Feature Stacking for Sentence Classification in Evidence-Based Medicine

Marco Lui
NICTA VRL
Department of Computing and Information Systems
University of Melbourne
mhlui@unimelb.edu.au

Abstract

We describe the feature sets and methodology that produced the winning entry to the ALTA 2012 Shared Task (sentence classification in evidence-based medicine). Our approach is based on a variety of feature sets, drawn from lexical and structural information at the sentence level, as well as sequential information at the abstract level. We introduce feature stacking, a metalearner to combine multiple feature sets, based on an approach similar to the well-known stacking metalearner. Our system attains a ROC area-under-curve of 0.972 and 0.963 on two subsets of test data.

1 Introduction

The ALTA Shared Task 2012[1] was a sentence-level classification problem in the domain of biomedical abstracts. Given a collection of abstracts pre-segmented into discrete sentences, the task is to label each sentence according to one of 6 pre-defined classes. The dataset used was introduced by Kim et al. (2011), which also give a description of the classes and an analysis of their distribution. In this work, we will describe the winning entry, focusing on the feature sets and machine learning techniques used.

The main contributions of this work are: (1) additional features to describe sentences for automatic classification of sentences to support evidence based medicine beyond those of Kim et al. (2011), (2) a method for performing the task that does not use a sequential learning algorithm, and (3) a method to combine multiple feature sets that outperforms a standard concatenation approach.

[1]http://www.alta.asn.au/events/sharedtask2012

2 Task Description

The dataset of Kim et al. (2011) (hereafter referred to as NICTA-PIBOSO) consists of 11616 sentences (10379 after headings are removed), manually annotated over the 6 PIBOSO classes (Kim et al., 2011). For the shared task, NICTA-PIBOSO was divided by the competition organizers into train and test partitions. Participants were given labels for the training sentences, and asked to produce an automatic system to predict the labels of the test instances. We do not give further details of the task as it will be covered in much greater depth by the shared task organizers in a paper that will appear alongside this paper.

The shared task was hosted on Kaggle,[2] and as part of Kaggle's standard competition structure, the test dataset was further subdivided into "public" and "private" subsets. Participants did not know which test sentence belonged to which subset. Each submission by a participant consisted of predictions over the entire test set, and Kaggle then automatically computed the competition metric broken down over the public and private subsets. Participants were allowed to submit up to 2 entries per day, and upon submission were immediately given a score on the public subset. The score on the private subset was withheld until after the conclusion of the submission period. Final ranking of competitors is based on the private subset of the test data; the breakdown between public and private serves to penalize entries that overfit the test data in the public subset. The method we describe in this work was the top-scoring system on both the public and private subsets.

[2]http://www.kaggle.com

3 Software Used

All experimentation and analysis was implemented using `hydrat`[3], a declarative framework for text categorization developed by the author. Word tokenization was carried out using NLTK (Bird et al., 2009). The learning algorithm used was logistic regression, as implemented in `liblinear` (Fan et al., 2008). For part-of-speech tagging, we used `TreeTagger` (Schmid, 1994).

4 Features

NICTA-PIBOSO contains two different types of abstracts, *structured* and *unstructured*. Unstructured abstracts are free text, as is the common format in NLP literature. Structured abstracts are divided into sections by headings, such as "Background" or "Outcome", and are becoming increasingly common in biomedical literature. For the shared task participants were not given an explicit indication of which abstracts were structured, or which "sentences" were actually headings. In this work, we applied a simple heuristic: any sentence which contained only uppercase letters was considered a heading, and any abstract containing a heading was considered structured. This definition is slightly more simplistic than that used by Kim et al. (2011), but in practice the difference is minimal.

4.1 Lexical Features

Lexical features are features drawn from the text of a sentence. The lexical feature sets we use are: (1) BoW, a standard bag-of-words. We retained the 15,000 most frequent words, and did not apply stopping or stemming. (2) LEMMAPOS, bigrams of part-of-speech tagged lemmas. (3) POS, bigrams and trigrams of part-of-speech tags, without the underlying lemma. Whereas BoW and LEMMAPOS are fairly standard lexical features, POS is relatively novel. We included POS based on the work of Wong and Dras (2009), which used POS n-grams to capture unlexicalized aspects of grammar in order to profile a document's author by their native language. The intuition behind the use of POS for our task is that sentences from different PIBOSO categories may have systematic differences in their grammatical structure.

Each of BoW, LEMMAPOS and POS are extracted for each sentence. We then use these features to define lexico-sequential features, which are simply the summation of the feature vectors of specific sentences in the same abstract as the target sentence. We refer to these other sentences as the *context*. The contexts that we use are: (1) all prior sentences in the same abstract, (2) all subsequent sentences, (3) n-prior sentences ($1 \leq n \leq 6$), (4) n-subsequent sentences ($1 \leq n \leq 6$), (5) n-window (i.e. n-prior and n-subsequent, $1 \leq n \leq 3$). These lexico-sequential features are intended to capture the information that would be utilized by a sequential learner.

