KNOW At The Social Book Search Lab 2016 Mining Track

Hermann Ziak and Andi Rexha and Roman Kern

Know-Center GmbH
Inffeldgasse 13
8010 Graz, Austria
hziak, arexha, rkern@know-center.at

Abstract. This paper describes our system for the mining task of the Social Book Search Lab in 2016. The track consisted of two task, the classification of book request postings and the task of linking book identifiers with references mentioned within the text. For the classification task we used text mining features like n-grams and vocabulary size, but also included advanced features like average spelling errors found within the text. Here two datasets were provided by the organizers for this task which were evaluated separately. The second task, the linking of book titles to a work identifier, was addressed by an approach based on lookup tables. For the dataset of the first task our approach was ranked third, following two baseline approaches of the organizers with an accuracy of 91 percent. For the second dataset we achieved second place with an accuracy of 82 percent. Our approach secured the first place with an F-score of 33.50 for the second task.

Keywords: Text Mining, Classification,

1 Introduction

The Social Book Search Lab on the CLEF 2016 conference consisted of three tracks: suggestion, mining and interactive track. Within this work we describe our approach on the mining track. The tracks cover challenges that relate to the field of Just in Time Information Retrieval [6] which is also closely related to the field of recommender systems. In particular this includes challenges like automated query formulation, document ranking, and relevant context identification. The mining track is most relevant for the last of these challenges. The task itself was organised via two tasks. Within the classification task a dataset consisting of postings from LibraryThing[1] and Reddit[2] were given. Here the task was to identify the postings that contained requests to book recommendations. The LibraryThing postings were therefore labelled to be either a request or a

[1] `www.librarything.com`
[2] `www.reddit.com`

normal thread posting. The Reddit threads were selected from two Subreddits: “suggestmeabook” and “books”.

The second task was linking in the reverse direction, thus linking books with postings. Here the goal was to identify a reference to a book within a thread. The threads were again taken from LibraryThing containing the about 200 initial postings with about five to fifty replies each. The task was not about highlighting the exact title and location within the text but stating the according work ID. For the classification task we applied traditional text mining feature engineering methods, like stemming and according feature extraction. We submitted different runs, which represent different classification algorithms. The first three runs were conducted using well known machine learning algorithms. The results submitted as fourth run was based on the idea of a Vote/Veto ensemble classifier [4]. For the linking task we followed an approach based on a lookup table. Here we made use of the provided Amazon and LibraryThing book dataset [1]. We managed to be ranked on the third and second place on the LibraryThing and Reddit dataset for the classification task and to placed on the first place for the linking task. This is particularly encouraging as we did not conduct extensive optimisation upon the basic algorithms.

2 Approach

The base of the two tracks, "Mining" and "Suggestion" tracks, are the provided book data collections from Amazon and LibraryThing, with about 2.7 million books and according meta-data. We decided to transform the given structured data into a data structure, which should be quicker to access, thus making use of an indexed format. Consequently the dataset was parsed and indexed with Apache Solr[3], which is based on the Apache Lucene[4] search-engine library.

The "Mining Track" of the SBS challenge consisted of two task: The "Classification Task" with the goal of classifying forum entries as book recommendation requests and the "Linking Task" where the task was to identify books within the text and report the according LiberyThing internal book ID. Within both approaches we used our Solr search index containing the Amazon book dataset.

2.1 Classification Task

For the classification task two datasets were provided by the organizers. The Reddit training set containing about 250 threads from the "suggestmeabook" and threads from the "books" Subreddit. The "suggestmeabook" threads were the positive examples and the "books" threads were considered to be the negative examples. A similar but smaller testing set was provided as well were the category field were masked.

The second, more comprehensive dataset was extracted from LibararyThing itself. Here 2,000 labelled threads were provided for training and another 2,000

[3] `http://lucene.apache.org/solr/`
[4] `https://lucene.apache.org/`

threads for testing. About 10 percent of the training threads were labelled as positive examples. Initially we started by parsing both datasets and uniting the given data within one data structure. We shuffled the entries within this data structure and split it two separate sets: the first part were used for training and the second part for validation, whereas the validation part was only a small fraction of the whole set.

The only preprocessing step, which we applied on the dataset, was stop word removal. To train the classifiers we extracted several types of features. The first types features are found in many text mining and natural language processing system, like n-grams. Although it is common to use TF/IDF based weighting scheme, for reasons of simplicity we decided to just use the sheer frequency of features within the text. Based on these basic features, we introduced a number of other features.

