A hybrid method for clause splitting in unrestricted English texts

Constantin Orăsan
School of Humanities, Languages and Social Sciences
University of Wolverhampton
Email: in6093@wlv.ac.uk

Abstract

It is important to know the structure of the sentence for many NLP tasks. In this paper we propose a hybrid method for clause splitting in unrestricted English texts which requires less human work than existing approaches. The results of a machine learning algorithm, trained on an annotated corpus, are processed by a shallow rule-based module in order to improve the accuracy of the method. The evaluation of the results showed that the machine learning algorithm is useful for identification of clause's boundaries and the rule-based module improves the results. Using some very simple rules we can report precision of around 88%.

1 Introduction

For many tasks within NLP it is important to know, at least at a shallow level, the structure of the sentence. Full syntactic analysis of a sentence is difficult to obtain and there are few programs capable of achieving good results for unrestricted texts. This paper attempts to present a method for splitting a complex sentence into clauses as a first step in obtaining the sentence structure and combines a machine learning technique with hand written rules.

The segmentation of complex sentences into clauses is important for various tasks such as machine translation, aligning parallel texts, text-to-speech systems and discourse processing, but has been surprisingly neglected by researchers. Early work in the field of clause splitting includes [4] where two experiments are described for finding the basic clauses in unrestricted English texts using in the first case a rule-based approach, and in the second a stochastic method. The system was designed to improve AT&T's text-to-speech system because, according to its author, the text sounds more natural if the final lengthening, boundary tones and pauses are inserted at clause boundaries. The results are surprisingly good for both experiments, but they are able only to identify a particular type of clause called a *basic clause* in the article, one of its shortcomings being that it fails to include embedded clauses. According to the paper, the results are between 95-99% depending upon the criteria of correctness used. The author points out that the results obtained with the stochastic method are marginally better, but the errors are due to both over-recognising and under-recognising clauses, while the finitary approach errors are systematically due to under-recognition.

A more recent system is the one presented in [8] where the system is used as a preprocessing step for bilingual alignment of parallel texts. The method adopted in this case is surface oriented and stepwise, and consists of preprocessing, tagging, clause marker tagging and partial parsing modules. A hand-crafted set of rules is used in order to find the clause boundaries. The system was designed for unrestricted English texts, but in the cited paper the evaluation is available only for ten regulations of the CELEX database, containing 562 clauses. The results stand at about 93%. In this case the rule-based approach is useful because the errors due to under-recognising clause boundaries, which are more likely to be made by this kind of method, are recovered in sequential modules for alignment.

Another method based on rules is presented in [5] as part of an English/Portuguese machine translation system. The main idea of this approach is that the clauses can ultimately be reduced to a noun, an adjective or an adverb, regardless of their length or the number of embedded clauses they may contain. The results are very good: 98% of the clauses are segmented correctly and 95% are correctly classified as nouns or adverbs. Being part of an automatic translation system this method is heavily reliant on a large number of resources such as dictionaries, part-of-speech tagger and lists of rules for solving ambiguities and marking the clause boundaries.

Although rule-based methods have proved to be successful in splitting clauses, they require a large set of hand-crafted rules, and, therefore, a great deal of human work. One way of avoiding this problem is to use learning algorithms to obtain a set of rules.

This paper discusses a hybrid method for clause splitting

which requires less human work than other approaches. The results of a machine learning algorithm, trained on an annotated corpus, are processed by a shallow rule-based module in order to improve the accuracy of the method. In our case, the machine learning algorithm is not used for finding rules, but for comparing an unseen case with all the instances in the training set and giving the most likely classification for it. A full description of the algorithm is presented in section 3.1. In this way, most of the clause boundaries are determined by the machine learning algorithm and hand-crafted rules are used only to further enhance the results.

Within this paper, discussion is structured as follows: In section 2 we define what constitutes for us a clause and the problems that can arise in identifying clauses in complex sentences. Section 3 will introduce some notions used within our method, the machine learning algorithm used and the training corpus and section 4 will present the method proposed. Section 5 contains an evaluation of the method and, finally, section 6 will present some conclusions and ideas for future research.

2 What is a clause?

It is difficult to define the notion of a clause predominantly because different grammar books suggest different interpretations. In order to define a clause [10] use its relation within the sentences; an independent clause constitutes a simple sentence, and more clauses related through subordination or coordination constitute a complex sentence, but the authors do not give a clear definition of a clause. In this grammar book the authors make a distinction between:

You can borrow my car if you need it. (1)

which is a complex sentence because the clause *if you need it* has an adverbial function and

You can borrow the car that belongs to my sister. (2)

which is a simple sentence because the clause *that belongs to my sister* is within the direct object. It is thus quite difficult to use such a definition.

