has been carried out corresponds to the original Stacked Generalization method introduced by Wolpert [38] (see Fig. 7). This method allows you to maximize the exploitation of the information contained in the training corpus. The different participant classifiers are trained and executed on different partitions of the training corpus to form the training database composed by a set of tag vectors, and finally on the test corpus to generate the test database. Once trained, the second level learning algorithm ultimately determines the tag to be selected among those proposed by the base classifiers in the test database. In our practical implementation we have generated the databases in the ARFF format and used the Weka tool for the second level learning. Fig. 8 shows a summary of this procedure.

The values obtained combining the four base classifiers are shown in Table 4. The results provided by this method are very good. The second level of classification seems to be capable of recognizing and successfully classifying patterns in which the correct tag is in inferiority.

3.3.6. Feature selection

Another point in the classification process where variability can be generated is when the feature vectors that represent the examples in the database are generated. We have carried out two types of experiments in order to cover this kind of methods.

The first approach involves the implementation of the Random Subspace Method. It has not been implemented as a separate method, but as a modifier applicable to the FV classifier, selecting subsets of features randomly. We have generated thirty versions of the FV classifier by selecting the features to be considered in each version with a probability of 50% each one. The results obtained by this method do not improve in any case those obtained using the FV classifier with all the features, loosing 0.39% points of accuracy on average. The explanation resides, from our point of view, in the correlation between different types of features that makes removing random elements of different groups reduce the likelihood of extracting knowledge from data by combination methods. Also the big number of features involved may require a big number of subsets to be considered making the system very inefficient. This belief led us to the second way of experimenting with this type of combination.

The second approach is to make natural groupings of the features used by the FV classifier and then proceed to their combination. We applied the algorithm shown in Fig. 9 to the following versions of the FV classifier:

- FV: Complete version that makes use of all the possible features.
- FVb: Includes all the features except the lexical features.
- FVc: Equivalent to the complete version leaving out the bigrams and trigrams although including the unigrams.
- FVd: Equivalent to the complete version, but removing the information of prefixes an suffixes.

Table 5 shows the results of the four versions of the FV classifier and also the improvements achieved by two types of combination, namely C–A and C–B. The only difference between them is that C–B combines the four versions including the complete version while C–A only combines FVb, FVc and FVd. For the combination we have used the stacking method because of its good performance and robustness. It is important to note that we have obtained benefits using combination with only one base classifier.

Stacking:

1. Create K partitions of the tagged training corpus T and use cross validation to tag these K subsets with all the base classifiers.
2. Build a training database where each word in T generates an instance containing the proposed tags given by the base classifiers for the current word.
3. Apply machine learning to obtain a second level classifier trained on the database created.
4. To tag the new sentences of the test corpus t, use the base classifiers to create the test database with their proposed tags.
5. Execute the second level classifier with the test database and assign the result tags to t.

Fig. 7. Stacking combination algorithm.

3.3.7. Cascading

This method has been implemented in a different manner because it is implicit in the implementation scheme that has been followed for the entire system. We have developed the whole framework through standardized interfaces that allow chaining combiners in different levels. This makes it possible to use the output of a combination method as input to another as if it were a base classifier.

To test this option different schemes have been executed using three levels as explained in Fig. 11. The results of the four base classifiers are used as inputs for several combination methods and the outputs of these are given to a different combination method. Fig. 10 shows the scheme with stacking as the second level combination method (C-SG in Table 6). The other experiments follow the

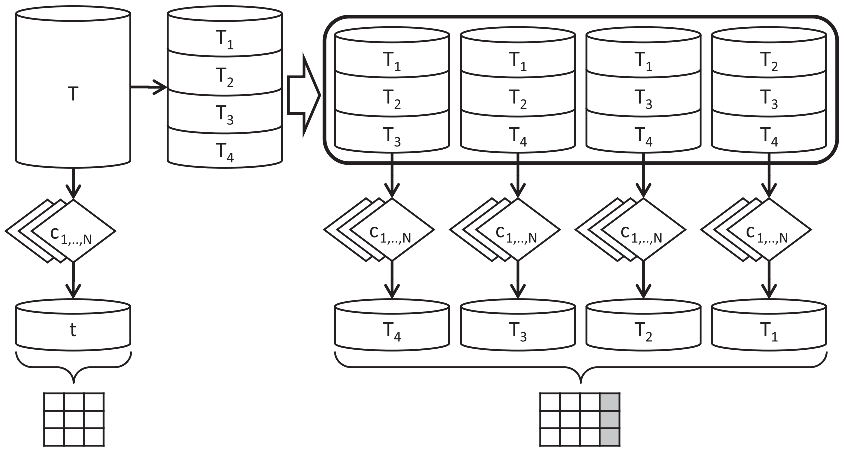

Fig. 8. Stacked Generalization scheme. The training corpus T is divided in K parts ($T_1, \ldots, T_K$), each one being tagged by the classifiers once trained with the remaining partitions. This leads to the training database while the test database is obtained using the entire corpus T for tagging the test corpus t.