4.2 Structural Features

Structural features model characteristics of a sentence not directly tied to the specific lexicalization.[4] In this work, our structural features are: (1) SENTLEN, the length of the sentence, in both absolute and relative terms, (2) HEADING, the heading associated with each sentence, (3) ABSTLEN, the length of the containing abstract, and (4) ISSTRUCT, a Boolean feature indicating if the abstract is structured.

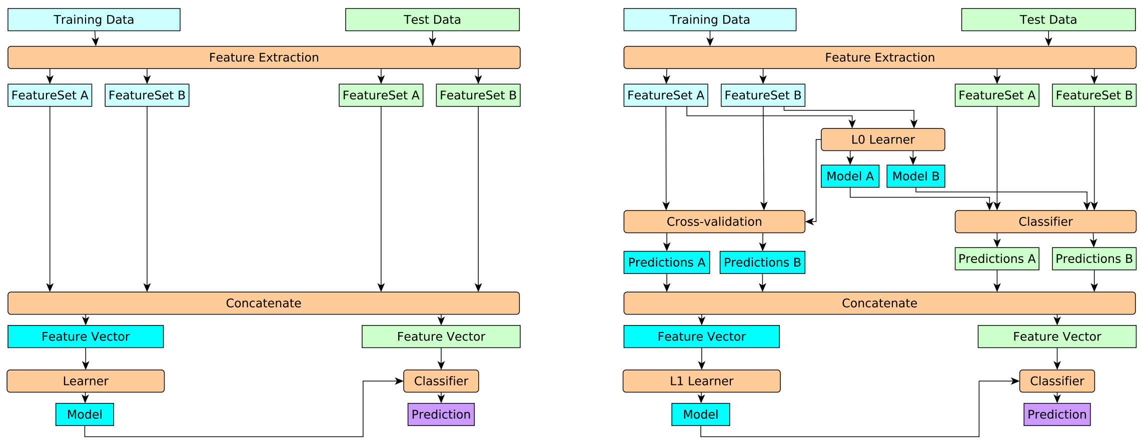
We treat HEADING similarly to BoW, LEMMAPOS and POS, and extract the same 5 types of sequential (indirect dependency) features. We also extract POSITION, a set of sequential features based on the position of the sentence in the abstract, in both absolute and relative terms.

4.3 Differences with Kim et al. (2011)

To summarize, the differences between our sentence features and those of Kim et al. (2011) are: (1) we use POS n-grams in addition to POS-tagged lemmas, (2) we used sentence length as a feature, (3) we expanded indirect dependencies to include sentences both before as well as after the target sentence, and (4) we increased the scope of indirect dependencies to include BoW, POS as well as section heading information. Differently to Kim et al. (2011), we did not use (1) MetaMap (or any thesaurus), (2) rhetorical roles to group headings, and (3) direct dependency features.

[3]`http://hydrat.googlecode.com`

[4]The distinction between lexical and structural is somewhat arbitrary, as for example the length of sentences is obviously dependent on the length of the words contained, but we maintain this distinction for consistency with Kim et al. (2011).



(a) concatenation (b) feature stacking

Figure 1: Comparison of (left) a standard concatenation-based approach to combining feature sets with (right) feature stacking, a metalearning approach.

5 Classifiers

Our main challenge in building a classifier was the need to integrate the large variety of features we extracted. The feature sets are very heterogeneous; some are large and sparse (e.g. BoW), whereas others are small and dense (e.g. structural features). Relative weighting between feature sets is difficult, and simply concatenating the feature vectors often led to situations where adding more features reduced the overall accuracy of the system. Rather than attempt to tune feature weights in an ad-hoc fashion, we opted for a metalearning approach. The intuition behind this is that in principle, the output of "weak" learners can be combined to produce a "strong(-er)" learner (Schapire, 1990).

The metalearner we implemented is closely related to stacking (Wolpert, 1992). We call our approach *feature stacking* in order to highlight the difference, the main difference being that in conventional stacking, a number of different learning algorithms (the *L0* learners) are used on the same training data, and their respective predictions are combined using another learner (the *L1* learner). In our approach, we do not use different algorithms as L0 learners; we always use logistic regression, but instead of training each L0 learner on all the available features, we train a learner on each feature set (e.g. BoW, LemmaPOS, etc). Hence, we are learning a "weak" learner for each feature set, which are then composed into the final "strong" learner. This approach has two main advantages over simple concatenation of features: (1) it learns the relative importance of each feature set, and (2), it allows learning of non-linear relationships between features.

Figure 1 shows a side-by-side comparison of the two approaches to feature combination. The key difference is that the stacking approach introduces an additional inner (L0) layer, where each instance is projected into the stacked feature space. Given that we have n feature sets and k possible classes, each sentence (training and test) is passed to the L1 learner as a $n \times k$ feature vector. The process for converting L0 features into L1 features is different for the training and the test data, because we only have labels for the training data. For the training data, we use a cross-validation to generate a vector over the k classes for each sentence. We repeat this once for each of the n feature sets, thus yielding the $n \times k$ feature L1 representation. For the test data, we do not have labels and thus for each of the n feature sets we train a classifier over all of the training sentences. We use each of these n classifiers to generate a k-feature vector for each test sentence, which we then concatenate into the final $n \times k$ feature L1 representation.

