Definition extraction using a sequential combination of baseline grammars and machine learning classifiers

Łukasz Degórski[1], **Michał Marcińczuk**[3], **Adam Przepiórkowski**[1,2]

[1]Institute of Computer Science, Polish Academy of Sciences, Warsaw
[2]Institute of Informatics, University of Warsaw
[3]Institute of Applied Informatics, Wrocław University of Technology

ldegorski@bach.ipipan.waw.pl, michal.marcinczuk@pwr.wroc.pl, adamp@ipipan.waw.pl

Abstract

The paper deals with the task of definition extraction from a small and noisy corpus of instructive texts. Three approaches are presented: Partial Parsing, Machine Learning and a sequential combination of both. We show that applying ML methods with the support of a trivial grammar gives results better than a relatively complicated partial grammar, and much better than pure ML approach.

1. Introduction

The aim of this paper is to contrast two approaches to the task of extracting definitions from relatively unstructured instructive texts (textbooks, learning materials in eLearning, etc.) in a morphologically rich, relatively free word order, determinerless language (Polish). The task is a part of a larger EU project, *Language Technology for eLearning* (LT4eL; `http://www.lt4el.eu/`), focusing on facilitating the construction and retrieval of learning objects (LOs) in eLearning with the help of language technology; the results of definition extraction are presented to the author or the maintainer of a LO as candidates for the glossary of this LO. Since it is easier to reject wrong definition candidates than to go back to the text and search for missed definitions manually, recall is more important than precision in the evaluation of the results.

Previous work (Przepiórkowski et al., 2007a; Przepiórkowski et al., 2007b) approached this task via the manual construction of partial (or shallow) grammars finding fragments of definition sentences. In this paper we attempt to quantify the extent to which the same task may be accomplished with the automatically trained machine learning classifiers, without the need to construct sophisticated manual grammars.

For the experiments described below, a corpus of instructive texts of over 300K tokens (with over 550 definitions) was automatically annotated morphosyntactically and then manually annotated for definitions. The corpus was split into two main parts: a training corpus (combined of what Przepiórkowski et al. (2007b) call a training corpus and a held-out corpus) and a testing corpus. The quantitative characteristics of these corpora is given in Table 1.

	training	testing	TOTAL
tokens	223 327	77 309	300 636
sentences	7 481	3 349	10 830
definitions	386	172	558
sentences with def.	364	182	546

Table 1: Corpora used in the experiments

2. Partial Parsing Experiments

Partial parsing experiments are most fully described in (Przepiórkowski et al., 2007b). Since the input texts were XML-encoded[1], the XML-aware efficient `lxtransduce` tool (Tobin, 2005) was used for the implementation of the grammar. The grammar, essentially a cascade of regular grammars, was developed within about 10 working days in over 100 iterations, where in each iteration the grammar was improved and the results were evaluated (on portions of the training corpus only) both quantitatively (automatically) and qualitatively (manually). The final grammar, called PG[2] (partial grammar), contains 13 top level rules (with 48 rules in total, in a 16K file, 12.5 lines for a rule, on the average).

For the evaluation, three baseline grammars were constructed: from the trivial B1 grammar, which marks all sentences as definition sentences, through B2, which marks as definitory all sentences containing a possible Polish copula (*jest*, *są*, *to*), the abbreviation *tj.* 'i.e.', or the word *czyli* 'that is', 'namely', to B3, a very permissive grammar marking as definitions all sentences containing any of the 27 very simple patterns (in most cases, single-token keywords) manually identified on the basis of manually annotated definitions (these patterns include all patterns in B2, as well as various definitor verbs, apparent acronym specifications, the equal sign '=', etc.).

For all grammars, a sentence was classified as a definition sentence if the grammar found a match in this sentence (not necessarily spanning the whole sentence).

All grammars were applied to the testing corpus, unseen during the development of the grammars; the results are given in Table 2. Apart from precision (P) and recall (R), also the usual F-measure is given (F_1), as well as F_2 used by Przepiórkowski et al. (2007a) and F_5 (apparently) used by Saggion (2004).[3] Note that for the task at hand, where

[1]More precisely, the input adheres to the XML Corpus Encoding Standard (Ide et al., 2000).