Feature subspaces:

1. Create K versions of the feature based classifier using K different feature subsets and train them with the tagged training corpus T.
2. Use the K versions of the base classifier to tag the test corpus t.
3. Use stacking (or any other combination method) to select the final tags for t among those proposed by the K base classifiers.

Fig. 9. Feature selection combination algorithm.

Table 5
Results obtained combining different versions of FV. First, the FV base classifier accuracy results are shown for all its versions. These versions are: complete (FV), without lexical features (FVb), without bigrams and trigrams (FVc) and without prefixes an suffixes (FVd). Finally, the last two columns show the accuracy improvements obtained combining all versions (C–B) or just the reduced ones (C–A).

CORPUS	FV	FVb	FVc	FVd	C–A	C–B
Brown	96.18	95.69	95.89	94.35	0.35	0.36
Conll00Pos	96.41	94.70	96.17	93.87	0.23	0.30
Conll02nedPos	95.01	94.84	94.60	93.11	0.49	0.49
Conll07espPos	95.35	94.82	95.17	91.75	0.03	0.05
Conll07eusPos	91.27	91.41	91.17	84.78	0.12	0.11
Floresta	96.52	95.35	96.27	93.95	0.09	0.11
Susanne	92.26	91.21	91.98	88.90	0.08	0.20
Talp	94.59	94.61	94.42	90.62	0.11	0.20
TreebankwsjPos	96.28	94.70	96.07	95.00	0.19	0.19
Mean	94.87	94.15	94.64	91.81	0.18	0.22

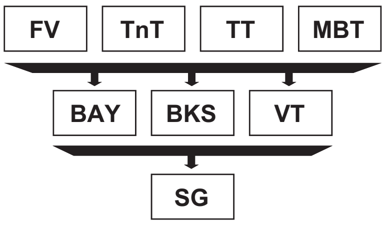

Fig. 10. Cascading execution scheme. There is a two layer combination where different methods combine the base classifier outputs, and their own outputs are combined by a final second level method, which is stacking (SG) in this figure.

Cascading:

1. Train the N base classifiers with the tagged training corpus T and use them to tag the test corpus t.
2. Use K combination methods to select the tags for t among those proposed by the base classifiers, obtaining a new set of K proposed tags in this first level of classification.
3. Use stacking (or any other combination method) to select the final tags for t among those proposed by the combination methods of the first level.

Fig. 11. Cascading combination algorithm.

same pattern changing the method that is placed in the second level.

The results show that improvements are very significant for all schemes that have been tested, exceeding even the best values obtained so far. We also have to highlight the robustness of this system that keeps very good levels of accuracy regardless of the classifier that occupies the second level.

4. Additional experiments with POS tagging

After successfully applying different kinds of combination methods, we wanted to go into a deeper analysis designing some additional experiments. Our objective was to test them with new scenarios and schemes that may be useful in practice, answering interesting questions about what would happen under certain circumstances. From now on we will employ some of the methods that have performed better on the tests conducted in the previous section.

Table 6
Cascading results. The accuracy improvement obtained by the cascading scheme for each corpus is shown. Each column represents the experiments with a different combination method in the second level: Bayes (C-BAY), behavior knowledge space (C-BKS), stacking (C-SG) and voting (C-VT). The final two columns show the average accuracy of the base classifiers (AVG1) and the combination methods (AVG2).

