CSI5387

Data Mining & Concept Learning

Text Mining - project report

Authors :

Xiaoke Liu

Yan Zhang

Diana Lucaci

dluca058@uottawa.ca

Instructor:

Dr.Olubisi Runsewe

April 10, 2020

Contents

List of Figures

List of Tables

1 Abstract

In this project, we apply various machine learning and data mining techniques on textual data for the task of sentiment analysis. This report discusses the proposed approaches for both the classification task and for the unsupervised keyword extraction task (text mining). For classification, we report the results obtained using Naïve Bayes, SVM, and MLP. The results show that SVM outperforms the other two methods we used in our experiment. For the unsupervised task of text mining, we compare the performance of different key-phrase extraction algorithms, by building a novel architecture that selects the phrase with the highest influence on the classification task, when added to the input of a bi-LSTM classifier. We compare the results obtained on this new task using the labelled sentences and present a discussion about the proposed methodology and the conclusions of our experiments.

2 Introduction

2.1 Problem Definition

With the rapid growth of the social media platforms, the amount of textual data to be processed has become overwhelming for humans. Whether the data comes as reviews for product or movies, short text shared on the social media platforms, or blogs and news, automatically extracting insights from it has become a necessity.

2.2 Background

Sentiment analysis is one of the Natural Language Processing tasks of classifying a text as conveying a positive or negative emotion. This tasks comes in multiple flavours: binary (the task we are focusing on), multi-aspect [4], or multiple opinion polarity sentiment. For the purpose of this project, we consider the binary task and extend it to further extract insights from the corpora

formed of the instances in each of the two classes through key-phrase extraction.

Text mining (text data mining) is the process of extracting non-trivial patterns and knowledge from text documents [11]. Unsupervised key-phrase (key-term) extraction is an important task for information retrieval and text mining for performing summarization, classification, topic detection or for determining the similarity for text clustering.

The NLP task of **keyword extraction** is important for fields such as information retrieval, summarization, classification, topic detection or as a measure of similarity for text clustering. The keywords refer to single words, while a keyphrase is a multi-word lexeme that is "clumped as a unit"[5]. They have the capability of summarizing a context, allowing one to organize and find documents by content. They are both be referred to as *key terms*.

2.3 Dataset

To support and test our aproaches, we use the UCI sentiment analysis dataset [2]. This dataset contains 3000 reviews extracted from three different websites (IMDB, Amazon and Yelp). Each entry is labelled with either a positive (denoted by 1) or a negative (denoted by 0) score.

2.4 Main Tasks

This project focuses on two main tasks: cross-domain sentiment classification (as we have merged the three sources of our data in one dataset) and unsupervised key-term extraction. Thus, each section of the report will contain subsections dedicated to each of these. Our approach focuses on two main goals:

1. Comparing the performance of different classification Machine Learning algorithms (Naïve Bayes Classifier, Support Vector Machine, Neural Network) - discussed in Section 4.1;

2. Comparing multiple unsupervised statistical key-term extraction methods

by measuring the impact on the classification of the key-phrases added to the instance under classification - detailed in Section 4.2.

3 Preprocessing

3.1 Data Cleaning

To preprocess textual data, we perform a number of steps for cleaning the data.

3.1.1 Punctuation removal

Although the punctuation sometimes contains semantic nuances, the removal of the punctuation allows us to embed the words without introducing noise in the data.

3.1.2 Markup language and emoji removal

Social media data and scraped reviews may contain HTML tags, emojis or additional characters such as "
", "", "<b>" that bring challenges when tokenization is performed for the text, also bringing noise in the data in the later steps of the preprocess pipeline. Empirical results also support this claim, the removal of such characters and HTML tags significally improving the results.

3.2 Data Integration

The dataset comprises textual data from three different sources, that have been integrated and jointly used in the training process.

3.3 Data Reduction

The data is formed out of short text (movie or product reviews). For the classification task, we are using the entire text, aiming to capture the contextual semantics behind it, without altering the original text. For the text mining task,

some of the key-phrase extraction algorithms can also be adapted to summarize the instances under classification, though our focus is on key-words and key-terms that describe the main sentiments conveyed in the training set.

