Using Machine Learning Techniques to Build a Comma Checker for Basque

Iñaki Alegria Bertol Arrieta Arantza Diaz de Ilarraza Eli Izagirre Montse Maritxalar
Computer Engineering Faculty. University of the Basque Country.
Manuel de Lardizabal Pasealekua, 1
20018 Donostia, Basque Country, Spain.
{acpalloi,bertol,jipdisaa,jibizole,jipmaanm}@ehu.es

Abstract

In this paper, we describe the research using machine learning techniques to build a comma checker to be integrated in a grammar checker for Basque. After several experiments, and trained with a little corpus of 100,000 words, the system guesses correctly not placing commas with a precision of 96% and a recall of 98%. It also gets a precision of 70% and a recall of 49% in the task of placing commas. Finally, we have shown that these results can be improved using a bigger and a more homogeneous corpus to train, that is, a bigger corpus written by one unique author.

1 Introduction

In the last years, there have been many studies aimed at building a grammar checker for the Basque language (Ansa et al., 2004; Diaz De Ilarraza et al., 2005). These works have been focused, mainly, on building rule sets —taking into account syntactic information extracted from the corpus automatically— that detect some erroneous grammar forms. The research here presented wants to complement the earlier work by focusing on the style and the punctuation of the texts. To be precise, we have experimented using machine learning techniques for the special case of the comma, to evaluate their performance and to analyse the possibility of applying it in other tasks of the grammar checker.

However, developing a punctuation checker encounters one problem in particular: the fact that the punctuation rules are not totally established. In general, there is no problem when using the full stop, the question mark or the exclamation mark. Santos (1998) highlights these marks are reliable punctuation marks, while all the rest are unreliable. Errors related to the reliable ones (putting or not the initial question or exclamation mark depending on the language, for instance) are not so hard to treat. A rule set to correct some of these has already been defined for the Basque language (Ansa et al., 2004). In contrast, the comma is the most polyvalent and, thus, the least defined punctuation mark (Bayraktar et al., 1998; Hill and Murray, 1998). The ambiguity of the comma, in fact, has been shown often (Bayraktar et al., 1998; Beeferman et al., 1998; Van Delden S. and Gomez F., 2002). These works have shown the lack of fixed rules about the comma. There are only some intuitive and generally accepted rules, but they are not used in a standard way. In Basque, this problem gets even more evident, since the standardisation and normalisation of the language began only about twenty-five years ago and it has not finished yet. Morphology is mostly defined, but, on the contrary, as far as syntax is concerned, there is quite work to do. In punctuation and style, some basic rules have been defined and accepted by the Basque Language Academy (Zubimendi, 2004). However, there are not final decisions about the case of the comma.

Nevertheless, since Nunberg's monograph (Nunberg, 1990), the importance of the comma has been undeniable, mainly in these two aspects: i) as a due to the syntax of the sentence (Nunberg, 1990; Bayraktar et al., 1998; Garzia, 1997), and ii) as a basis to improve some natural language processing tools (syntactic analysers, error detection tools…), as well as to develop some new ones (Briscoe and Carroll, 1995; Jones, 1996). The relevance of the comma for the syntax of the sentence may be easily proved with some clarifying examples where the sentence is understood in one or other way, depending on whether a comma is placed or not (Nunberg, 1990):

a. Those students who can, contribute to the United Fund.

b. Those students who can contribute to the United Fund.

In the same sense, it is obvious that a well punctuated text, or more concretely, a correct placement of the commas, would help considerably in the automatic syntactic analysis of the sentence, and, therefore, in the development of more and better tools in the NLP field. Say and Akman (1997) summarise the research efforts in this direction.

As an important background for our work, we note where the linguistic information on the comma for the Basque language was formalised. This information was extracted after analysing the theories of some experts in Basque syntax and punctuation (Aldezabal et al., 2003). In fact, although no final decisions have been taken by the Basque Language Academy yet, the theory formalised in the above mentioned work has succeeded in unifying the main points of view about the punctuation in Basque. Obviously, this has been the basis for our work.