CORPUS	C-BAY	C-BKS	C-SG	C-VT	AVG1	AVG2
Brown	0.69	0.66	0.68	0.67	96.05	97.23
Conll00Pos	0.67	0.67	0.66	0.66	96.74	97.99
Conll02nedPos	0.67	0.66	0.72	0.67	93.49	95.89
Conll02nedPosb	0.59	0.59	0.64	0.56	93.87	96.76
Conll07espPos	1.23	1.23	1.18	1.18	95.45	97.19
Conll07eusPos	0.52	0.52	0.50	0.59	92.43	94.66
Floresta	0.77	0.81	0.73	0.71	96.50	97.78
Susanne	1.33	1.35	1.15	1.52	92.08	94.95
Talp	1.14	1.08	1.12	1.18	95.21	96.95
TreebankwsjPos	0.57	0.56	0.49	0.55	95.92	96.82
Mean	0.84	0.84	0.80	0.86	94.87	96.60

4.1. Eliminating the best

What happens if you remove the best base classifier from the combination scheme? So far we have conducted experiments with the four base classifiers and when we had to pick one of them usually TnT was chosen as it is the one with the best overall results. Now we will do the opposite, removing the best classifier from the system to see if combination continues providing improvements over the individual results of the remaining classifiers. After removing the best participant for each corpus the results shown in Table 7 are obtained. We find out that the improvements are even greater than with all the base classifiers.

Therefore, combination methods can further help when we do not have the best classifiers for the task we are working on. In fact, in Table 8 we see that the combination of the remaining classifiers almost always significantly improves the results given by the best performing classifier available for each corpus. This suggests that it is better to combine several average classifiers than to run only one, even when it is clearly better than the rest.

4.2. The corpus quality

How does the quality of the corpus affect the combination results? To answer this question, we created three additional

Table 7
Results obtained when the best classifier is removed. The accuracy improvements achieved by the combination methods in relation to the best classifier that remains in the system are shown for each corpus. Methods include Bayes (BAY), behavior knowledge space (BKS), stacking (SG), and voting (VT). The last two columns calculate the average accuracies of the base classifiers (AVG1) and combination methods (AVG2) for each corpus.

CORPUS	BAY	BKS	SG	VT	AVG1	AVG2
Brown	0.64	0.86	0.94	0.66	95.88	96.96
Conll00Pos	0.84	0.98	1.11	0.91	96.54	97.76
Conll02nedPos	0.99	1.04	1.00	0.99	92.71	96.39
Conll07espPos	1.51	1.43	1.51	1.22	95.27	96.86
Conll07eusPos	0.28	0.64	0.68	0.29	92.82	94.20
Floresta	0.86	0.98	0.96	0.82	96.33	97.57
Susanne	1.35	2.19	1.95	1.24	91.56	93.94
Talp	0.93	1.09	1.11	0.85	95.00	96.62
TreebankwsjPos	0.07	0.15	0.31	0.08	95.80	96.36
Mean	0.83	1.04	1.06	0.78	94.66	96.29

Table 8
Comparing the best classifier and the combination of the rest. The accuracy improvements achieved by the combination methods in relation to the best classifier available are shown for each corpus. Methods include Bayes (BAY), behavior knowledge space (BKS), stacking (SG), and voting (VT). The best classifier, whose accuracy is shown under the 'BEST-BC' column, did not participate in the combination schemes it is being compared with.

CORPUS	BEST-BC	BAY	BKS	SG	VT
Brown	96.55	0.27	0.49	0.57	0.29
Conll00Pos	97.32	0.32	0.46	0.59	0.39
Conll02nedPos	95.81	0.56	0.61	0.57	0.56
Conll07espPos	95.98	0.97	0.89	0.97	0.68
Conll07eusPos	94.13	−0.12	0.24	0.28	−0.11
Floresta	97.02	0.50	0.62	0.60	0.46
Susanne	93.61	0.00	0.84	0.60	−0.11
Talp	95.82	0.73	0.89	0.91	0.65
TreebankwsjPos	96.28	0.00	0.08	0.24	0.01
Mean	95.84	0.36	0.57	0.59	0.31

Table 9
Results with reduced versions of the Penn Treebank corpus. The accuracy improvements achieved by the combination methods in relation to the best classifier are shown for each corpus. Methods include Bayes (BAY), behavior knowledge space (BKS), stacking (SG), voting (VT), and stacking without the FV classifier (SG″). The last two columns calculate the average accuracies of the base classifiers (AVG1) and combination methods (AVG2) for each corpus.

CORPUS	BAY	BKS	SG	VT	SG″	AVG1	AVG2
TreebankwsjPos	0.27	0.47	0.59	0.45	0.31	95.92	96.75
TreebankwsjPos200k	0.33	0.57	0.56	0.47	0.84	94.83	95.97
TreebankwsjPos100k	0.34	0.49	0.65	0.41	1.06	93.73	95.26
TreebankwsjPos50k	0.25	0.22	0.73	0.36	1.22	92.42	94.20
Mean	0.30	0.44	0.63	0.42	0.86	94.22	95.55

versions the Penn Treebank corpus reducing the number of words it contains. Thus we have the following corpora:

- treebankwsjPos: complete corpus with 766,463 words.
- treebankwsjPos200k: reduced version with 198,550 words.
- treebankwsjPos100k: reduced version with 95,924 words.
- treebankwsjPos50k: reduced version with 47,739 words.