[2]This is the GR' grammar of Przepiórkowski et al. (2007b)

[3]It should, however, be noted that Saggion (2004) uses F_5 to evaluate definition answers to particular questions.

recall is more important than precision, the latter two measures seem appropriate, although whether recall is twice as important as precision (F_2) or five times as important (F_5) is ultimately an empirical issue that should be settled by user case evaluation experiments.

	P	R	F_1	F_2	F_5
B1	5.43	100.00	10.31	14.71	25.64
B2	9.69	61.54	16.74	22.11	32.53
B3	10.54	88.46	18.84	25.54	39.64
PG	18.69	59.34	28.42	34.39	43.55

Table 2: Partial grammar evaluation on the testing corpus

3. Machine Learning Experiments

Given the relatively small amount of data at our disposal (cf. Table 1 above) and the inherent difficulty of the task, we were skeptical about the applicability of machine learning approaches in this case, but, nevertheless, decided to perform some experiments, starting with some traditional well-known classifiers as implemented in the Weka toolset (Witten and Frank, 2005): Naïve Bayes, decision trees (ID3 and C4.5), the lazy classifier IB1 and AdaBoostM1 with Decision Stump, as well as the nu-SVC (EL-Manzalawy and Honavar, 2005) implementation of Support Vector Machines.

In case of the AdaBoost classifier, the number of iterations in the reported experiments was set to 1000. Other values (10, 100 and 10000) were also tested. Increasing the number of iterations led to the increase of the results, but also to the significant increase of the time of operation of the classifier. 10000 iterations took unacceptably long and the results were not much better than for 1000 iterations.

The nu-SVC classifier was used with radial basis kernel. Other kernels were also tested and proved worse. The nu parameter was set to 0.5 for 1:1 subsampling, 0.4 for 1:3, 0.2 for 1:5, 0.1 for 1:10 and 0.05 for no subsampling. In general, the higher the nu value, the better the results, but a value too high for a given subsampling ratio causes an error.

The attribute set was constructed as follows: for selected n-gram types (see Table 3[4]), we took the most frequent n-grams of every type from all the sentences in the corpus. The maximum number was arbitrarily chosen for each n-gram type; the numbers really used are smaller than this value when there are not enough possibilities (e.g., cases) in the corpus, and a little higher when there are a few n-grams with the same frequency exactly on the threshold.

For those experiments we used the whole corpus (training and testing, cf. Table 1 and Przepiórkowski et al. (2007b)), and applied the usual 10-fold cross-validation for the preliminary evaluation. For the sake of reproducibility, the corpus was split into folds once and this division was used in the following cross-validation experiments. The positive and negative examples were randomly assigned to one of the 10 subsets; the ratio of positive to negative examples in every subset was balanced.

n-gram type	max allowed	really used
base	100	100
base-base	100	100
base-base-base	100	115
ctag	100	100
ctag-ctag	100	100
ctag-ctag-ctag	100	100
case	100	8
case-case	100	59
case-case-case	100	100

Table 3: Feature set used in initial 10-fold cross-validation experiments

The corpus has (and every corpus of this type will inherently have) a prevalent number of negative instances. The ratio of non-definitions to definitions in our training texts is about 19. Thus, we decided to subsample the negative instances. For example, for the 1:5 subsampling ratio, all positive instances were used, and 5 times more negative instances were chosen randomly from the whole set of negative instances. A side effect of this approach is that the results of experiments with subsampling are still not 100% reproducible, owing to the randomisation factor. Some classifiers are more influenced by this factor, some are less, but the absolute differences of precision and recall between the results of two independent tests we have conducted very rarely exceed 0.5%, and tend to be balanced with regard to F-measures.

The experiments have shown that reducing the prevalence of negative examples noticeably increases recall — of course not without a loss of precision, but the change in terms of F_2 is always positive or negligible. For this reason, in further research we focused on configurations with the high subsampling ratio of negative instances. One positive side effect of subsampling was a substantial (up to 13 times in case of AdaBoost) decrease in execution time, as fewer examples have to be analysed.