2 Learning commas

We have designed two different but combinable ways to get the comma checker:

- based on clause boundaries
- based directly on corpus

Bearing in mind the formalised theory of Aldezabal et al. (2003)[1], we realised that if we got to split the sentence into clauses, it would be quite easy to develop rules for detecting the exact places where commas would have to go. Thus, the best way to build a comma checker would be to get, first, a clause identification tool.

Recent papers in this area report quite good results using machine learning techniques. Carreras and Màrquez (2003) get one of the best performances in this task (84.36% in test). Therefore, we decided to adopt this as a basis in order to get an automatic clause splitting tool for Basque. But as it is known, machine learning techniques cannot be applied if no training corpus is available, and one year ago, when we started this process, Basque texts with this tagged clause splits were not available.

Therefore, we decided to use the second alternative. We had available some corpora of Basque, and we decided to try learning commas from raw text, since a previous tagging was not needed. The problem with the raw text is that its commas are not the result of applying consistent rules.

[1] From now on, we will speak about this as "the accepted theory of Basque punctuation".

Related work

Machine learning techniques have been applied in many fields and for many purposes, but we have found only one reference in the literature related to the use of machine learning techniques to assign commas automatically.

Hardt (2001) describes research in using the Brill tagger (Brill 1994; Brill, 1995) to learn to identify incorrect commas in Danish. The system was developed by randomly inserting commas in a text, which were tagged as incorrect, while the original commas were tagged as correct. This system identifies incorrect commas with a precision of 91% and a recall of 77%, but Hardt (2001) does not mention anything about identifying correct commas.

In our proposal, we have tried to carry out both aspects, taking as a basis other works that also use machine learning techniques in similar problems such as clause splitting (Tjong Kim Sang E.F. and Déjean H., 2001) or detection of chunks (Tjong Kim Sang E.F. and Buchholz S., 2000).

3 Experimental setup

Corpora

As we have mentioned before, some corpora in Basque are available. Therefore, our first task was to select the training corpora, taking into account that well punctuated corpora were needed to train the machine correctly. For that purpose, we looked for corpora that satisfied as much as possible our "accepted theory of Basque punctuation". The corpora of the unique newspaper written in Basque, called *Egunkaria* (nowadays *Berria*), were chosen, since they are supposed to use the "accepted theory of Basque punctuation". Nevertheless, after some brief verifications, we realised that the texts of the corpora do not fully match with our theory. This can be understood considering that a lot of people work in a newspaper. That is, every journalist can use his own interpretation of the "accepted theory", even if all of them were instructed to use it in the same way. Therefore, doing this research, we had in mind that the results we would get were not going to be perfect.

To counteract this problem, we also collected more homogeneous corpora from prestigious writers: a translation of a book of philosophy and a novel. Details about these corpora are shown in Table 1.

	Size of the corpora
Corpora from the newspaper *Egunkaria*	420,000 words
Philosophy texts written by one unique author	25,000 words
Literature texts written by one unique author	25,000 words

Table 1. Dimensions of the used corpora

A short version of the first corpus was used in different experiments in order to tune the system (see section 4). The differences between the results depending on the type of the corpora are shown in section 5.

Evaluation

Results are shown using the standard measures in this area: precision, recall and f-measure[2], which are calculated based on the test corpus. The results are shown in two colums ("0" and "1") that correspond to the result categories used. The results for the column "0" are the ones for the instances that are not followed by a comma. On the contrary, the results for the column "1" are the results for the instances that should be followed by a comma.

Since our final goal is to build a comma checker, the precision in the column "1" is the most important data for us, although the recall for the same column is also relevant. In this kind of tools, the most important thing is to first obtain all the comma proposals right (precision in columns "1"), and then to obtain all the possible commas (recall in columns "1").

Baselines

In the beginning, we calculated two possible baselines based on a big part of the newspaper corpora in order to choose the best one.

The first one was based on the number of commas that appeared in these texts. In other words, we calculated how many commas appeared in the corpora (8% out of all words), and then we put commas randomly in this proportion in the test corpus. The results obtained were not very good (see Table 2, baseline1), especially for the instances "followed by a comma" (column "1").