The first of the custom features are the number of terms within the text. Next we extracted the tags and browse nodes from the Amazon Dataset. The found tags and browse nodes within the user's text were higher than the basic features. Finally we extracted a feature based on the count of average spelling errors within the posting. We decided to introduce this feature based on our assumption that user asking for book recommendations might be more literate than the average user and therefore might as well make fewer spelling errors. This feature has the additional benefit that postings not containing decent text at all would be penalized further. Some of the classification algorithms we initially intended to use could not cope with missing features within the dataset. Therefore all missing features had to be added to the each single feature vector with zero weight.

For the very first test run we used only a single, dedicated feature: The quantity of question marks with in the entry. To our surprise with this simple feature the Naive Bayes approach already reached an accuracy of over 80 percent. Since we considered this to be an error at our end, we investigated this issue more closely. The final conclusion was that the imbalance of positive and negative examples has led to this result. Therefore we further separated the validation data into the positive and negative examples to get a more detailed information about the performance of the approach and features. We also created a more balanced training set by keeping all positives examples but using only a fraction of the negative examples for some of the classification algorithms.

With our feature extraction pipeline and the individually balanced training sets we could finally train the three chosen classification algorithms: A Random Forest classifier [2], a Naive Bayes classifier [3] and finally a Decision Tree [5]. The parameters like maximum depth of the Random Forest classifier or amount of negative postings within the training set of the Naive Bayes approach where chosen by manually optimizing on the accuracy on our custom validation dataset. For example, we obtained the best results for the Random Forest classifier by sticking with the default of 10 as a target tree depth. Additionally we also worked an approach that was based on the idea of a Vote/Veto ensemble classifier, where we implemented a dedicated voting schema. Only if the majority of the

algorithms decided that the posting contained a book recommendation request the posting was labelled as such.

2.2 Linking Task

As basis for the linking task a dataset extracted from the LibraryThing website was provided by the lab organizers. This dataset contained of about 500 threads from users discussing about books while often mentioning book titles. Furthermore those threads included the replies to the initial posting and also contained potential candidates.

Our initial approach to tackle this task was to implement a lookup table. To generate our initial lookup table we extracted all titles and selected parts of the metadata (e.g. authors, creator, International Standard Book Number (ISBN)) from the Amazon dataset. To reduce the size and clean the data we conducted a number of preprocessing steps. We removed English stop words, removed or replaced special characters, removed additional information about the book provided within the title (e.g. binding information) and stemmed the title terms. The same preprocessing steps were applied upon the text of the posting entries.

Finally we implemented a lookup algorithm to match the potential candidates ISBN to the LibraryThing work IDs which had to be reported. Basically all the preprocessed book titles from the Amazon dataset were used for a simple string matching algorithm on each sentence in the posting.

The biggest issue with this kind of approach is the high amount of false positives, i.e. matches, which do not refer to any books. Most of in the following described approaches we tried were not included in the final results. Nevertheless we briefly describe our strategies how to resolve this problem. To reduce the amount of false candidates one strategy is to introduce a weight to all candidates and then remove all those, which falling below a certain threshold. As potential factors for such weighting scheme we considered the occurrence of the author's name within the same sentence as the corresponding book title. Often this cooccurrences of the author's name within the same sentences are either stated directly ahead of the book title (e.g. Stephen King's The Dark Tower) or directly following the title (e.g. The Dark Tower by Stephen King).

Furthermore we experimented with a supervised approach, to train a classifier to distinguish between sentences containing books and those, which do not mention books. The basic idea was to lower the weight for the book candidate if the book titles were found in sentences potentially not containing a book. This was made possible as parts of the dataset consisted of texts were the titles of the book were annotated. We extracted each of this sentences and applied the same feature extraction pipeline than in the classification task. Although both of the approaches may appear valid, we decided against using them, because of these reasons: First of all the classifier did not work on a satisfying accuracy level, with only about 60 to 65 percent on average. Secondly, even though the co-occurrence of an author's name within the text might validate the book title candidate, it might not necessarily mean that the other candidates are less likely

correct. And finally it is hard to estimate the amount of actual titles within the text, i.e. it is hard to find an appropriate threshold for the weights. Finally, to reduce the false positives at least to a certain degree we decided to just remove book titles from the dataset that consisted only of one non stop word term.

3 Results

In this section we describe the results of our system.

