Combining a rule-based approach and machine learning in a good-example extraction task for the purpose of lexicographic work on contemporary standard German

Lothar Lemnitzer[1], **Christian Pölitz**[2],
Jörg Didakowski[1], **Alexander Geyken**[1]

[1] Berlin-Brandenburgische Akademie der Wissenschaften, 10117 Berlin, Jägerstr. 22
[2] Technische Universität Dortmund, Fakultät für Informatik, Otto-Hahn-Str. 14, 44227 Dortmund
E-mail: {lemnitzer,didakowski,geyken}@bbaw.de, poelitz@uni-dortmund.de

Abstract

The work we will present in this paper is part of a dictionary project at the Berlin-Brandenburg Academy of Sciences and Humanities. For a large number of headwords, example sentences for their respective lexicographic descriptions have to be retrieved from a corpus of contemporary German. Lexicographers are typically faced with a huge number of corpus citations. Therefore, a tool that selects only good examples (those which are considered for inclusion into the dictionary) and dismisses the other ones would be time and effort effective. A rule-based good-example extractor proved to offer a good starting point, but the tool still delivers too many inacceptable citations. We have therefore tried to combine this tool with a machine learner that is trained on the decisions of an experienced lexicographer. The learner has been optimized to reject a large share of the example sentences. We present the machine learning results on a test data set with various combinations of linguistic features and quantify the gain in time and effort for the lexicographers. We also discuss the shortcomings of our approach and suggest some measures to counter them.

Keywords: example extraction; machine learning; corpus linguistics; German

1. Introduction and motivation

The work that will be reported in this paper originates from a large dictionary project at the Berlin-Brandenburg Academy of Sciences and Humanities (BBAW). The task is to update a legacy dictionary of contemporary German (Klein & Geyken, 2010). Approximately 45,000 lexical units that have become part of the German vocabulary during the last 40 years have to be registered and handled lexicographically (cf. Geyken & Lemnitzer, 2012). One of the principles of the work is to illustrate the lexicographical description, in particular concerning the meanings and usages of lexical items, with citations from a large German corpus.

The underlying corpus has been built and continually extended at the BBAW (cf. Geyken, 2007). A large share of it can be consulted and queried through a search engine on the website of the project (www.dwds.de). The corpus currently contains approximately 3 billion tokens. The sampling of new headwords from this corpus was mainly frequency based – most of the new headwords occur in these corpora with a frequency of >0.3 ppm (cf. Geyken & Lemnitzer, 2012); in absolute numbers: at least one thousand times. It is therefore impossible for a team of currently six lexicographers to read and check all these citations and to select the best three to five of them for inclusion in the dictionary article. Other straightforward alternatives such as sampling of k examples out of n, or just the first k examples, would not be satisfying either. Too many interesting contexts would escape the lexicographers' attention just because these citations occur further down the list. It has therefore been decided early in the project to work with a "good example extractor". The number of citations is parametrizable, i.e. the tool delivers for a headword those n citations that are ranked highest according to some qualitative criteria (see section 3 for further details). In the course of the lexicographical work – several hundred entries have currently been edited with the help of this tool – it was revealed that the selection of citations offered by this tool is still far from optimal. In particular, the number of "false positives", i.e. citations which are ranked high but are rejected by the lexicographers, is still far too high. As little of the lexicographers' work as possible should be wasted by checking bad corpus citations. To achieve this goal, it has been decided to post-process the output of the good-example extractor by a machine learning approach. The applied method should ideally learn lexicographical quality criteria and thus reduce the number of examples to those which are most likely to be considered by them for inclusion in the dictionary article.

In this paper we will report first results of this approach, i.e. of combining a rule-based good-example extractor with a machine learning component into a processing pipeline. In section 2, we will give an overview of related wok. In section 3 we will briefly outline the operation mode of the rule-based extractor. In section 4 we will characterize the data we use for our machine learning experiments. Section 5 will be devoted to a description of our machine leaning approach. The results of the experiments will be presented in section 6. We will end with a conclusion and an outline of our further work.