3.4 Data Transformation

Unlike numeric data, the textual data raises challenges for automatic processing, needing to be transformed into numerical features. Thus, data transformation is required to translate the dataset into a new format that can be interpreted by the model. The following subsections describe the techniques applied on the data.

3.4.1 Lowercasing

One of the preprocessing steps needed before embedding the text is transforming the entire dataset to lowercase. For the English language, capitalization does not bring additional semantic information (for the common words), thus this step is needed to avoid out of vocabulary words (both the words "review" and "Review" should have the same embedding).

3.4.2 Vectorization

Vectorization is the process of transforming natural language into vectors of numerical features.

One of the earliest methods for vectorization is replacing each word by its frequence in a document. The disadvantage is that there are words that often appear in numerous sentences, such as pronouns or prepositions, their weight becoming very large in the resulting embedding matrix.

TF-IDF To overcome this problem, TF-IDF score is used for text vectorization. The related formulae are:

1. TF (Term Frequency): $TF(t,d) = \frac{Number\ of\ items\ term\ t\ appear\ in\ the\ document\ d}{Total\ number\ of\ terms\ in\ the\ document\ d}$;
2. IDF (Inverse Document Frequency): $IDF(t) = \frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}$;

3. TF-IDF: $TF - IDF(t, d) = TF(t, d) * IDF(t)$.

TF-IDF not only considers the local frequency, but also the global frequency in the corpus. Thus, the frequent words that do not bring much semantic information will have relatively low weight. The words with high TF-IDF scores have a strong relationship with the document they appear in, suggesting that if that word were to appear in a query, the document could be of interest to the user.[9]

Word embeddings Another approach frequently used in the literature is to use pretrained word embeddings. We have chosen to use GloVe (Global Vectors for Word Representation [8]) pre-trained embeddings, trained on the Wikipedia corpus.

4 Model Architectures

4.1 Classification

4.1.1 Naïve Bayes Classifier

The Naïve Bayes classifier is a probabilistic classifier based on the Bayes' theorem. Bayes' theorem describes the probability of an event, given some prior knowledge, using the formula: $P(Class|Sentence) = \frac{P(Sentence|Class)P(Class)}{P(Sentence)}$. By determining the posteriori probability of *Class*, conditioned by Sentence, we can classify a given instance.

Model Construction After preprocessing the text and vectorizing it using TF-IDF, we instantiate a new multinomial Naïve Bayes model using *scikit-learn*[7]. We trained the model according to the default settings, with smoothing factor=1.0 and predicted the *class* probability.

Discussion This model performs well, obtaining above 80% accuracy and precision, which means that both TF-IDF and the probabilistic learning process

are able to capture the semantics of the text under classification.

4.1.2 Support Vector Machine

The Support Vector Machine (SVM) algorithm is a linear classification model. Although it performs very well on linear data, it can also be applied on nonlinear data. The SVM method is designed to search for the optimal linear separating hyperplane using support vectors and margins. The support vectors are training tuples that are correctly labelled, while the margins are defined by the support vectors from different classes. The optimal hyperplane is the one for which the margins are maximal, thus separating the data points under classification.

Model Construction We have experimented with a linear kernel with a coefficient of $\frac{1}{\#offeatures}$ and a regularization parameter of 1.0, using the *scikit-learn* ML library once more.

Discussion Compared to the previous approach, the linear SVM model seems to perform better, which translates into the representation of the instances under classification being linearly separable in the multi-dimentional space defined by the TF-IDF vectors.

4.1.3 Neural Network

We have also applied a neural network technique to classify the reviews. For this model, we have trained embeddings specifically for the text under classification, with the goal of obtaining word embeddings that are suited for the informal style of writing present in this dataset. The word embeddings obtained using this mechanism have multiple advantages. One of the major advantages is that words that have the same semantic meaning have similar representations, which is expected to improve the classification performance.