[10] identifies three main structural types of clause:

- finite clause: a clause whose verb is finite
- non-finite clause: a clause whose verb is non-finite
- verbless clause: a clause that does not have a verb element, but is nevertheless capable of being analysed into clause elements

As we said a non-finite clause contains a non-finite verb. Usually, they do not require a subject, but in some cases it is possible to have one present as in the following example:

The best thing would be / to tell everybody (3)
The best thing would be / for you to tell everybody (4)

In (4) we prefer to analyse it as:

The best thing would be for you / to tell everybody (5)

where *for you* is the sister rather than daughter of the clause. This analysing scheme is used in the Susanne Corpus, the corpus we used for training.

A simpler definition for the clause is given in [1]: *a group of words containing a verb*. Using this definition we cannot identify the verbless clauses, a special kind of clause in which the verb was omitted in order to facilitate communication. The identification of verbless clauses is based mainly on the understanding of the text, which is far beyond our purposes.

In some cases syntactic ambiguities make it impossible for us to decide the structure of a sentence, such as in *I enjoy teaching* where *teaching* can be a noun phrase or a clause containing only a verb. The valence ambiguity of some verbs makes more than one structure acceptable as [5] shows for the verb *love*.

If you love money / problems show up (6)
If you love money problems / show up (7)
If you love / money problems show up (8)

Another problem appears when a participle occurs after the verb BE. In this case a choice must be made between on the one hand analysing the participle as the head of a subordinate clause, or treating the BE+participle sequence as a progressive or passive verb on the other. In ...*[they were [relieved] [to hear ...]]*, the word *relieved* can be replaced by *glad* or *happy*, so we consider it to form a clause, but depending on context the same sequence can be analysed in two different ways as shown in (9) and (10).

[The house was [sold]][when I enquired about it]] (9)
[The house was sold last week] (10)

The same problem appears with continuous tenses:

[This is [disturbing], for ...] (11)
[He is disturbing us] (12)

Given that in (9) – (12) semantic information is required to correctly split the sentence, the proposed method will not be able to tackle these cases correctly. Instead it will consider all the BE+participle and BE+ing form sequences as being a verb phrase.

In this section we showed that the task of clause splitting is not trivial, the main problems being caused by the impossibility of having a clear definition for the clause. In our approach we decided to use the definition proposed by [1] due to its simplicity, in this way being able to identify the clause boundaries using knowledge-poor methods.

3 Background

This section gives some background information regarding our method. In section 3.1 we introduce some notions about memory based learning and about TiMBL[3], a program which implements it. To train these algorithms it is necessary to have a training set and in our case we chose to extract this information from the Susanne corpus. Section 3.2 explains our reasons for choosing this corpus.

3.1 Memory based learning

Machine learning techniques have proved to be very successful for corpora processing. Reasoning is based on the similarity between new situations and ones present in the training corpus. In some cases, it is possible to obtain a list of rules abstracted from the training corpus; for decision tree learning the results can be if-then rules. In other cases, it is difficult to transform the results of learning into an intelligible format as in the case with neural networks and memory-based learning. A detailed explanation of all these methods can be found in [7].

Memory based learning has proved useful in many NLP tasks such as in letter-phoneme transliteration and part-of-speech tagging [2] and PP-attachment [14]. The success of the method in the case of NP-extracting [13] encouraged us to try the method for the task of clause splitting.

Another reason for choosing this machine learning algorithm is because it proved to be robust to noisy training data and is effective when a large set of training data is provided.

Memory-based learning simply stores the training examples; the generalisation beyond these examples is postponed until a new instance must be classified. This makes the algorithm fast in the phase of learning, but slow in the phase of classification, because the new example has to be compared with all the cases from the training set.

The most basic memory-based learning method is the k-NEAREST NEIGHBOUR algorithm. It considers that all the instances correspond to points in an n-dimensional space, where n is the number of attributes. To classify a new example means to assign the most frequent category within the nearest k most similar examples. Usually the distance used is the overlap metric:

$$\Delta(X,Y) = \sum_{i=1}^{N} \delta(x_i, y_i),$$
$$where\ \ \delta(a,b) = \begin{cases} 1, & a = b \\ 0, & otherwise \end{cases} \qquad (1)$$

The results obtained using this metric were poor, so in order to improve them we used a weighted overlap metric. In this case the relevance of each feature is computed and used for weighting their contribution to classifying an unseen example. Evaluating the results with different metrics, we found that the best results are obtained when we use gain ratio [9].