2. Related Work

Activities in the field of good-example extraction are comparatively recent. Of course discussion among lexicographers regarding what counts as good examples and for which purposes have been taking place for a long time. See, for example, Harras (1989) who mentions a list of linguistic criteria that a good lexicographic example should meet. Many of the introductions into (practical) lexicography, e.g. Svensén (2004: 281ff.), Atkins & Rundell (2008: 452ff.) and Engelberg & Lemnitzer (2009: 235ff.) devote at least a section to the function and quality of citations and other examples. However, only the advent of very large corpora that provide large numbers of citations made a (semi-)automatic pre-selection of material necessary. The seminal work in this field is that of Adam Kilgarriff and colleagues (Kilgarriff et al., 2008). They present a rule-based approach to extracting good examples on the basis of some operationalisable quality criteria. The good example extractor implemented at the BBAW largely follows the approach presented in their paper (see section 3 and Didakowski et al., 2012). However, bringing ML methods into the field of automatic Gdex has recently become more impactful (cf. Rundell, 2014). In February 2015, a workshop of the "European Network of e-Lexicography was devoted exclusively to this topic (http://www.elexicography.eu/working-groups/working-group-3/wg3-workshops/automatic-extraction-of-good-dictionary-examples). On this occasion researchers from several European dictionary projects presented their work on that topic. To the best of our knowledge none of the work presented there has been published so far (but cf. Kosem et al., 2011 and Volodina et al., 2012). However, from the slides that are available on the website it can be deduced that some of the projects involve machine learning methods and tools in order to improve the precision of the extraction task.

3. Combining machine-learning with a rule-based approach

In Didakowski et al. (2012) we presented a good-example extractor that serves the lexicographers at the DWDS project by reducing the number of citations to be inspected. The extractor provides only those citations for a headword which are classified as most suitable with regards to a set of predefined rules. The extractor implements hard and soft rules which work on sentence level and global rules which work on a set of citations. The violation of a hard rule leads to immediate rejection of a citation. An example of such a rule is that a citation must be within a predefined range for sentence length. On the other hand, soft rules are used to rank the remaining citations by score. If a citation does not meet a soft rule it receives a lower score than a citation which does. A typical soft rule is that a citation should contain as few free pronouns as possible (for further details, cf. Didakowski et al., 2012). Additionally, the set of citations which is presented to the lexicographers should be well distributed among several text types (newspapers, novels, scientific prose, etc.) as well as over time – the dictionary should cover the period between 1900 and the present. For this purpose, global rules are applied to the ranked citation set making use of bibliographic metadata. In this connection the extractor is parametrizable – the users can decide how many citations are presented to them. The motivation behind using such a tool is not only to save time and effort for the lexicographers – who have more important things to do than reading hundreds of nearly identical and mostly uninteresting citations – but also to provide them with a "starter set" of typical usage types from which they should be able to construct the various senses of the headword. Furthermore, for the dictionary user the examples should be comprehensible without further context.

In the course of the work with that tool it became evident that 15 to 20 examples serve as a good material basis for the lexicographers to obtain an overview of the various uses of most of the lexical items. It also arose that the ratio of good to bad examples was less than optimal. Lexicographers are still confronted with too many examples which they dismiss for various reasons. For example, many of the dismissed examples a) are structurally too complex to be exposed to the dictionary user; b) contain still too many pronouns and are therefore hard to comprehend without further context; c) are structurally incomplete even if the parser provides a “complete” analysis (list items are typical examples of such incomplete structures) or d) contain spelling or slight grammatical errors. It could thus offer a considerable saving of time (and money) if the lexicographers are provided with a smaller and better sample of citations. Such a task, however, is beyond the capabilities of a rule-based extractor that has to balance internal features, such as linguistic information, and external features, such as the temporal and topical distribution of the citations. For such reasons the idea arose to apply a machine learner to the output of the rule-based example extractor. The learner should be trained on the examples which have been already classified as either appropriate or not appropriate for inclusion into a dictionary article. In the future, the machine learning component should ideally reduce the inappropriate examples and keep the appropriate ones. In the following section we describe the data used for the training and testing of the learner.