Model Construction For this model, we experimented with simple architectures presented in Table 5 from the Appendices. We have used a sentence length

of 74 words, and the dimension of word embeddings 32. The fully connected dense layer has 250 nodes, while the output layer has one neuron (as we are working on binary classification).

Discussion This simple architecture underperforms, according to the results reported in Section 5, the model not being able to capture the semantic complexity of the cross-domain text under classification. To improve its performance, there are a number of approaches to be considered as future work such as increasing the embedding dimension, number of training epochs (with mechanisms of early stopping), extending the architecture to multiple layers, and experimenting with different activation functions.

4.2 Keyword and Key-phrase Extraction

We started our approach with a baseline, consisting of a bi-LSTM classifier that classifies the input text embedded with GloVe pre-trained embeddings.

We will refer to corpus as the collection of all the training instances for a class: positive and negative. To gain a better understanding of the semantics of each class's texts, we further applied different statistical and graph-based text mining approaches to extract a dictionary of 300 key-words and key-phrases for each of the two corpora (positive and negative). As an alternative, we also report the results obtained by applying these techniques considering each review as a stand-alone document.

While for the implementation of the algorithms described in the following sections we have used python libraries (*yake*, *rake-nltk*, and *pytextrank*), our contributions focus on the design of the evaluation model described in Section 4.3.

4.2.1 Statistical methods

TF-IDF One of the most common numerical statistic is TF-IDF (described in section 3.4.2). We use this approach to compute the score of each term in a

review (document), ranking the scores of the resulting keywords and preserving the top 300 candidates.

RAKE (Rapid Automatic Keyword Extraction) [10] is a statistical domain-independent approach that uses both the frequencies and the co-occurrences of the words. The score of each word is computed as the ratio between its frequency and the degree of the word in the matrix of co-occurrences (the sum of the number of co-occurrences the word has with any other word in the original text).

YAKE [1] is a language-independent single document keyword extraction algorithm that uses the following features: Casing, Word Positional, Word Frequency, Word Relatedness to Context, and Word DifSentence to heuristically compute a score for each keyword candidate. For n-grams, the score is divided by the term frequency of the keyword. The resulting similar candidates are filtered out based on the Levenshtein [3] distance and ranked based on the score, outputting 1,2 and 3-grams.

4.2.2 Graph-based methods

TextRank TextRank is an unsupervised graph-based method proposed in 2004 [6]. This method builds a graph where each node represents a lexical unit (one word), as a keyword candidate. The edges between the nodes are added when the words are situated in a window of maximum N words in the original document, where N is a hyperparameter. The scores computed based on the number of edges between the nodes are then sorted in decreasing order and the top k keywords are extracted. The hyperparameter k is set to a third of the number of the total words in the input text. In a post-processing phase, the selected keywords that are adjacent in the original document are grouped to form multi-word expressions.

4.3 Sentiment Analysis Keyword Generator

After building the dictionary of possible key-terms candidates, we design a MLP generator that outputs a probability distribution over the dictionaries. The one-hot encoding of these distributions are further used to select the key-phrase that the baseline bi-LSTM classifier uses to improve its confidence on the classification task. This choice is fine-tuned through the MLP design in the training process on the classification task. The optimization problem that we are trying to solve is defined by minimizing the loss function: $\mathcal{L}(e_{emb}, s_{emb}, y) = \|\mathrm{C}(e_{emb} \cdot sep \cdot s_{emb}) - y\|^2$. For a better understanding, we provide an architectural diagram in Figure 1.

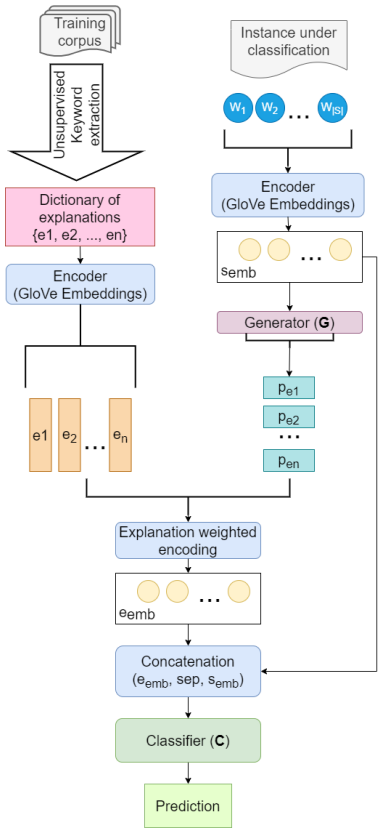

Figure 1: Generator architecture

5 Evaluation

5.1 Classification

For the classification task, we have reported the results in Table 1.