The results of ten-fold cross-validation on the whole corpus (using the balanced random split), for different subsampling ratios, as well as results achieved on the same corpus by the grammars, are presented in Table 4. Even the best classifiers are significantly worse than the partial grammar PG. Note that some ML configurations achieve PG's precision, while other configurations — PG's recall, but never both at the same time.

4. Sequential combination of grammars with classifiers

In order to improve the precision, we applied the B3 grammar sequentially before the classifiers came into play. In this approach the classifiers filter the results of the grammar: all sentences rejected by B3 are unconditionally

[4]The number of ctags reported in the table is higher than the number of parts of speech in the IPI PAN Tagset used here, due to an error in corpus annotation (in two files, instead of part of speech only, full morphosyntactic information has been assigned as ctag). This additionally increased the noise in the data, so the results reported in this paper should be treated as a lower boundary on the actually attainable results.

Classifier	Ratio	P	R	F_1	F_2	F_5	Comments
NB	1:1	8.77	57.69	15.23	20.18	29.90	
	1:5	9.94	51.65	16.67	21.53	30.39	
	1:10	10.21	49.08	16.90	**21.62**	30.02	
	1:all	10.16	46.70	16.69	21.24	29.20	
C4.5	1:1	8.20	62.45	14.50	19.48	29.70	
	1:5	13.90	28.21	18.62	**21.00**	24.08	
	1:10	18.47	15.93	17.11	16.70	16.31	
	1:all	35.94	8.42	13.65	11.31	9.66	
ID3	1:1	8.26	64.29	14.65	19.72	30.18	
	1:5	13.11	36.63	19.31	**22.93**	28.20	
	1:10	15.52	26.56	19.59	21.47	23.74	
	1:all	17.57	19.05	18.28	18.53	18.78	
IB1	1:1	9.72	50.00	16.28	21.00	29.58	
	1:5	15.86	25.09	19.43	**21.01**	22.87	
	1:10	19.88	17.95	18.86	18.55	18.24	
	1:all	22.19	14.47	17.52	16.37	15.36	
nu-SVC	1:1	9.88	65.93	17.19	22.81	33.89	nu=0.5
	1:5	20.39	38.46	26.65	**29.69**	33.51	nu=0.2
	1:10	26.88	28.21	27.52	27.75	27.97	nu=0.1
	1:all	31.51	16.85	21.96	19.94	18.27	nu=0.05
AB+DS	1:1	10.59	54.95	17.75	22.92	32.35	10 iterations
	1:5	27.95	16.48	20.74	19.09	17.69	10 iterations
	1:1	11.89	66.48	20.17	**26.27**	37.66	100 iterations
	1:5	28.07	18.86	22.56	21.18	19.95	100 iterations
	1:1	11.67	68.32	19.94	26.10	37.77	1000 iterations
	1:5	27.49	20.70	23.62	22.55	21.59	1000 iterations
B3		9.12	89.56	16.55	22.73	36.26	
PG		18.08	67.77	28.54	**35.37**	46.48	

Table 4: Performance of the classifiers for different ratio of positive to negative examples evaluated using 10-fold cross-validation on the whole corpus with balanced random split, and evaluation of the grammars on the same (whole) corpus for comparison

marked as non-definitions, and only sentences accepted by B3 are passed to the ML stage, where their status is determined.

In these experiments the classifiers were trained on the training corpus and evaluated on the testing corpus. It effectively takes almost 10 times less time than cross-validation on the whole corpus, so an augmented set of features could be used. Apart from 1, 2 and 3-grams of single features, mixed combinations of attributes were added — see Table 5 for details.

4.1. Single classifiers

As the grammar in the preliminary stage takes some care of precision, classifier configurations with high recall turned out to be optimal, as they are complementary to the grammar. Thus, for all types of classifiers, the 1:1 subsampling ratio ensured the best results. The evaluation on the testing corpus for single classifiers with subsampling 1:1 is presented in Table 6.