4. The data

From the list of headwords that are to be included in the updated dictionary, we selected approximately 1,050 headwords. For each of these headwords, the good example extractor provided 18 examples at most – for some of the headwords only a smaller number of good examples were available. This totaled approximately 13,200 examples. All examples that had passed the rule-based good-example extractor were classified by one of the authors, a trained and experienced lexicographer, into one of two classes: (1) appropriate for inclusion, and (2) not appropriate for inclusion. These classified examples are used as training and test data for the machine learning task. The numbers in the data set are as follows: 5,984 have been labeled as appropriate (= class 1, “good”); 7,328 examples as not appropriate (= class 2, “bad”).

For the machine learning experiment, the set of classified examples was split into two, half for training and half for testing. Assignment to one of the two groups was done randomly. The distribution of good and bad examples over the two sets is shown in Table 1.

Quality Dataset	Good (= class 1)	Bad (= class 2)
Training set	3,607	3,011
Test set	2,377	4,317
Sum total	5,984	7,328

Table 1: Distribution of examples between the training and test sets.

Due to the random sampling, the distribution of class 1 and class 2 examples varies in the training and test sets. However, this difference in distribution does not affect the performance of the machine leaning component.

5. The machine-learning approach

Our goal is to further refine the output of the good-example extractor from Didakowski et al. (2012) by combining it with a machine learning approach. We use Support Vector Machines (SVM, cf. Joachims, 1998) to “learn” which classifiers should be able to separate the good examples from the bad ones. The SVM learns a non-linear decision function that maps a set of features extracted from the example citations to a binary variable. We use several distinct representations of the texts in order to extract these features. In particular, we use a bag-of-words representation that encodes frequencies of words in the texts, parts-of-speech representations that assign word classes to the text tokens, and parse trees that encode syntactic structures. The text of each example is transformed into a sequence of these elements according to the different representations: for bag-of-words, the text is represented as sequence of words, for part-of-speech, the text is represented as sequence of the morpho-syntactic classes of the words; for parse trees, we represent the texts as sequences of trees in bracket notation.

For example, the text “I went to Lancaster” is represented as follows. For bag-of-words representation we receive “I went to Lancaster”; for part-of-speech we get “PP VVD TO NP”; and for the parse tree we are given “(S(NP-SBJ(PRD),VP(VVD,PP(TO,NP(NNP)))))”.

Sub-string kernels as proposed by Vishwanathan & Smola (2004) are used to calculate the similarity between examples based on common subsequences in the corresponding representations. All subsequences are used as features, i.e. all of the resulting substrings, sub-trees of the parse trees and sequences of part-of-speech tags. For instance, one feature of the above text “I went to Lancaster” in its part-of-speech representation is how many times the two labels “PR” and “VP” co-occur. Similarities between texts are encoded in a so-called kernel matrix that is used for the SVM. The entries in this matrix can be considered as indicators of the similarity of two texts based on the number of shared features, hence common sub-strings, common sub-graphs in the parse trees or common subsequences of parts-of-speech. Using the kernel matrix, we are able to train the SVM even on large feature sets, since we need only to calculate common subsequences instead of enumerating all possible subsequences of our texts in the corresponding representations. Further details on kernel methods can be found in Hoffmann et al. (2007).

We implemented our method in Java as Plugin in RapidMiner (Mierswa, 2009), a state of the art Data Mining tool. The bag-of-words representation was built by transforming the tokens of the example texts into normalized words (‘lemmas’). The parts-of-speech and the parse trees were assigned to the texts by the Stanford Parser with a grammar for German (cf. Rafferty et al., 2008). The SVM was learned using the LibSVM library (cf. Chang et al., 2011) which is available in the RapidMiner software. The calculation of the kernel matrix was also implemented in Java as Plugin for RapidMiner. The individual kernel entries were calculated following Vishwanathan & Smola (2004). The implementation uses efficient data structures and hashing mechanisms that facilitate and therefore speed up the calculations. Thus we are able to calculate the kernel matrix for large data sets of many long text examples.

