Keyword Extraction Using Artificial Neural Networks and a TextRank Variant

The University of Georgia

Justin Payan with Advisor Dr. Bill Hollingsworth

INSTITUTE FOR ARTIFICIAL INTELLIGENCE
Franklin College

INTRODUCTION

Keyword extraction is the process of selecting representative phrases from a document. It is very useful for information retrieval, document clustering, summarization, and event detection. Keywords can be selected by hand, but automated extraction is much more efficient.

In this work, we extract basic linguistic and statistical features for presentation to a supervised artificial neural network classifier. The classifier is then ensembled with a TextRank model, based on Google's PageRank and developed by Mihalcea and Tarau [3]. The resulting model can distinguish key phrases from a list of candidate phrases.

CORPUS

Our corpus consists of abstracts from the journals *Computers and Control* and *Information Technology*. Each abstract is annotated with its title and a list of human-selected key phrases. We removed any key phrases that do not appear in the abstract title or body.

The corpus was randomly split into 1500 training documents and 500 testing documents. The corpus is available at [2]. The 1,500 training documents yield 83,230 candidate phrases with their extracted features.

The best approach in Hulth's original paper attained a precision of 25.2% and a recall of 51.7%, for an F-score of 33.9%. The best recent score on this task was in [1], which combines two well-known systems with conditional random fields to achieve an overall F-score of about 51%.

ALGORITHMS

Each candidate phrase has a series of features computed for it: relative within document frequency, estimated relative overall corpus frequency, relative placement of the first occurrence, phrase length, its Word2Vec vector space embedding, and the Word2Vec vector space embeddings of the first two nouns/adjectives in the title of the abstract. These features are fed as inputs into the artificial neural network model, which we train using R's caret package. The model is trained using 10 repetitions of 10-fold cross-validation, and we up-sample the data to counteract the fact that there are many more negative instances of key phrases than positive.

The TextRank [3] portion of the algorithm works as follows on each document individually: First we construct a graph with nodes consisting of the noun and adjective unigrams (single words). We draw edges between words that have a normalized NPMI score of at least 0.4 (NPMI can be between 0 for not correlated, to 1 for completely correlated). We then iteratively compute the nodes' scores until convergence, computing each node's score as

$$S(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

In the above equation, d = 0.85. To transform the unigram scores into phrasal scores, we take the product of the scores of the phrase's unigrams.

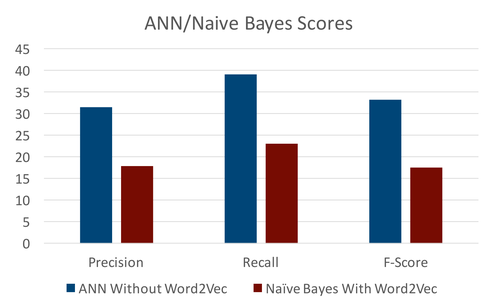
The two models output a list of selected key phrases, and they are ensembled via a weighted majority voting scheme.

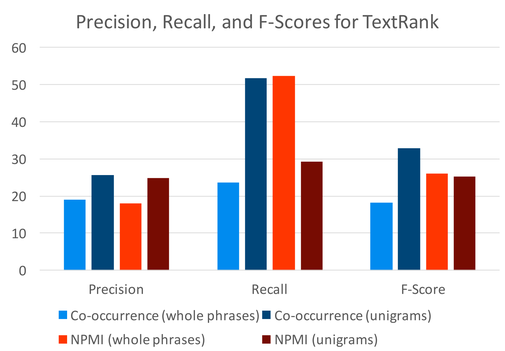
PRELIMINARY RESULTS

To evaluate the success of the model, we employ the statistics of precision, recall, and F-score. Precision measures the percentage of key phrases suggested by the model that are actually present in the human-supplied gold standard, while recall measures how much of the human-supplied key phrases the model was able to successfully suggest. F-score is a combination of the two statistics, computed as the harmonic mean of precision and recall:

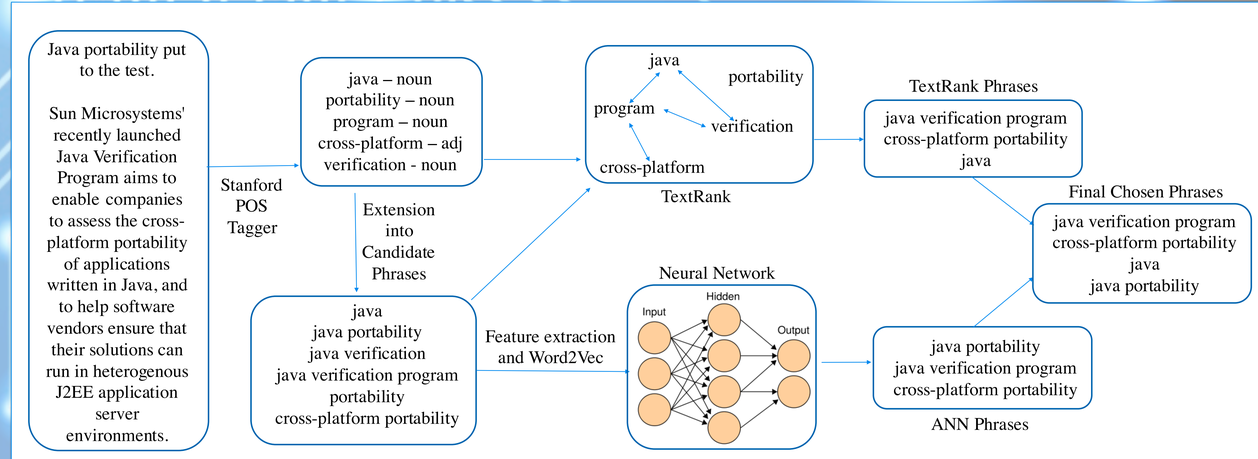
$$\frac{2 * precision * recall}{precision + recall}$$

Due to computational constraints, we compare an artificial neural network without the Word2Vec embeddings to a Naïve Bayes classification machine learning model that does use Word2Vec embeddings. In the future, we will find tools that will enable us to actually use Word2Vec embeddings with a neural network. We also compare four variants of the TextRank algorithm on the three measures.





EXAMPLE WORKFLOW



DISCUSSION

Our best models have F-scores in the low to mid-30s, which is acceptable given the literature.

It does not seem as though the Word2Vec embeddings provide an improvement in performance. However, it is quite likely that the lack of improvement is due to the fact that the Naïve Bayes model was unable to capture the latent information in the vector embeddings. We must develop a more sophisticated classification model to take proper advantage of the vector embeddings.

The NPMI TextRank variant performs almost as well as the traditional co-occurrence based TextRank. The lower performance was most likely due to the fact that a very small minority of word pairs actually had NPMI scores associated to them. To get such scores, we need a larger corpus. We have a corpus consisting of comments and posts from 57 computer science-related Stack Exchange forums. The raw text amounts to 6.22 GB of information, so we were unable to use it in this study, but will find a way to do so in the future.

FUTURE WORK

The next step for the neural network is to use deep learning on a multi-core GPU computer. It is likely that the semantic information created by the Word2Vec embeddings cannot be adequately exploited by the simple neural network architecture present in R's nnet library. The combination of deep learning and high-performance computing will allow us to take full advantage of the vector-space embeddings.

It would be a good idea to improve the TextRank algorithm by using a version that allows for weighted edges. Then the exact NPMI score between two words can be used, rather than simply treating all word pairs as either completely related or not related at all. TextRank could also use various other measures to determine which nodes are adjacent. One such measure could be derived from WordNet's vast collection of relations. We could also use dependency parse trees as a means of extracting information that bridges the gap between semantic and syntactic information.

REFERENCES

[1] Geadas, P., Alves, A., and Ribeiro, B., 2014. "Ensemble learning for keyword extraction from event descriptions". International Joint Conference on Neural Networks, pp. 2669–2676.
[2] Hulth, A. https://github.com/snkim/automatickeyphraseextraction/blob/master/hulth2003.tar.gz.
[3] Mihalcea, R., and Tarau, P., 2004. "TextRank: Bringing order into texts". Association for Computational Linguistics.

Visit my website at justinpayan.github.io