3.1 Classification Task

In Table 1 we present the results of our approach on the classification task with the validation dataset created out of the original training dataset. The figures represent the accuracy of each approach. Here the Random Forest approach and the Naive Bayes classifier performed on the same level. The Vote/Veto ensemble classifier inspired algorithm achieved slightly lower results, about one percent, whereas the Decision Tree achieved the lowest results with about eight percent lower than the top algorithms.

Table 1. Results of the first run with only a small validation dataset created out of the training data. The results represent the accuracy on the combined datasets of "LibraryThing" and "Reddit".

	Naive Bayes	**Decision Tree**	**Random Forest**	**Vote/Veto**
Accuracy	84.10	78.12	84.09	83.21

Table 2 shows the result of the official test run where the two datasets are evaluated separately. Here on the LibraryThing dataset the Naive Bayes classifier has the best accuracy, ranking on place three, followed by the Vote/Veto classifier ranking on place six.

Table 2. Official results on the testing data. The accuracy on the "Reddit" and "LibraryThing" data are reported separately.

	Naive Bayes	**Decision Tree**	**Random Forest**	**Vote-Veto**
LibraryThing	91.59	83.38	74.82	90.63
Reddit	82.02	76.40	74.16	76.40

Table 3 shows parts of the official results stated on the SBS Lab website [5]. It contains a comparison of our top performing approach, based on Naive Bayes, versus the top performing baseline approach based on a Linear Support Vector Classifier. The third results originate from the baseline based on Naive Bayes.

[5] `http://social-book-search.humanities.uva.nl/#/mining16`

Table 3. Official results on the baseline versus our approach. The baseline provided used 4-grams as features classified by Linear Support Vector classifier and a Naive Bayes classifier.

	Naive Bayes KNOW	Naive Bayes Baseline 4-gram	Linear SVC Baseline 4-gram
LibraryThing	91.59	87.59	94.17
Reddit	82.02	76.40	78.65

3.2 Linking Task

Table 4 presents the figures of the linking task. Here our system performed with an accuracy and recall of 41.14 and a precision of 28.26 resulting in the F-score of 33.50 and was ranked on the first place.

Table 4. Official results on the testing data for the linking task.

	Accuracy	Recall	Precision	F-score
Linking Task	41.14	41.14	28.26	33.50

4 Discussion

Given that the Naive Bayes approach is of low complexity compared to the best performing system, the baseline with an Linear Support Vector Classifier, it appears that our selected features worked well. This is especially apparent, when comparing our Naive Bayes approach with the provided baseline, see Table 3. Within the official run both the Decision Tree and the Random Forest approach fared behind the others. Interestingly within our preliminary tests upon our own validation set, the Random Forest based approach achieved nearly the best results. This could be based on the fact that we did not apply any further optimization, like pruning on the tree based algorithms.

Given the simplicity of our approach for the linking task it seemed to work, especially well in regards to the recall. As expected the precision is low in comparison. The datasets and results indicate that users tend to be quite accurate when it comes to stating book titles within written text. A bigger issue, than to identify the titles itself, seems to be the identification of false positives within the candidate list. Many book titles have the tendency to be short or use phrases that occur often within natural language.

5 Conclusion and Future Work

Given we trained only one set of classifiers for both datasets it seems that our approach generalizes well. For future work we want to investigate the performance of our selected feature set by applying different classification algorithms.

We expect the linking task to allow the most room for further improvement. In particular, we plan to rise the precision of the approach. Investing in a novel approach to detect sentences containing books, might be associated with the biggest gain.

Acknowledgments

The presented work was developed within the EEXCESS project funded by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement number 600601. The Know-Center is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

1. Beckers, T., Fuhr, N., Pharo, N., Nordlie, R., Fachry, K.N.: Overview and results of the inex 2009 interactive track. In: Research and Advanced Technology for Digital Libraries, pp. 409–412. Springer (2010)
2. Breiman, L.: Random forests. Machine learning 45(1), 5–32 (2001)
3. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. pp. 338–345. Morgan Kaufmann Publishers Inc. (1995)
4. Kern, R., Seifert, C., Zechner, M., Granitzer, M.: Vote/veto meta-classifier for authorship identification. In: CLEF 2011: Proceedings of the 2011 Conference on Multilingual and Multimodal Information Access Evaluation (Lab and Workshop Notebook Papers), Amsterdam, The Netherlands (2011)
5. Quinlan, J.R.: C4. 5: programs for machine learning. Elsevier (2014)
6. Rhodes, B.J.: Just-in-time information retrieval. Ph.D. thesis, Massachusetts Institute of Technology (2000)