The second baseline was developed using the list of words appearing before a comma in the training corpora. In the test corpus, a word was tagged as "followed by a comma" if it was one of the words of the mentioned list. The results (see baseline 2, in Table 2) were better, in this case, for the instances followed by a comma (column named "1"). But, on the contrary, baseline 1 provided us with better results for the instances not followed by a comma (column named "0"). That is why we decided to take, as our baseline, the best data offered by each baseline (the ones in bold in table 2).

	0			1		
	Prec.	Rec.	Meas.	Prec.	Rec.	Meas.
baseline 1	**0.927**	**0.924**	**0.926**	0.076	0.079	0.078
baseline 2	0.946	0.556	0.700	**0,096**	**0.596**	**0.165**

Table 2: The baselines

Methods and attributes

We use the WEKA[3] implementation of these classifiers: the Naive Bayes based classifier (NaiveBayes), the support vector machine based classifier (SMO) and the decision-tree (C4.5) based one (j48).

It has to be pointed out that commas were taken away from the original corpora. At the same time, for each token, we stored whether it was followed by a comma or not. That is, for each word (token), it was stored whether a comma was placed next to it or not. Therefore, each token in the corpus is equivalent to an example (an instance). The attributes of each token are based on the token itself and some surrounding ones. The application window describes the number of tokens considered as information for each token.

Our initial application window was [-5, +5]; that means we took into account the previous and following 5 words (with their corresponding attributes) as valid information for each word. However, we tuned the system with different application windows (see section 4).

Nevertheless, the attributes managed for each word can be as complex as we want. We could only use words, but we thought some morphosyntactic information would be beneficial for the machine to learn. Hence, we decided to include as much information as we could extract using the shallow syntactic parser of Basque (Aduriz et al., 2004). This parser uses the tokeniser, the lemmatiser, the chunker and the morphosyntactic disambiguator developed by the IXA[4] research group.

The attributes we chose to use for each token were the following:

- word-form
- lemma
- category
- subcategory
- declension case
- subordinate-clause type

[2] f-measure = 2*precision*recall / (precision+recall)

[3] WEKA is a collection of machine learning algorithms for data mining tasks (http://www.cs.waikato.ac.nz/ml/weka/).

[4] http://ixa.si.ehu.es

- beginning of chunk (verb, nominal, entity, postposition)
- end of chunk (verb, nominal, entity, postposition)
- part of an apposition
- other binary features: multiple word token, full stop, suspension points, colon, semicolon, exclamation mark and question mark

We also included some additional attributes which were automatically calculated:

- number of verb chunks to the beginning and to the end of the sentence
- number of nominal chunks to the beginning and to the end of the sentence
- number of subordinate-clause marks to the beginning and to the end of the sentence
- distance (in tokens) to the beginning and to the end of the sentence

We also did other experiments using binary attributes that correspond to most used collocations (see section 4).

Besides, we used the result attribute "comma" to store whether a comma was placed after each token.

4 Experiments

Dimension of the corpus

In this test, we employed the attributes described in section 3 and an initial window of [-5, +5], which means we took into account the previous 5 tokens and the following 5. We also used the C4.5 algorithm initially, since this algorithm gets very good results in other similar machine learning tasks related to the surface syntax (Alegria et al., 2004).

	0			1		
	Prec.	Rec.	Meas.	Prec.	Rec.	Meas.
100,000 train / 30,000 test	0,955	0,981	0,968	0,635	0,417	0,503
160,000 train / 45,000 test	0,947	0,981	0,964	0,687	0,43	0,529
330,000 train / 90,000 test	**0,96**	**0,982**	**0,971**	**0,701**	**0,504**	**0,587**

Table 3. Results depending on the size of corpora (C4.5 algorithm; [-5,+5] window).

As it can be seen in table 3, the bigger the corpus, the better the results, but logically, the time expended to obtain the results also increases considerably. That is why we chose the smallest corpus for doing the remaining tests (100,000 words to train and 30,000 words to test). We thought that the size of this corpus was enough to get good comparative results. This test, anyway, suggested that the best results we could obtain would be always improvable using more and more corpora.