3.2 The Susanne corpus

The learning algorithm needs quite a large training set in order to obtain good results. In our case it was necessary to prepare a set which indicates the positions of clause boundaries in the sentences. This information can be obtained from corpora in which the structure of the sentence is marked as in the Penn Tree Bank and the Susanne Corpus.

Our choice was the Susanne Corpus [12], a freely available corpus developed at Oxford University. It contains a subset of Brown Corpus, more precisely 64 files with about 130000 words from 4 categories: A press reportage; G belles letters, biography, memories; J scientific and technical writing; N adventure and Western fiction.

Furthermore, the Susanne Corpus contains a very detailed grammatical annotation scheme. This annotation was done manually and we noticed a few errors in it, but due to the robustness of the learning algorithm they do not introduce too much noise into the training set.

Figure 1 displays a small example from the corpus.

As can be observed, the information is organised into columns. The first one represents a unique reference for each word indicating the file to which belongs, the sentence number and its position in the sentence. The second one is a status field encoding some information about the word. The tag set used by the Susanne Corpus is derived from Lancaster tag set. This information is stored in the third field of the line. The word and its lemma are in the fourth and fifth column of the line. The last field contains the information used by our method. It encodes the grammatical structure of the sentence using bracketing.

From this detailed annotation scheme we kept only the information about the clause. We decided to replace the part-of-speech tags associated with the words with tags generated by the QTAG [6]. The reason for this was because this level of encoding cannot be obtained using an automatic tool, so the clause splitter would not work on texts which are not from the Susanne Corpus.

The Susanne Corpus marks 17 types of clauses: main clause, subordinate clauses (adverbial, nominal, relative, comparative etc.), non-finite clauses (present participle clause, infinitival clause, for-to clause etc.) and verbless clauses (with clause, special as clause, reduced relative clause).

In order to process the corpus we converted it to an SGML format marking only the sentence and clause boundaries.

A01:0010b	-	AT	The	the	[O[S[Nns:s.
A01:0010c	-	NP1s	Fulton	Fulton	[Nns.
A01:0010d	-	NNL1cb	County	county	.Nns]
A01:0010e	-	JJ	Grand	grand	.
A01:0010f	-	NN1c	Jury	jury	.Nns:s]
A01:0010g	-	VVDv	said	say	[Vd.Vd]
A01:0010h	-	NPD1	Friday	Friday	[Nns:t.Nns:t]
A01:0010i	-	AT1	an	an	[Fn:o[Ns:s.
A01:0010j	-	NN1n	investigation	investigation	.
A01:0020a	-	IO	of	of	[Po.
A01:0020b	-	NP1t	Atlanta	Atlanta	[Ns[G[Nns.Nns]
A01:0020c	-	GG	+<apos>s	-	.G]
A01:0020d	-	JJ	recent	recent	.
A01:0020e	-	JJ	primary	primary	.
A01:0020f	-	NN1n	election	election	.Ns]Po]Ns:s]
A01:0020g	-	VVDv	produced	produce	[Vd.Vd]
A01:0020h	-	YIL	<ldquo>	-	.
A01:0020i	-	ATn	+no	no	[Ns:o.
A01:0020j	-	NN1u	evidence	evidence	.
A01:0020k	-	YIR	+<rdquo>	-	.
A01:0020m	-	CST	that	that	[Fn.
A01:0030a	-	DDy	any	any	[Np:s.
A01:0030b	-	NN2	irregularities	irregularity	.Np:s]
A01:0030c	-	VVDv	took	take	[Vd.Vd]
A01:0030d	-	NNL1c	place	place	[Ns:o.Ns:o]Fn]Ns:o]Fn:o]S]
A01:0030e	-	YF	+.	-	.O]

Figure 1. An example from the Susanne Corpus

4 The method

The method proposed in this section combines machine learning with rules for improving the results. The chosen architecture of the system is a pipeline (Figure 2), where the text is read from the standard input and processed by different modules, each adding more information to it.

The preprocessing module consists of a sentence splitter [11] and a part-of-speech tagger. We decided to use the sentence splitter because during the evaluation we noticed an increase of the system performance by about 2% if we know where the sentence boundaries are. For tagging we used QTAG [6], a probabilistic tagger. We decided to use this one because when there is an ambiguity in attaching a tag to a word it returns all the possible tags sorted according to their probability. In the case of ambiguities, we thought to use the first two tags returned by the tagger, but this doubled the time necessary for the machine learning algorithm to process the data, without obtaining noticeable improvement. If the input text is in SGML format, the preprocessing module saves the tags attached to words because the rest of the components cannot process SGML text. When the result is produced, the tags are added back to the texts.