Note that the best of these results, especially, B3 combined with the AdaBoost classifier, approach the results of the grammar PG, but still do not exceed them in terms of F_2.

n-gram type	max allowed	really taken
base	400	404
base-base	100	100
base-base-base	100	101
ctag	100	106
ctag-ctag	100	100
ctag-ctag-ctag	100	100
case	100	8
case-case	100	59
case-case-case	100	101
base-case	100	100
case-base	100	100
base-ctag	100	100
ctag-base-ctag	50	50
ctag-base-base-ctag	50	50
case-base-case	50	50

Table 5: Feature set used in filtering experiments

4.2. Ensembles of classifiers

In the next step we created homogeneous ensembles of classifiers. Every classifier in the ensemble was trained on all positive examples and a different subset of the nega-

Classifier	P	R	F_1	F_2	F_5
ID3	15.54	58.24	24.54	30.40	39.95
IB1	16.17	47.80	24.17	28.94	36.05
C4.5	15.97	56.59	24.91	30.62	39.74
NB	16.20	53.58	24.90	30.34	38.81
nu-SVC	17.44	62.09	27.23	33.50	43.52
AB+DS	18.27	60.44	28.06	34.16	43.65

Table 6: Filtering approach: results of single classifiers with subsampling 1:1

#	Classifier	P	R	F_1	F_2	F_5
7	**ID3**	19.94	69.23	30.96	37.95	49.03
3	**IB1**	16.98	45.05	24.66	29.04	35.32
7	**C4.5**	19.67	59.34	29.55	35.49	44.41
1	**NB**	16.20	53.58	24.90	30.34	38.81
3	**nu-SVC**	19.06	64.29	29.40	35.89	46.06
7	**AB+DS**	19.59	63.19	29.91	36.28	46.09
	B3	10.54	88.46	18.84	25.54	39.64
	PG	18.69	59.34	28.42	34.39	43.55

Table 7: Filtering approach: Best results of ensembles of classifiers with subsampling 1:1, and evaluation of the grammars on the same (testing) corpus for comparison

tive ones (size of the subset was determined by subsampling ratio). Then, majority voting was used to determine the decision of the whole ensemble for each sentence. In this way errors in classification made by one of the classifiers — especially those caused by an "unlucky" choice of negative examples — may be corrected by other classifiers in the ensemble.

The six classifiers were tested in ensembles of 3, 5, 7, 9 and 13, with variable subsampling ratios. The evaluation results for combinations of B3 with the best ensembles of classifiers with subsampling ratio 1:1 (as for single classifiers, this ratio always rendered the best results) are presented in Table 7, together with the analogous results for the pure grammars B3 and PG, repeated from Table 2. Note that four of these ensembles of classifiers, when combined with the baseline grammar B3, exceeded the results of the relatively sophisticated PG; only the lazy learner IB1 and the Naïve Bayes classifier resulted in F_2 significantly worse than that of PG.

The dependence of the results on the number of classifiers in each ensemble was not as straightforward as the monotonic dependence on subsampling ratio: some larger ensembles achieved worse results than smaller ensembles of the same type of classifiers.

It is however worth noting that, in case of some of the classifiers, the differences between results of bigger and smaller ensembles are negligible; taking into account the randomisation factor mentioned before, they could be treated as of equal quality. For instance, while AB+DS achieves the best results in an ensemble of 7, all other ensembles of this classifier are nearly as good. It is in a way similar to nu-SVC that scores best in an ensemble of 3 and almost as good in any larger configuration (but its results, when it is used on its own, are significantly worse). C4.5 achieves comparably good results in any ensemble of 7 or more. Other classifiers behave differently: ID3 is quite clearly the best in an ensemble of 7, IB1 performs equally well in an ensemble of any size (including 1), while combining a number of NB classifiers into an ensemble actually gives worse results than for a single NB classifier.

For practical applications the execution time may also be important. The approximate measurements show that, among the ensembles in Table 7, 7xID3, 7xC4.5, 1xNB and 3xnu-SVC are equally fast (1-2 minutes in our environment), 3xIB1 is slower (6 minutes[5]), and 7xAB+DS with 1000 iterations is the slowest (25 minutes[6]).

4.3. Role of B3 grammar

We conducted an additional experiment to verify the influence of the B3 grammar on the combination of classifiers. This was done by repeating some tests — single classifiers with the 1:1 subsampling ratio — without the initial filtering by the grammar. For details, see Table 8. This was in a way similar to the pure ML experiments described in the previous section — but this time the evaluation has been performed on the testing corpus and with the augmented set of features, so the results may be compared directly to those in Table 6.