After running different combination methods, results in Table 9 are obtained. Even in the shorter version we are still getting improvements over the base classifiers, although they provide many more errors to the system due to the lack of training data. It is also interesting how stacking can get greater improvements with the smaller corpus than the rest. This suggests that while base classifiers suffer from the low quality data, the results of combination and especially stacking do not decrease in the same amount, being more robust. We also show the stacking results without the best classifier (SG″), which in this case is FV, achieving greater improvements as expected after the previous section.

Table 10
Results of stacking using lexical information. Accuracy improvements of the stacking scheme are shown with and without the lexical information being part of the learning database.

CORPUS	SG	SG-lex
Brown	0.64	0.72
Conll00Pos	0.71	0.77
Conll02nedPos	0.66	0.70
Conll07espPos	1.34	1.48
Conll07eusPos	0.49	0.51
Floresta	0.78	0.79
Susanne	1.26	1.45
Talp	1.10	1.10
TreebankwsjPos	0.59	0.67
Mean	0.84	0.91

4.3. Heterogeneous information

In this section we will try to exploit a stacking ability, which is to integrate heterogeneous information in the learning process extracting knowledge from different sources. This phase of testing consists of several schemes that generate the stacking database with the outputs of the base classifiers along with other values of different types.

A first approach would be to add lexical information about the words in the corpus. We used the following regular expressions to detect lexical features:

- Lowercase: `^[a–z]+$`
- 1st-Uppercase: `^[A–Z].*$`
- Uppercase: `^[A–Z]+$`
- Abbreviation: `^[A–Z].*\.$`
- Acronym: `^[A–Z]\.([A–Z]\.)+$`
- Number: `^[0–9]+$` `^[0–9]+[,.][0–9]+$`
- Range: `^[0–9]+"-"[0–9]+$`
- Quantity: `^[0–9]+[,.]?[0–9]+[%$]$`
- 4-Numbers: `^[0–9][0–9][0–9][0–9]$`
- Hour: `^[0–9][0–9]:[0–9][0–9]$`
- Punctuation: `^[\-.,;\!?"/()\[\]{}]+$`

The results of stacking with and without lexical information are shown in Table 10. It can be verified that these results, which were initially good, are indeed better with the new information entered.

Another approach is to add tags to the database obtained using different corpora, with equal or even different tagsets that may come from other tasks. To perform these experiments we implemented the possibility of specifying a list of classifiers along with the corpus with whom we wish them to be trained before being executed on the target corpus. Thus, in addition to receive the usual tags obtained by the current training corpus, we also receive the tags provided by the classifiers using other corpora, where these classifiers may be the same or different. In Fig. 12 we see the execution scheme where *T* is the training corpus, *t* the test corpus and c_i the base classifiers used, while the T_j are the training corpora that provide the heterogeneous information. The additional

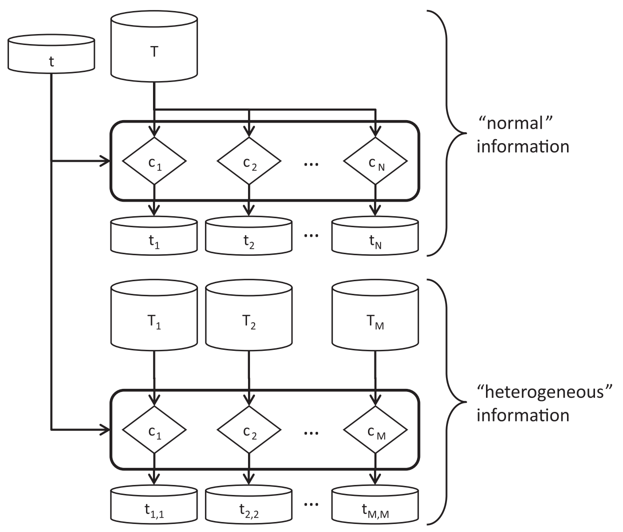

Fig. 12. Stacking using heterogeneous information. In addition to the *N* base classifier opinions, obtained using the original training corpus *T*, we make use of *M* other training resources that can give us additional information. These can be resources of related tasks or maybe the same task, but using different although similar languages for example. Any information that can help the learning algorithm find different patterns can be helpful improving the final results.