Selecting the window

Using the corpus and the attributes described before, we did some tests to decide the best application window. As we have already mentioned, in some problems of this type, the information of the surrounding words may contain important data to decide the result of the current word.

In this test, we wanted to decide the best application window for our problem.

	0			1		
	Prec.	Rec.	Meas.	Prec.	Rec.	Meas.
-5+5	0,955	0,981	0,968	0,635	0,417	0,503
-2+5	0,956	0,982	0,969	0,648	0,431	0,518
-3+5	0,957	0,979	0,968	0,627	0,441	0,518
-4+5	0,957	0,98	0,968	0,634	0,446	**0,52**
-5+2	0,956	0,982	0,969	**0,65**	0,424	0,514
-5+3	0,956	0,981	0,969	0,643	0,432	0,517
-5+4	0,955	0,982	0,968	0,64	0,417	0,505
-6+2	0,956	0,982	0,969	0,645	0,421	0,509
-6+3	0,956	0,982	0,969	0,646	0,426	0,514
-8+2	0,956	0,982	0,969	0,645	0,425	0,513
-8+3	0,956	0,979	0,967	0,615	0,431	0,507
-8+8	0,956	0,978	0,967	0,604	0,422	0,497

Table 4. Results depending on the application window (C4.5 algorithm; 100,000 train / 30,000 test)

As it can be seen, the best f-measure for the instances followed by a comma was obtained using the application window [-4,+5]. However, as we have said before, we are more interested in the precision. Thus, the application window [-5 ,+2] gets the best precision, and, besides, its f-measure is almost the same as the best one. This is the reason why we decided to choose the [-5 ,+2] application window.

Selecting the classifier

With the selected attributes, the corpus of 130,000 words and the application window of [-5 , +2], the next step was to select the best classifier for our problem. We tried the WEKA implementation of these classifiers: the Naive Bayes based classifier (NaiveBayes), the support vector machine based classifier (SMO) and the decision tree based one (j48). Table 5 shows the results obtained:

	0			1		
	Prec.	Rec.	Meas.	Prec.	Rec.	Meas.
NB	0,948	0,956	0,952	0,376	0,335	0,355
SMO	0,936	0,994	0,965	**0,672**	0,143	0,236
J48	0,956	0,982	0,969	0,652	0,424	**0,514**

Table 5. Results depending on the classifier (100,000 train / 30,000 test; [-5, +2] window).

As we can see, the f-measure for the instances not followed by a comma (column "0") is almost the same for the three classifiers, but, on the contrary, there is a considerable difference when we refer to the instances followed by a comma (column "1"). The best f-measure gives the C4.5 based classifier (J48) due to the better recall, although the best precision is for the support vector machine based classifier (SMO). Definitively, the Naïve Bayes (NB) based classifier was discarded, but we had to think about the final goal of our research to choose between the other two classifiers. Since our final goal was to build a comma checker, we would have to have chosen the classifier that gave us the best precision, that is, the support vector machine based one. But the recall of the support vector machine based classifier was not as good as expected to be selected. Consequently, we decided to choose the C4.5 based classifier.

Selecting examples

At this moment, the results we get seem to be quite good for the instances not followed by a comma, but not so good for the instances that should follow a comma. This could be explained by the fact that we have no balanced training corpus. In other words, in a normal text, there are a lot of instances not followed by a comma, but there are not so many followed by it. Thus, our training corpus, logically, has very different amounts of instances followed by a comma and not followed by a comma. That is the reason why the system will learn more easily to avoid the unnecessary commas than placing the necessary ones.

Therefore, we resolved to train the system with a corpus where the number of instances followed by a comma and not followed by a comma was the same. For that purpose, we prepared a *perl* program that changed the initial corpus, and saved only x words for each word followed by a comma.

In table 6, we can see the obtained results. One to one means that in that case, the training corpus had one instance not followed by a comma, for each instance followed by a comma. On the other hand, one to two means that the training corpus had two instances not followed by a comma for each word followed by a comma, and so on.