Classifier	P	R	F_1	F_2	F_5
ID3	9.91	60.99	17.05	22.44	32.81
IB1	9.41	51.65	15.92	20.69	29.54
C4.5	10.90	60.44	18.47	24.03	34.39
NB	11.68	56.04	19.34	24.74	34.32
nu-SVC	10.76	64.29	18.44	24.19	35.15
AB+DS	13.47	63.19	22.20	28.33	39.12

Table 8: Pure ML approach: results of single classifiers with subsampling 1:1

The results — in terms of F_2 — are much better when B3 is applied before the classifiers. Table 9 visualises the relative differences between the results with and without B3, counted using the following formula:

$$\textit{value in Table 9} = \frac{\textit{value in Table 6}}{\textit{value in Table 8}} - 1$$

Note that a significant increase of precision is accompanied by only a small decrease of recall. Even though the B3 grammar rejects less than 12% of the sentences, it greatly improves the final result by apparently rejecting significant part of potential false positives.

5. Conclusion

The main result of this paper is that, for the task of definition extraction, a sequential combination of a very simple baseline partial grammar with machine learning algorithms gives results which are as good as — and sometimes

[5]As mentioned before, 1xIB1 is almost equally good, and two times faster.

[6]1xAB+DS with 1000 iterations, and even 1xAB+DS with 100 iterations, achieve F_2 exceeding 34%, the latter in 1.5 minutes.

Classifier	P	R	F_1	F_2	F_5
ID3	56,8%	-4,5%	43,9%	35,5%	21,8%
IB1	71,8%	-7,5%	51,8%	39,9%	22,0%
C4.5	46,5%	-6,4%	34,9%	27,4%	15,6%
NB	38,7%	-4,4%	28,7%	22,6%	13,1%
nu-SVC	62,1%	-3,4%	47,7%	38,5%	23,8%
AB+DS	35,6%	-4,4%	26,4%	20,6%	11,6%

Table 9: Relative gain of applying B3 before the classifiers (for single classifiers, 1:1 subsampling ratio)

significantly better than — the results of the application of manually constructed partial grammars, and much higher than the results of ML classifiers alone. Two corollaries of this result are: 1) even if only a small amount of noisy training data is available, the application of automatic machine learning methods may exceed pure grammar-based approaches, 2) but the clear improvement is observed only when such ML algorithms are supported by some — relatively trivial — *a priori* linguistic knowledge.

6. References

Yasser EL-Manzalawy and Vasant Honavar, 2005. *WLSVM: Integrating LibSVM into Weka Environment*. `http://www.cs.iastate.edu/~yasser/wlsvm`.

Nancy Ide, Patrice Bonhomme, and Laurent Romary. 2000. XCES: An XML-based standard for linguistic corpora. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2000*, pages 825–830, Athens. ELRA.

Adam Przepiórkowski, Łukasz Degórski, Miroslav Spousta, Kiril Simov, Petya Osenova, Lothar Lemnitzer, Vladislav Kuboň, and Beata Wójtowicz. 2007a. Towards the automatic extraction of definitions in Slavic. In Jakub Piskorski, Bruno Pouliquen, Ralf Steinberger, and Hristo Tanev, editors, *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing at ACL 2007*, pages 43–50, Prague.

Adam Przepiórkowski, Łukasz Degórski, and Beata Wójtowicz. 2007b. On the evaluation of Polish definition extraction grammars. In Zygmunt Vetulani, editor, *Proceedings of the 3rd Language & Technology Conference*, pages 473–477, Poznań, Poland.

Horacio Saggion. 2004. Identifying definitions in text collections for question answering. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004*, Lisbon. ELRA.

Richard Tobin, 2005. *Lxtransduce, a replacement for fsgmatch*. University of Edinburgh. `http://www.cogsci.ed.ac.uk/~richard/ltxml2/lxtransduce-manual.html`.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition. `http://www.cs.waikato.ac.nz/ml/weka/`.