Table 11
Results of heterogeneous combination. First, only one base classifier is used (TnT) while different heterogeneous sources are introducing additional information to the system. In 'SG-H1' we include more corpus of the same task, although with different tagsets and sizes. In 'SG-H2' we include even more information adding more corpora from other tasks. Second, previous schemes are repeated using three base classifiers (TnT, TreeTagger and MBT) instead of just TnT ('SGm-H1' and 'SGm-H2'). All three base classifiers are executed with every corpus. Finally, we repeat the experiments once more, but adding lexical information to the learning database. The results obtained are shown under the 'SGmL-H1' and 'SGmL-H2' columns.

	Baseline	1 Classifier		Baseline	3 Classifiers		Adding lexical info.	
CORPUS	(TnT)	SG-H1	SG-H2	(SG)	SGm-H1	SGmH2	SGmL-H1	SGmL-H2
Brown	96.55	0.21	0.25	96.89	0.07	0.11	0.10	0.13
Conll00Pos	97.32	0.39	0.42	97.81	0.02	0.02	0.12	0.05
TreebankwsjPos	96.21	0.35	0.43	96.52	0.14	0.13	0.17	0.18
TreebankwsjPos200k	95.48	0.70	0.70	95.74	0.47	0.59	0.62	0.62
TreebankwsjPos100k	94.82	1.11	1.16	95.14	0.79	0.89	0.79	0.82
TreebankwsjPos50k	93.88	1.79	1.80	94.13	1.44	1.49	1.63	1.63
Mean	95.71	0.76	0.79	96.04	0.49	0.54	0.57	0.57

corpora generate the classifiers that, once executed with the *t* corpus, provide the additional tags.

Following this idea we started using the TnT classifier and carrying out the next experiments:

- SG-H1: Using the Brown, CoNLL 2000 and Penn Treebank corpora, taking in each case one as the target corpus and the other two as heterogeneous information sources.
- SG-H2: The following sources are added to the H1 scheme:
  - The Susanne corpus.
  - The BioCreAtIvE[12] (Critical Assessment of Information Extraction Systems in Biology) corpus that comes from the National Library of Medicine of the United States and contains the POS tag among others.
  - The CoNLL 2000 Chunking corpus.
  - A portion of the IE-ER (Information Extraction: Entity Recognition Evaluation)[13] NER corpus from the NIST 1999. In particular we have worked with the data made available to researchers for testing and developing their systems.

Results appear in Table 11, which also shows the results of TnT as baseline.

After confirming that adding heterogeneous information was beneficial we repeated the same experiments, but this time using more base classifiers. We combined TnT, TreeTagger and MBT, obtaining the results shown in Table 11 under the columns 'SGm-H1' and 'SGm-H2'. This table also shows the improvements obtained by stacking without adding the heterogeneous information, which can be considered the baseline for these experiments.

Finally, we added the lexical information as explained earlier in this section, leading to the 'SGmL-H1' and 'SGmL-H2' experiments shown in Table 11.

The results show that stacking can take advantage of the information that we introduce in the database resulting in a noticeable performance improvement. Furthermore, when we repeat the same experiments on smaller versions of the Penn Treebank we see the importance of this schemes, because the benefits are even greater when the original corpus is small (see Table 11).

5. Applying combination to other NLP tasks

After verifying the usefulness of combination methods with the POS tagging task, we now repeat some experiments with other sequential tagging tasks such as NER, Bio-NER, and Chunking. The objective is to confirm the ability to improve the individual classifiers in other tasks. This also allows us to evaluate the methods with problems for which the base classifiers were not optimized (except for FV that has not been optimized for any particular task). In addition to the recognition accuracy, the measure we use is the $F_{\beta=1}$ value (see Eq. (2)), as it takes into account both precision and coverage while detecting words that are part of an entity or a chunk. This is the most commonly used measure in the literature for these tasks.

$$F_{\beta=1} = \frac{(\beta^2 + 1) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{2}$$

5.1. Corpora

We begin as we did with the POS section with the description of the corpora used, showing their main features in Table 12.