	0			1		
	Prec.	Rec.	Meas.	Prec.	Rec.	Meas.
normal	0,955	0,981	0,968	**0,635**	0,417	0,503
one to one	0,989	0,633	0,772	0,164	**0,912**	0,277
one to two	0,977	0,902	0,938	0,367	0,725	0,487
one to three	0,969	0,934	0,951	0,427	0,621	0,506
one to four	0,966	0,952	0,959	0,484	0,575	0,526
one to five	0,966	0,961	0,963	0,534	0,568	**0,55**
one to six	0,963	0,966	0,964	0,55	0,524	0,537

Table 6. Results depending on the number of words kept for each comma (C4.5 algorithm; 100,000 train / 30,000 test; [-5, +2] window).

As observed in the previous table, the best precision in the case of the instances followed by a comma is the original one: the training corpus where no instances were removed. Note that these results are referred as *normal* in table 6.

The corpus where a unique instance not followed by a comma is kept for each instance followed by a comma gets the best recall results, but the precision decreases notably.

The best f-measure for the instances that should be followed by a comma is obtained by the one to five scheme, but as mentioned before, a comma checker must take care of offering correct comma proposals. In other words, as the precision of the original corpus is quite better (ten points better), we decided to continue our work with the first choice: the corpus where no instances were removed.

Adding new attributes

Keeping the best results obtained in the tests described above (C4.5 with the [-5, +2] window, and not removing any "not comma" instances), we thought that giving importance to the words that appear normally before the comma would increase our results. Therefore, we did the following tests:

1) To search a big corpus in order to extract the most frequent one hundred words that precede a comma, the most frequent one hundred pairs of words (bigrams) that precede a comma, and the most frequent one hundred sets of three words (trigrams) that precede a comma, and use them as attributes in the learning process.

2) To use only three attributes instead of the mentioned three hundred to encode the information about preceding words. The first attribute would indicate whether a word is or not one of

the most frequent one hundred words. The second attribute would mean whether a word is or not the last part of one of the most frequent one hundred pairs of words. And the third attribute would mean whether a word is or not the last part of one of the most frequent one hundred sets of three words.

3) The case (1), but with a little difference: removing the attributes "word" and "lemma" of each instance.

	0			1		
	Prec.	Rec.	Meas.	Prec.	Rec.	Meas.
(0): normal	0,956	0,982	0,969	0,652	0,424	0,514
(1): 300 attributes	**0,96**	**0,983**	**0,972**	**0,696**	**0,486**	**0,572**
(2): 3 attributes +	0,96	0,981	0,97	0,665	0,481	0,558
(3): 300 attributes +, no lemma, no word	0,955	0,987	0,971	**0,71**	0,406	0,517

Table 7. Results depending on the new attributes used (C4.5 algorithm; 100,000 train / 30,000 test; [-5, +2] window; not removed instances).

Table 7 shows that case number 1 (putting the 300 data as attributes) improves the precision of putting commas (column "1") in more than 4 points. Besides, it also improves the recall, and, thus, we improve almost 6 points its f-measure.

The third test gives the best precision, but the recall decreases considerably. Hence, we decided to choose the case number 1, in table 7.

5 Effect of the corpus type

As we have said before (see section 3), depending on the quality of the texts, the results could be different.

In table 8, we can see the results using the different types of corpus described in table 1. Obviously, to give a correct comparison, we have used the same size for all the corpora (20,000 instances to train and 5,000 instances to test, which is the maximum size we have been able to acquire for the three mentioned corpora).

	0			1		
	Prec.	Rec.	Meas.	Prec.	Rec.	Meas.
Newspaper	0.923	0.977	0.949	0.445	0.188	0.264
Philosophy	0.932	0.961	0.946	**0.583**	**0.44**	**0.501**
Literature	0.925	0.976	0.95	0.53	0.259	0.348

Table 8. Results depending on the type of corpora (20,000 train / 5,000 test).