6. Results

The machine learner would be perfect if it would sort out (and remove) all examples from the test set which have been hand-labeled as not appropriate by the human annotator and, on the other hand, accept all examples which have been labeled as appropriate. We know that this is impossible. First, the decisions of the human annotator are arbitrary to some degree and cannot be predicted by even the best machine learner. Second, the training and test set may differ in many regards. Therefore, we can imagine the optimal result as either of the two following strategies: a) the learner tries to keep as many good (= class 1) examples as possible, at the price of also keeping (too) many of the bad (= class 2) examples. In other words, the learner will be optimized for a lower precision and a higher recall. That would be a conservative approach (i.e. one that conserves many examples for further inspection by the lexicographers); b) the learner tries to remove as many bad examples as possible, at the price of removing (too) many good examples along the way. In other words, the learner will be optimized for a higher precision and a lower recall. That would be the more radical approach.

Since our goal is to is to reduce the lexicographers' time spent reading and considering a surplus of bad examples, and in light of the fact that most headwords are represented by many examples in the corpus, we chose the second, radical, approach for the training strategy of the machine learner.

Subsequently, we will report on the performance of the learner on the test data set with three different sets of features: bag-of-words (or, more correctly, bag-of-lemmas), sequences of parts-of-speech and sub-trees of parse trees as well as combinations thereof. For each of these features we use the sub-sequence kernel described earlier to train a support vector machine such as machine learning. Since the decision is a binary one, i.e. assigning an example to one of two classes, and the performance of the learner is compared to human judgement, the data can be ordered and presented in a four-cell (2x2) contingency table. The four cells contain the number of examples that are a) assigned to class 1 by the human annotator ('ha') and by the machine learner ('ml'), b) assigned to class 2 by ha and class 1 by ml; c) assigned to class 1 by ha and class 2 by ml and d) assigned to class 2 by both ha and ml.

ha ml	class 1 (good)	class 2 (bad)	
class 1	603 (a)	487 (b)	1,090 (e)
class 2	1,774 (c)	3,830 (d)	5,604 (f)
	2,377 (g)	4,317 (h)	6,694 (i)

Table 2: A 2x2 contingency table for an example data set.

We can compute the marginal sums for each of the rows and columns (cells e–h) and the sum total (cell i). In Table 2 we present the full contingency table for one of our experiments. We can derive the following measures from this table:

- recall for class 1 examples = 603 / 2,377 = 25.3% (i.e. approx. one fourth of the class 1 examples according to ha are labeled as such by ml)
- recall for class 2 examples = 3,830 / 4,317 = 88.7%
- precision for class 1 examples: 603 / 1,090 = 55.3% (i.e. slightly more than half of the class 1 examples according to ha are accepted by ml, the rest are dismissed)
- precision for class 2 examples: 3,830 / 5,604 = 68.3%.

From these figures we further derive the F-score, i.e. the weighted mean of recall and precision as well as the accuracy. Accuracy is defined as the number of correctly-classified examples divided by the sum total of examples (i.e (cell a + cell d) / cell i). For our example:

- the F-score for class 1 examples is 0.34
- the F-score for class 2 examples is 0.76
- the accuracy is 0.66

In Table 3, we list the recall and precision for both class 1 and class 2 examples, which are listed for several feature settings.

Feature representation	Recall class 1	Precision class 1	Recall class 2	Precision class 2
Bag-of-lemmas	0.23	0.55	0.89	0.68
Part-of-speeches	0.30	0.57	0.87	0.69
Parse trees	0.32	0.60	0.88	0.70

Table 3: Recall and precision for both classes and different sets of features

From these values, the F-score for class 1 and class 2 examples, as well as the accuracy value, can be derived, see Table 4.

Feature representation	F-score class 1	F-score class 2	Accuracy
Bag-of-lemmas	0.32	0.76	0.66
Part-of-speeches	0.39	0.78	0.67
Parse trees	**0.42**	**0.78**	**0.68**

Table 4: F-score and accuracy for different sets of features.
The best values achieved are highlighted.

The data in Tables 3 and 4 show that all feature settings work reasonably well, i.e. we achieve a significant reduction of class 2 examples while still preserving a sufficient number of class 1 examples. The differences between the feature configurations are minimal, with the parse tree feature generating the best result.