As we said in section 2, every clause has to contain exactly one verb. The prepare module uses hand-written rules to identify the verb phrase of the sentences. Using [1] we designed rules for all important tenses of the English grammar. This information is used for computing the distance between verb phrases, which we found improves the results of the machine learning algorithm.

Using the results from the preprocessing module and of the rules for identification of verb phrases, the text is presented in columns suitable for the learning algorithm. After different tests the best results were produced using a three word window, one on the left side of the clause boundary and two on the right side and using the following attributes:

$$W_1 \quad T_1 \quad W_2 \quad T_2 \quad W_3 \quad T_3 \quad \text{BOS/—} \quad D_1 \quad D_2$$

where:

- W_1, W_2, W_3 are the words from the input text
- T_1, T_2, T_3 are the part-of-speech tags attached by QTAG to words
- BOS/— marks if a new sentence starts before the second word. During the tests we noticed better results if we have this information
- D_1, D_2 represents the distance from the previous verb phrase to W_2, and from W_2 to the next verb phrase

The final module, postprocessing, works at sentence level applying rules to improve the results from the classification algorithm. For this reason at this step the sentence boundaries are very important. During the evaluation of the learning algorithm's results we noticed a few errors which could be resolved using simple rules such as rules for identifying complex predicates, enumeration and the subject of the clause if it is a finite clause. All these rules use surface clues and rely on the results from the machine learning algorithm.

In a first step we apply two different types of rules. The first one removes the boundaries marked wrongly by the learning, referred in the text as the false positives. Sometimes this error appears inside of the verb phrase, because

Figure 2. The system's architecture

of noise in the training data. In this case the rules which identify the verb phrase make sure that there is no clause boundary inside of them.

The second type of rule is for correcting the false negatives, boundaries which were missed by the learning algorithm. One of the most frequent errors which appear in the results of the machine learning algorithm are when between two verb phrases there is no clause boundary. As we pointed out in section 2, every clause has exactly one verb. In the cases when between two verb phases there is no clause boundary we apply different heuristics for finding the correct place for it. We start from the second verb phrase and move the boundary to the left till the first appropriate place is found. For non-finite verb phrases, if the previous word is not a conjunction then the boundary is just before it. In the rest of the cases we mark the boundary before the conjunction. The cases where the second verb phrase is finite one, so it needs a subject, are more difficult. The subject is identified by searching immediately before the verb phrase for one of the following structures.

- the + [adj] +noun
- personal pronoun
- proper noun
- noun preceded by a determiner

If any of these structures is found then the subject is considered identified and the boundary is marked before the subject. In cases where the subject is preceded by a conjunction it is also included in the clause.

The classification algorithm indicates whether there is a clause boundary before a word in the input text, but it does not give any information about the type of boundary (if it marks the beginning or the end of the clause) and how to pair them in order to mark the limits of the clause. For this we use a rule-based approach and it constitutes the second step of the rule-based module. After the verb has been identified, rules search to the left and to the right of it until the clause boundaries are reached.

For finite clauses we have to make sure that they have a subject. In some cases, when there are embedded clauses, the left boundary is reached, before a word or group of words which can be the subject of the clause are found. In this case the search for the subject is continued to the left until a word or group of words which can be subject and do not belong to any clause are found. If the subject cannot be found in this way, the nearest word or group of words from the previous clause are selected as subject. The rules for deciding if a word or a group of words can be a subject are exactly the same as mention before. It should be emphasised that our purpose is not to find the subject, but to make sure that the clause has one.

5 Evaluation

We did not have another corpus to evaluate the proposed method, so we used the ten-fold cross-validation method. We randomly split the whole corpus in ten disjoint parts and we used nine for training and one for evaluation. We repeated the training-evaluation cycle ten times making sure that the whole corpus was used for training and evaluation. Note that for each iteration of the cross-validation, the learning process begins from scratch. The error rate is obtained by averaging the error rates from each of the 10 runs.

For measuring the performance of our system we used two measures: *precision* as being the ratio of the number of correct clause boundaries identified and the total number of clause boundaries within the text, and *recall* as being the ratio of number of the correct clause boundaries identified by the system and the total number of boundaries identified.

Firstly we evaluated the results of the machine learning algorithm. From a total of 28270 clause boundaries the algorithm correctly identified 23348 and produced 4922 under-recognition and 2202 over-recognition errors. This results in high recall, 91.38%, but poor precision 82.58%.