The first line shows the results obtained using the short version of the newspaper. The second line describes the results obtained using the translation of a book of philosophy, written completely by one author. And the third one presents the results obtained using a novel written in Basque.

In any case, the results prove that our hypothesis was correct. Using texts written by a unique author improves the results. The book of philosophy has the best precision and the best recall. It could be because it has very long sentences and because philosophical texts use a stricter syntax comparing with the free style of a literature writer.

As it was impossible for us to collect the necessary amount of unique author corpora, we could not go further in our tests.

6 Conclusions and future work

We have used machine learning techniques for the task of placing commas automatically in texts. As far as we know, it is quite a novel application field. Hardt (2001) described a system which identified incorrect commas with a precision of 91% and a recall of 77% (using 600,000 words to train). These results are comparable with the ones we obtain for the task of guessing correctly when not to place commas (see column "0" in the tables). Using 100,000 words to train, we obtain 96% of precision and 98.3% of recall. The main reason could be that we use more information to learn.

However, we have not obtained as good results as we hoped in the task of placing commas (we get a precision of 69.6% and a recall of 48.6%). Nevertheless, in this particular task, we have improved considerably with the designed tests, and more improvements could be obtained using more corpora and more specific corpora as texts written by a unique author or by using scientific texts.

Moreover, we have detected some possible problems that could have brought these regular results in the mentioned task:

- No fixed rules for commas in the Basque language
- Negative influence when training using corpora from different writers

In this sense, we have carried out a little experiment with some English corpora. Our hypothesis was that a completely settled language like English, where comma rules are more or less fixed, would obtain better results. Taking a comparative English corpus[5] and similar learning attributes[6] to Basque's one, we got, for the instances followed by a comma (column "1" in tables), a better precision (%83.3) than the best

[5] A newspaper corpus, from Reuters

[6] Linguistic information obtained using Freeling (http://garraf.epsevg.upc.es/freeling/)

one obtained for the Basque language. However, the recall was worse than ours: %38.7. We have to take into account that we used less learning attributes with the English corpus and that we did not change the application window chosen for the Basque experiment. Another application window would have been probably more suitable for English. Therefore, we believe that with a few tests we easily would achieve a better recall. These results, anyway, confirm our hypothesis and our diagnosis of the detected problems.

Nevertheless, we think the presented results for the Basque language could be improved. One way would be to use "information gain" techniques in order to carry out the feature selection. On the other hand, we think that more syntactic information, concretely clause splits tags, would be especially beneficial to detect those commas named delimiters by Nunberg (1990).

In fact, our main future research will consist on clause identification. Based on the "accepted theory of the comma", we can assure that a good identification of clauses (together with some significant linguistic information we already have) would enable us to put commas correctly in any text, just implementing some simple rules. Besides, a combination of both methods —learning commas and putting commas after identifying clauses— would probably improve the results even more.

Finally, we contemplate building an ICALL (Intelligent Computer Assisted Language Learning) system to help learners to put commas correctly.

Acknowledgements

We would like to thank all the people who have collaborated in this research: Juan Garzia, Joxe Ramon Etxeberria, Igone Zabala, Juan Carlos Odriozola, Agurtzane Elorduy, Ainara Ondarra, Larraitz Uria and Elisabete Pociello.

This research is supported by the University of the Basque Country (9/UPV00141.226-14601/2002) and the Ministry of Industry of the Basque Government (XUXENG project, OD02UN52).

References

Aduriz I., Aranzabe M., Arriola J., Díaz de Ilarraza A., Gojenola K., Oronoz M., Uria L. 2004. *A Cascaded Syntactic Analyser for Basque Computational Linguistics and Intelligent Text Processing.* 2945 LNCS Series.pg. 124-135. Springer Verlag. Berlin (Germany).

Aldezabal I., Aranzabe M., Arrieta B., Maritxalar M., Oronoz M. 2003. *Toward a punctuation checker for Basque.* Atala Workshop on Punctuation. Paris (France).