We chose logistic regression as the learner after initial results indicated it outperformed naive Bayes and SVM in feature stacking on this task. Logistic regression is theoretically well-suited to feature stacking, as stacked logistic regression corresponds to an artificial neural network (Dreiseitl and Ohno-Machado, 2002).

Combination	Output	Public	Private
Concatenation	Boolean	0.885	0.883
Stacking	Boolean	0.893	0.875
Stacking	Probability	0.972	0.963

Table 1: ROC area-under-curve for Public and Private test sets, using (1) feature stacking or concatenation for feature combination, and (2) Boolean or probabilistic output.

6 Results

In this work, we report results that were made available on the Kaggle leaderboard. These results are not directly comparable to previous work (Kim et al., 2011; Verbeke et al., 2012), because the evaluation metric used is fundamentally different. Previously, the task was evaluated using metrics based on precision, recall and F1-score, which is standard for classification tasks. However, in the shared task the metric used was the Receiver Operating Characteristic area under curve (ROC AUC). This metric is common in information retrieval, and it takes into consideration not just a single classification for each sentence, but rather the relative ordering between classes, in order to evaluate a system's ability to trade off precision for recall. This is an easier problem than classification, because classification is all-or-nothing; an instance label is either correct or wrong. Ranking-based metrics such as ROC AUC soften this, by penalizing ranking the correct class second much less than ranking it sixth.

Despite this ranking based metric, there was some initial confusion amongst competitors as to whether classification predictions (i.e. a Boolean value for each possible class) or ranking predictions (i.e. a probability value for each class, which is used to rank the classes) should be submitted. This was clarified by the organizers, and led to all participants seeing substantial increases in score. This difference can be seen in Table 1, where our system improved from 0.893 to 0.972 on the public leaderboard. For Boolean output, we assigned only the most probable label to each sentence, whereas for probabilistic output, we provided the computed probability of each label. Our Boolean output essentially ignored the small proportion of multi-label sentences, treating all sentences as mono-label. This likely accounts for some of the increase in score, though we expect that a good proportion is also due to instances where the correct class was ranked second.

In Table 1, our performance on the public leaderboard suggested that the stacking-based approach to feature combination improved over the concatenation approach (also using logistic regression). On this basis, we focused all further development on using the stacking-based approach. However, the private leaderboard results (which were only released at the conclusion of the competition) tell a different story; here the stacking result is lower than the concatenation result on Boolean output. Unfortunately, we did not submit a run using probabilistic output from concatenation, so we do not have this data point for comparison. Based on just these results, we cannot draw any conclusions on whether the stacking approach outperforms concatenation. We are currently carrying out further evaluation, based on the full dataset (including the goldstandard labels of the test data), which was only made available to participants after the conclusion of the shared task. This evaluation will be based on micro-averaged F1-score, in order to enable direct comparison to the results of Kim et al. (2011) and Verbeke et al. (2012). Our early analysis is highly encouraging, indicating that feature stacking clearly outperforms concatenation, and that our method outperforms the published state-of-the art on this task (Verbeke et al., 2012), with particular improvement on unstructured abstracts. We are currently investigating if this is attributable to our extended features or to feature stacking. We expect to make our full analysis available at a later date.

7 Conclusion

In this work, we presented the features and methodology that were used to produce the winning entry to the ALTA2012 Shared Task. We provided an overview of the feature sets, and a detailed description of feature stacking, a metalearning approach to combining feature sets to produce a high-accuracy classifier for the task. In future work, we will provide a more detailed analysis of the impact of the metalearning approach, as well as the relative impact of the different feature sets, using micro-averaged F1-score as the metric for direct comparability to previous work. We will also compare the use of sequential features with stacked logistic regression to a sequential learning algorithm.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Sebastopol, USA.

Stephan Dreiseitl and Lucila Ohno-Machado. 2002. Logistic regression and artificial neural network classification models: a methodology review. *Brain and Cognition*, 35:352–359.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Su Nam Kim, David Martinez, Lawrence Cavedon, and Lars Yencken. 2011. Automatic classification of sentences to support evidence based medicine. *BMC Bioinformatics*, 12:1–10.

Robert E. Schapire. 1990. The Strength of Weak Learnability. *Machine Learning*, 5:197–227.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the Conference on New Methods in Natural Language Processing*, Manchester, 1994.

Mathias Verbeke, Vincent Van Asch, Roser Morante, Paolo Frasconi, Walter Daelemans, and Luc De Raedt. 2012. A statistical relational learning approach to identifying evidence based medicine categories. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 579–589, Jeju Island, Korea, July. Association for Computational Linguistics.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

Sze-Meng Jojo Wong and Mark Dras. 2009. Contrastive analysis and native language identification. In *Proceedings of the Australasian Language Technology Workshop 2009 (ALTW 2009)*, pages 53–61, Sydney, Australia, December.