From the point of view of the lexicographer, two questions are important beyond the measurable performance of the learner: i) how many (good, bad) examples do I get rid of? and ii) do I have to face, at the end of the selection process, a significant share of headwords with no example left at all? Let us look into both questions on the basis of our test data set and the example given in Table 1.

i) From the 6,694 examples that have been selected by the rule-based example extractor, only 1,090, i.e. 16.3%, have been accepted by the learner and therefore are available for the lexicographers' inspection. Of course, the loss of good examples is also considerable. In the example setting 1,774 class 1 citations would be lost, which leads us to the second question.

ii) The test data consist of examples for 438 headwords. For 415 of these, there is at least one example which has been classified into class 1 by the learner. Unfortunately, for only 342 of the headwords there is at least one example that has also been assigned to class 1 by the human annotator. The loss of (really) good examples is therefore considerable and should be remedied somehow.

As we have shown above, the implementation of a machine learning component as a filter is also a matter of choosing a good measure of permeability of such a filter. However, there is no invariant optimal setting for this measure. The optimal setting depends upon the task and the context. In our context, there were a sufficiently large number of citations to draw from, a limited amount of time for the lexicographers to inspect these examples and a rather small number of citations which were eventually selected for inclusion in the dictionary. The optimal setting in such a context equates to a reduction of as many bad examples as possible, at the price of also removing many good examples. Nevertheless, it is not acceptable that for a larger amount of headwords no example is accepted at all. In the next section we will therefore present some suggestions of how to cope with this 'collateral damage'.

7. Conclusions and further work

We have learned from our experiment that the machine learner, using the radical approach to removing example sentences from the initial set, also removes a considerable number of examples the lexicographers might want to see and potentially consider for inclusion in their articles. We, therefore, suggest the following strategies to remedy this 'collateral damage'.

1. The simplest strategy would be to increase the initial data set, i.e. to instruct the rule-based good-example extractor to provide a larger number of example sentences. As a consequence, the number of examples that are accepted by the machine learner is larger but still of a higher quality than the set of examples that is initially delivered by the good-example extractor.

2. A more ambitious approach would be to use more information in order to balance the number of false negatives (= rejected examples we would like to see) against the number of false positives (= accepted examples which we would not like to see). One of the interesting characteristics of the machine learning approach that we have been using is that it does not only deliver a decision but also a confidence level for the decision. The confidence values for all possible decisions add up to 1; therefore, they can be interpreted as the probability that the decision is correct. Currently, the value is set to class 2 if the confidence towards this class is >0.5. One could try to set a higher confidence level for the rejection of an example sentence. We have not yet looked into this, but an experiment with different thresholds might improve the results.

Another issue which affects all forms of example selection is the polysemy of many headwords. Typically, a polysemous word is very often used in one major sense, and less often or infrequently in its other(s) senses. This kind of distribution of usage examples over sentences makes each kind of sampling prone to the error of missing all examples for the infrequent sense(s). The burden to detect such gaps is again with the lexicographer. Ideally, the example sentences for a headword are initially grouped into clusters that, with more or less precision, represent different senses of the headword and outliers that cannot be easily assigned to any sense. Such an approach to combining good example extraction with word sense induction has been suggested by Rundell et al. (2014). We will in our future research follow the ideas expressed in this paper and apply them to our (German) data.

8. Acknowledgements

This research has been carried out in the context of the BMBF-funded project KobRA (*Korpus-basierte Recherche und Analyse mit Hilfe von Data-Mining*, grant ID 01UG1245) and the "Digitales Wörterbuch der Deutschen Sprache" (DWDS) at the Berlin-Brandenburg Academy of Sciences. We also want to express our gratitude to the developers of the "Rapid Miner" data mining tool.

9. References

Atkins, S. & Rundell, M. (2008). *The Oxford Guide to Practical Lexicography.* Oxford: Oxford University Press.

Chih-Chung C. & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2, 3, Article 27 (May 2011), 27 pages. DOI=10.1145/1961189.1961199 http://doi.acm.org/10.1145/1961189. 1961199

Didakowski, J. et al. (2012). Automatic example sentence extraction for a contemporary German dictionary. In: Proceedings EURALEX 2012, Oslo, pp. 343-349.