The next step was to evaluate the results after applying the rules. In this case the number of correctly identified clauses went up to 24986, but the number of over-recognition errors also rose to 3755, resulting 86.84% recall and 89.01% precision.

Our conclusion would be that a rule-based module helps to improve the results, but more complicated rules are necessary for better results. In terms of f-measure the overall improvement was 1.16 units, from 86.75 to 87.91.

The next experiment was to determine if the genre of the text influences the results. For this, we created 5 training files and 5 evaluation files with a comparable number of clause boundaries, four of them corresponding to the four domains in the Susanne Corpus and the fifth with texts from the whole corpus randomly selected. The results are presented in Table 1.

The results are lower than in the previous experiment because the training set was restricted only to the files from

Cat.	Total clause boundaries	Total sentences	False negative	False positive	Precision	Recall	F-measure
A	1481	381	176	268	88.12%	82.92%	85.46
G	1512	311	191	245	87.37%	84.36%	85.83
J	1476	297	209	197	85.84%	86.54%	86.19
N	1521	369	165	251	89.15%	84.38%	86.70
mix	1511	377	196	265	87.03%	83.23%	85.09

Table 1. The results for different categories

that domain and so the size of training set was smaller. The results obtained do not vary much across genres. Higher precision was obtained for A, press reportage, and N, adventure and western fiction, because the sentences are shorter and thus their structure is simpler. For the scientific texts we obtained the lowest precision and the highest recall. This suggests that their structure is more complicated, but more uniform.

Many of the errors are mainly because of preprocessing errors; in many cases the tagger, confuses NN and VB. An interesting task would be to evaluate the performances of the program without preprocessing errors. In order to do this we need a mapping scheme between the tag set used in the Susanne Corpus and the one used by Qtag. This is not a trivial task because in some cases a group of words is tagged with one tag in the corpus and more than one in Qtag.

6 Conclusion

In this paper we investigated a hybrid method for clause splitting. The results showed that a machine learning algorithm is very useful as the first step for this task and that the usage of very simple rules can improve the results slightly, but in order to obtain better results more complicated rules are necessary.

Future work includes rewriting the rule-based module for improving the results. In some tasks it is important to distinguish between different types of clauses. An interesting task would be to develop a classification algorithm for this.

7 Acknowledgments

I would like to thank Prof. Ruslan Mitkov, Richard Evans, Ramesh Krishnamurthy and Dr. Andrew Caink for their helpful comments and suggestions.

References

[1] Collins. *Collins Cobuild English Grammar*. Harper Collins, 1992.

[2] W. Daelemans, A. van den Bosh, and T. Weijters. Igtree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, (11):407–423, 1997.

[3] W. Daelemans, J. Zavarel, K. van der Sloot, and A. van den Bosch. Timbl: Tilburg memory based learner, version 2.0, reference guide, ilk technical report 99-01. ILK 99-01, Tilburg University, 1999.

[4] E. Ejerhed. Finding clauses in unrestricted text by finitary and stochastic methods. In *Proceedings of the 2nd conference on applied natural language processing*, pages 219 – 227, Austin, Texas, 1988.

[5] V. J. Leffa. Clause processing in complex sentences. In *Proceedings of the First International Conference on Language Resource&Evaluation*, volume 1, pages 937 – 943, May 1998.

[6] O. Mason. Qtag - a portable probabilistic tagger. 1997.

[7] T. M. Mitchell. *Machine learning*. McGraw-Hill, 1997.

[8] H. V. Papageorgiou. Clause recognition in the framework of alignment. In R. Mitkov and N. Nicolov, editors, *Recent Advances in Natural Language Processing*, pages 417 – 425. John Benjamins Publishing Company, Amsterdam/Philadelphia, 1997.

[9] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[10] R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A Comprehensive Grammar of the English Language*. Longman, 1985.

[11] J. J. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proc. of the Fifth Conference on Applied Natural Language Processing*, 1997.

[12] G. Sampson. *English for the computer: the SUSANNE corpus and analytic scheme*. Oxford University Press, 1995.

[13] J. Veenstra. Fast np chunking using memory-based learning techniques. In F. Verdenius and W. van den Broek, editors, *Proceedings of Benelearn*, pages 71–79, 1998.

[14] J. Zavarel, W. Daelemans, and J. Veenstra. Resolving pp attachement ambiguities with memory-based learning. In M. Ellison, editor, *Proc. of the Workshop on Computational Natural Language Learning (CoNLL'97)*. ACL, 1997.