Alegria I., Arregi O., Ezeiza N., Fernandez I., Urizar R. 2004. *Design and Development of a Named Entity Recognizer for an Agglutinative Language.* First International Joint Conference on NLP (IJCNLP-04). Workshop on Named Entity Recognition.

Ansa O., Arregi X., Arrieta B., Ezeiza N., Fernandez I., Garmendia A., Gojenola K., Laskurain B., Martínez E., Oronoz M., Otegi A., Sarasola K., Uria L. 2004. *Integrating NLP Tools for Basque in Text Editors.* Workshop on International Proofing Tools and Language Technologies. University of Patras (Greece).

Aranzabe M., Arriola J.M., Díaz de Ilarraza A. 2004. *Towards a Dependency Parser of Basque.* Proceedings of the Coling 2004 Workshop on Recent Advances in Dependency Grammar. Geneva (Switzerland).

Bayraktar M., Say B., Akman V. 1998. *An Analysis of English Punctuation: the special case of comma.* International Journal of Corpus Linguistics 3(1):pp. 33-57. John Benjamins Publishing Company. Amsterdam (The Netherlands).

Beeferman D., Berger A., Lafferty J. 1998. *Cyberpunc: a lightweight punctuation annotation system for speech.* Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pages 689-692, Seattle (WA).

Brill, E. 1994. *Some Advances in rule-based part of speech tagging.* In Proceedings of the Twelfth National Conference on Artificial Intelligence. Seattle (WA).

Brill, E. 1995. *Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging.* Computational Linguistics 21(4). MIT Press. Cambridge (MA).

Briscoe T., Carroll J. 1995. *Developing and evaluating a probabilistic lr parser of part-of-speech and punctuation labels.* ACL/SIGPARSE 4th international Workshop on Parsing Technologies, Prague / Karlovy Vary (Czech Republic).

Carreras X., Màrquez L. 2003. *Phrase Recognition by Filtering and Ranking with Perceptrons.* Proceedings of the 4th RANLP Conference. Borovets (Bulgaria).

Díaz de Ilarraza A., Gojenola K., Oronoz M. 2005. *Design and Development of a System for the Detection of Agreement Errors in Basque*. CICLing-2005, Sixth International Conference on Intelligent Text Processing and Computational Linguistics. Mexico City (Mexico).

Garzia J. 1997. *Joskera Lantegi.* Herri Arduralaritzaren Euskal Erakundea. Gasteiz, Basque Country (Spain).

Hardt D. 2001. *Comma checking in Danish.* Corpus linguistics. Lancaster (England).

Hill R.L., Murray W.S. 1998. *Commas and Spaces: the Point of Punctuation.* 11th Annual CUNY Conference on Human Sentence Processing. New Brunswick, New Jersey (USA).

Jones B. 1996. *Towards a Syntactic Account of Punctuation*. Proceedings of the 16th International Conference on Computational Linguistics. Copenhagen (Denmark).

Nunberg, G. 1990. *The linguistics of punctuation.* Center for the Study of Language and Information. Leland Stanford Junior University (USA).

Say B., Akman V. 1996. *Information-Based Aspects of Punctuation.* Proceedings ACL/SIGPARSE International Meeting on Punctuation in Computational Linguistics, pages pp. 49-56, Santa Cruz, California (USA).

Tjong Kim Sang E.F. and Buchholz S. 2000. *Introduction to the CoNLL-2000 shared task: chunking.* In proceedings of CoNLL-2000 and LLL-2000. Lisbon (Portugal).

Tjong Kim Sang E.F. and Déjean H. 2001. *Introduction to the CoNLL-2001 shared task: clause identification.* In proceedings of CoNLL-2001. Tolouse (France).

Van Delden S., Gomez F. 2002. *Combining Finite State Automata and a Greedy Learning Algorithm to Determine the Syntactic Roles of Commas.* 14th IEEE International Conference on Tools with Artificial Intelligence. Washington, D.C. (USA)

Zubimendi, J.R. 2004. *Ortotipografia. Estilo liburuaren lehen atala.* Eusko Jaurlaritzaren Argitalpen Zerbitzu Nagusia. Gasteiz, Basque Country (Spain).