- Coling 2004: This corpus was created for the task evaluated in the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications.[14] Contains over 2000 MEDLINE abstracts extracted from the GENIA corpus of biomedical entities, which in turn is a product of the project of the same name, related to the Information Mobility Project (CREST, JST) and the Genome Information Science Project (MEXT). From the GENIA corpus only five classes of entities were left, which are: DNA, RNA, protein, cell_line and cell_type, tagged using the IOB tagging format with one word per line.
- CoNLL 2000: As discussed in the POS tagging section, this corpus is the one used in the CoNLL competition in its 2000 edition, dedicated to the Chunking task. In this case, we use the Chunking tags column ignoring the POS tags.
- CoNLL 2002: The CoNLL 2002 corpus is a corpus developed for the NER task in two languages, Spanish and Dutch. It consists of two columns separated by a space, where the first contains the words and the second the NER tag in IOB format. The entities considered are four: names of persons (PER), organizations (ORG), locations (LOC) and "miscellaneous" (MISC). Data comes from the Spanish news agency EFE on May 2000 and was tagged by members of the Polytechnic University of Catalonia and the University of Barcelona. The Dutch part consists of four editions of the Belgian newspaper "De Morgen" of 2000 and tagging was carried out by members of the University of Antwerp.

5.2. Applying combination methods

Table 13 shows the results obtained with some of the combination methods that have performed best for the POS task. Labels cor-

[12] http://www.mitre.org/public/biocreative/.
[13] http://www.itl.nist.gov/iad/894.01/tests/i.e.-er/er_99/er_99.htm.
[14] http://www.genisis.ch/~natlang/JNLPBA04/.

Table 12
Summary of the NER and Chunking corpora used. In this table we show several characteristics of the corpora used in our experiments with other NLP tasks different from POS tagging.

CORPUS	Language	Task	Tags	TRAIN		TEST	
				Sentences	Words	Sentences	Words
Coling'04	ENG	BIO	11	18,546	492,551	3856	101,039
CoNLL'00	ENG	CHK	23	8936	211,727	2012	47,377
CoNLL'02	SPA	NER	9	8323	264,715	1915	52,923
	DUT	NER	9	15,806	202,931	2895	37,761

Table 13
Combination results on NER, Bio-NER and Chunking. The accuracy and $F_{\beta=1}$ improvements for several combination methods are shown. The 'AVG' column calculates the average results obtained by the base classifiers for each corpus. The following columns show the improvements in relation to the best base classifier. The combination methods are stacking (SG), stacking using lexical features (SG-l), voting (VT), and four cascading schemes (C-*x*), each one with a different method *x* in the second layer (Bayes, behavior Knowledge space, stacking or voting).

CORPUS	AVG		SG		SG-l		VT		C-BAY		C-BKS		C-SG		C-VT	
	Acc	$F_{\beta=1}$	Acc	$F_{\beta=1}$	Acc	$F_{\beta=1}$	Acc	$F_{\beta=1}$	Acc	$F_{\beta=1}$	Acc	$F_{\beta=1}$	Acc	$F_{\beta=1}$	Acc	$F_{\beta=1}$
Coling04	89.56	57.63	0.19	2.99	0.03	1.75	0.01	0.57	0.10	1.10	0.15	2.66	0.04	2.45	0.08	1.89
Conll00Chunk	90.51	84.12	0.71	1.24	0.84	1.40	0.00	0.00	0.55	0.89	0.70	1.27	0.47	0.88	0.55	0.94
Conll02esp	94.73	64.72	0.39	2.32	0.33	1.18	0.28	1.10	0.42	1.02	0.38	1.04	0.45	2.43	0.45	1.76
Conll02nedNer	95.96	60.82	0.28	4.03	0.27	4.37	0.28	2.72	0.35	4.00	0.36	3.92	0.24	2.40	0.36	4.21
Mean	92.69	66.82	0.39	2.65	0.37	2.18	0.14	1.10	0.36	1.75	0.40	2.22	0.30	2.04	0.36	2.20

respond to stacking (SG), stacking with lexical information (SG-l), voting (VT) and different cascading schemes (C-*x*). The *x* of the latter is the method that occupies the second combination level: Bayes (BAY), behavior knowledge space (BKS), stacking (SG) or voting (VT).

We found that improvements also occur in tasks very different from POS tagging and for which the base classifiers were not optimized. This is a situation that often arises in research, where the availability of optimal tools and resources is not always guaranteed and the combination methods should be seen as a good option to consider. In these tasks, where the elements to classify are often formed by more than one word, individual errors may have some effect on the surrounding words. The difference between accuracy and $F_{\beta=1}$ reflects the fact that one simple tag that is corrected can make a set of words form a correctly classified chunk.