Engelberg, S.n & Lemnitzer, L. (2009). *Lexikographie und Wörterbuchbenutzung. 4. Auflage.* Tübingen: Stauffenburg.

Geyken, A. (2007). The DWDS corpus: A reference corpus for the German language of the 20th century. In: Fellbaum, C. (ed.): *Collocations and Idioms: Linguistic, lexicographic, and computational aspects.* London: Continuum, pp. 23-41.

Geyken, A. & Lemnitzer, L. (2012). Using Google Books Unigrams to Improve the Update of Large Monolingual Reference Dictionaries. In *Proceedings of EURALEX 2012, Oslo*, pp. 362-366.

Harras, G. (1989). Theorie des lexikographischen Beispiels. In F.J. Hausmann, O. Reichmann, H.E. Wiegand & L. Zgusta (eds.) *Wörterbücher Dictionaries Dictionnaires: Ein internationales Handbuch zur Lexikographie*, Berlin/New York: de Gruyter, pp. 1003-1114.

Hofmann, T., Scholkopf, B. & Smola, A. J. (2007). 'Kernel Methods in Machine Learning', Published online at arXiv.org \url{http://arxiv.org/abs/math/0701907v2}.

Joachims, T. (1998). Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. In C. Nedellec & C. Rouveirol (eds.) *Proceedings of the 10th European Conference on Machine Learning (ECML '98),* London: Springer-Verlag,, pp. 137-142.

Kilgarriff, A. et al. (2008). GDEX: Automatically Finding Good Dictionary Examples in a Corpus, In E. Bernal & J. DeCesaris (eds.) *Proceedings of the Thirteenth EURALEX International Congress*, Barcelona, Spain, pp. 425-432.

Klein, W. & Geyken, A. (2010). Das Digitale Wörterbuch der Deutschen Sprache (DWDS). In U. Heid et al. (eds.). *Lexikographica.* Berlin/New York, pp. 79-93.

Kosem, I., Husak, M. & McCarthy, D. (2011). GDEX for Slovene. Ljubljana. Trojina, Institute for Applied Slovene Studies. In I. Kosem & K. Kosem (eds.) *Electronic lexicography in the 21st Century: New Applications for New Users. Proceedings of eLex2011, Bled, Slovenia, 10 – 12 November 2011*, Ljubljana; Trojina, Institute for Applied Slovene Studies, pp. 151-159.

Lodhi, H. et al. (2002). Text classification using string kernels. *J. Mach. Learn.* Res. 2 (March 2002), 419-444. DOI=10.1162/153244302760200687 http://dx.doi.org/10.1162/153244302760200687

Mierswa, I. (2009), 'Non-Convex and Multi-Objective Optimization in Data Mining - Non-Convex and Multi-Objective Optimization for Statistical Learning and Numerical Feature Engineering.', PhD thesis, Universität Dortmund.

Rafferty, A.N. & Manning, C.D. (2008). Parsing three German treebanks: lexicalized and unlexicalized baselines. In *Proceedins of the Workshop on Parsing German (PaGe '08).* Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 40-46.

Rundell, M. et al. (2014): Applying a Word-sense Induction System to the Automatic Extraction of Diverse Dictionary Examples. In: *Proc. 16th EURALEX International congress*, *Bolzano, July 2014*, Bolzano: EURAC, pp. 319-329.

Svensén, B. (2004). *A Handbook of Lexicography.* Cambridge: Cambridge University Press.

Vishwanathan, S. V. N. & Smola, A. J. (2004). Fast Kernels for String and Tree Matching. In K. Tsuda, B. Schölkopf & J. Vert (eds.) 'Kernels and Bioinformatics' , MIT Press, Cambridge, MA, USA.

Volodina, E. et al. (2012): Semi-automatic selection of best corpus examples for Swedish: initial algorithm evaluation. Workshop on NLP in Computer-Assisted Language Learning. Proceedings of the SLTC 2012 workshop on NLP for CALL. Linköping Electronic Conference Proceedings 80: pp. 59–70. Accesed online: http://spraakbanken.gu.se/sites/spraakbanken.gu.se/files/SLTC2012_hitex_reviewed.pdf