6. Conclusions

Like in many other fields, Natural Language Processing (NLP) research deals with many problems that, due to their complexity, often are divided into separate tasks that once resolved provide a portion of the information needed to achieve the ultimate goals. These tasks are often considered classification problems, consisting of assigning a category to a word, phrase, piece of text or an entire document. As a result we have seen an extensive use of multiple algorithms and classification methods that have been developed in recent years. Progress in resolving these tasks has been made, with very good results in some cases and not so successful in others.

On the other hand, in the area of Machine Learning many researchers have worked hard studying other methods that attempt to exploit the various approaches taken by the classification methods. These are algorithms that combine the categories proposed by the classifiers in order to exploit the virtues of each one of them. Among combination methods we find very popular cases like the voting techniques, and others not so well known, but with very convincing results as shown in published papers.

We have studied the combination methods from different points of view, looking for distinct theoretical aspects and trying to shed light on what we should consider when deciding to combine classifiers in a particular situation. The main contributions can be summarized as follows:

1. Bibliographical analysis: We have carried out an initial bibliographical analysis, briefly explained in the first section, in which we have tried to summarize the historical use of combination methods in NLP. Analyzing a large number of papers that apply combination methods to a NLP task, we found that while the use of classification methods is varied, demonstrating the high level of knowledge about the various alternatives, the use of combination methods is quite biased towards voting and stacking.
2. Comparative study: We have implemented a large number of combination methods and conducted several experiments to test their effectiveness solving the task of POS tagging. This is a very well known task and also very demanding due to the great results obtained by the base classifiers that are combined. We have shown results that support the application of these methods with significant improvements. Some of the experiments were carried out even with only one base classifier, demonstrating that diversity can be created in different manners and combination can be applied in multiple situations. The meta-learning performance was outstanding demonstrating its robustness and flexibility. Both stacking and cascading have obtained very good results in all scenarios. In addition to the POS task, we made experiments with other tasks such as NER, Chuncking, and even Bio-NER with very good results. Note that 90% of the 360 combination experiments carried out have exceeded the values obtained by the base classifiers and in most cases very significantly.
3. Additional experiments: We have completed the comparative study with other experiments that focus on special situations of interest. First we reduced the quality of the classifiers and corpora involved in the combination process. This may be important for those who find themselves with poor resources and cannot afford the cost of tagging a larger corpus or construct better classifiers. Second, we tried to take advantage of one main feature of stacking, that is the ability of incorporating heterogeneous information to the learning process. This allows us to get the best out of all resources obtaining the highest accuracy.

Therefore we believe that combination methods always give us a great opportunity to improve existing systems. The development of tools that take advantage of its virtues can boost the results for classification tasks and by extension, for the NLP problems that are based on them.

References

[1] L. Hansen, P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (10) (1990) 993–1001.

[2] T.G. Dietterich, Ensemble methods in machine learning, in: J. Kittler, F. Roli, (Eds.), Multiple Classifier Systems, Lecture Notes in Computer Science, vol. 1857, 2000, pp. 1–15.

[3] Y. Freund, Y. Mansour, R. Schapire, Why averaging classifiers can protect against overfitting, in: Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics, 2001.

[4] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, 2004.

[5] L. Xu, A. Krzyzak, C.Y. Suen, Methods of combining multiple classifiers and their application to handwriting recognition, IEEE Transactions on Systems, Man, and Cybernetics 22 (1992) 418–435.

[6] N. Belkin, J. Callan, The effect of multiple query representations on information retrieval system performance, in: Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1993, pp. 339–346.

[7] R. Frederking, S. Nirenburg, Three heads are better than one, in: Proceedings of the fourth Conference on Applied Natural Language Processing (ANLP-94), 1994, pp. 95–100.

[8] D. Hull, J. Pedersen, H. Schutze, Method combination for document filtering, SIGIR Forum (ACM Special Interest Group on Information Retrieval) (1996) 279–288.

[9] J. Fiscus, A post-processing system to yield reduced word error rates: recognizer output voting error reduction (rover), in: IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-97), 1997, pp. 347–354.

[10] H. Halteren, J. Zavrel, W. Daelemans, Improving data driven wordclass tagging by system combination, in: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, vol. 1, 1998, pp. 491–497.

[11] E. Brill, J. Wu, Classifier combination for improved lexical disambiguation, in: Proceedings of the 17th International Conference on Computational Linguistics, 1998, pp. 191–195.

[12] L. Marquez, H. Rodriguez, J. Carmona, J. Montolio, Improving pos tagging using machine-learning techniques, in: Proceedings of the 1999 Faint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999, pp. 53–62.

[13] H. Halteren, W. Daelemans, J. Zavrel, Improving accuracy in word class tagging through the combination of machine learning systems, Computational Linguistics 27 (2) (2001) 199–229.

[14] M. Kuta, M. Wrzeszcz, P. Chrza¸szcz, J. Kitowski, Accuracy of baseline and complex methods applied to morphosyntactic tagging of polish, in: ICCS ’08: Proceedings of the 8th International Conference on Computational Science, Part I, 2008, pp. 903–912.

[15] T. Pedersen, A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation, in: Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference, 2000, pp. 63–69.

[16] R. Florian, D. Yarowsky, Modeling consensus: classifier combination for word sense disambiguation, in: EMNLP ’02: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, 2002, pp. 25–32.

[17] L.A. Cuong, A study of classifier combination and semi-supervised learning for word sense disambiguation, Ph.D. thesis, Japan Advanced Institute of Science and Technology, 2007.

[18] R. Florian, A. Ittycheriah, H. Jing, T. Zhang, Named entity recognition through classifier combination, in: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, 2003, pp. 168–171.

[19] Z. Kozareva, O. Ferrández, A. Montoyo, R. Muoz, A. Suárez, J. Gómez, Combining data-driven systems for improving named entity recognition, Data & Knowledge Engineering 61 (3) (2007) 449–466.

[20] H. Wang, T. Zhao, Identifying named entities in biomedical text based on stacked generalization, in: Proceedings of the World Congress on Intelligent Control and Automation (WCICA), 2008, pp. 160–164.

[21] J. Henderson, E. Brill, Bagging and boosting a treebank parser, in: Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference, 2000, pp. 34–41.

[22] D. Zeman, Z. Žabokrtský, Improving parsing accuracy by combining diverse dependency parsers, in: Proceedings of IWPT-2005, 2005, pp. 171–178.

[23] J. Huang, O. Madani, C. Giles, Error-driven generalist+experts (edge): a multi-stage ensemble framework for text categorization, in: CIKM ’08: Proceeding of the 17th ACM Conference on Information and Knowledge Management, 2008, pp. 83–92.

[24] X. Qi, B. Davison, Web page classification: features and algorithms, ACM Computing Surveys (CSUR) 41 (2) (2009) 1–31.

[25] G. Sigletos, G. Paliouras, C. Spyropoulos, M. Hatzopoulos, Combining information extraction systems using voting and stacked generalization, Journal of Machine Learning Research 6 (2005) 1751–1782.

[26] M. Banko, O. Etzioni, The tradeoffs between open and traditional relation extraction, in: Proceedings of ACL-08: HLT, 2008, pp. 28–36.

[27] K. Tsutsumi, K. Shimada, T. Endo, Movie review classification based on a multiple classifier, in: The 21th Pacific Asia Conference on Language, Information and Computation (PACLIC), 2007.

[28] S. Li, C. Zong, X. Wang, Sentiment classification through combining classifiers with multiple feature sets, in: IEEE NLP-KE 2007 – Proceedings of International Conference on Natural Language Processing and Knowledge Engineering, vol. 4368024, 2007, pp. 135–140.

[29] T. Brants, Tnt. A statistical part-of-speech tagger, in: Proceedings of the 6th Applied NLP Conference (ANLP00), 2000, pp. 224–231.

[30] H. Schmid, Probabilistic part-of-speech tagging using decision trees, in: Proceedings of the Conference on New Methods in Language Processing, 1994.

[31] W. Daelemans, J. Zavrel, A. Bosch, K. Sloot, Mbt: memory-based tagger, reference guide, Tech. Rep. 03-13, ILK, 2003.

[32] T. Joachims, Making Large-Scale SVM Learning Practical, MIT Press, 1999 (chapter 11).

[33] K. Arrow, Social Choice and Individual Values, Wiley, New York, 1951.

[34] S. French, Group consensus probability distributions: a critical survey, Bayesian Statistics 2 (1985) 183–202.

[35] Y.S. Huang, C.Y. Suen, A method of combining multiple experts for the recognition of unconstrained handwritten numerals, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (1995) 90–93.

[36] L. Breiman, Bagging predictors, Tech. Rep. 421, Department of Statistics, University of California, Berkeley, 1994.

[37] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufman, 2000.

[38] D. Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259.