	Accuracy	Sensitivity	Specificity	Precision	AUC
Naïve Bayes	82.17%	84%	81%	81%	89.89%
SVM	83.67%	85%	83%	83%	91.19%
MLP	77.33%	76%	79%	80%	85%

Table 1: Classification evaluation

SVM outperformed both Naïve Bayes and MLP in all the metrics reported, the F1 score (0.84) also being the highest among all models. This indicates that SVM has certian robustness against the other two models. To further prove that SVM outperforms the other models, we have analyzed the ROC (Receiver Operating Characteristic) curves that can be found in the Appendices.

5.2 Keyword and Key-phrase Extraction

5.2.1 Quantitative Analysis

This analysis provides insights about the quality of the extracted key-terms by analyzing the improvement in performance the generator architecture manages to gain when learning to choose the best candidate from the extracted dictionaries.

Since all the other hyperparameters and variables are invariable in the reported experiments, what it is of interest in this analysis is the performance difference compared to the baseline. As we can observe from Table 2, the best accuracy is obtained with the YAKE dictionary. This approach is most accurate in terms of precision, but it underperforms in terms of recall. The best recall and balance between precision and recall (F1-score) is obtained using the RAKE algorithm. This conveys that YAKE is able to correctly classify a subset of the dataset very well, identifying some patterns in the data, while the higher

RAKE's performance conveys that word co-occurences are more important for this dataset.

It is also worth noting that jointly training the model on the mixed-source dataset is a challenging task, increasing the difficulty of extracting a key-phrase correlated with the review, since the dictionary also contains mixed phrases (such as movie reviews, product, or service feedback). The proportion of the source of the key-terms is not enforced for each subdataset, but only for the label (300 entries for each class).

Model	Dictionary	Accuracy	Precision	Recall	F1
Vanilla (bi-LSTM)	-	82.68	0.8449	0.7938	0.8135
bi-LSTM+ MLP_gen	TF-IDF	79.44	0.8619	0.7612	0.7828
	RAKE_corpus	82.51	0.8253	0.8158	0.8169
	RAKE_instance	82.68	0.8276	**0.8168**	**0.8186**
	YAKE	**83.00**	0.8501	0.7919	0.8153
	TextRank	82.84	**0.8629**	0.7788	0.8139

Table 2: Keyword extraction evaluation

5.2.2 Qualitative Analysis

To further understand the performance of the compared text mining algorithms, we propose some qualitative analysis methods such as exploring the obtained key-terms (through visualizations such as cloud-words - examples in the Appendices C - or through the resulted associations between the instances and the generated keyphrases - Table 3).

Review	Class	Selected Key-term	Dictionary
we will not be coming back	0	worst people behind	RAKE
worked perfectly	1	good product incredible	YAKE
the bacon is soooo good	1	chef	TextRank

Table 3: Review-keyphrases examples

Furthermore, in Table 4, we also report statistics such as the average number of words for each dictionary individually, combined dictionaries and the number of key-terms present in both dictionaries (overlap count).

Combining the qualitative and quantitative analysis, we can observe that the best accuracy (and high precision) is obtained by 1 and 2-grams, while longer key-terms (of an average of 3.98-4.3 words per keyphrase) are more suitable for improving the recall of the baseline.

Dictionary type	Average words/ keyphrase (dataset)	Avg words/ keyphrase (pos)	Avg words/ keyphrase (neg)	Overlap count
TF-IDF	1	1	1	21
RAKE (corpus)	3.61	3.74	3.48	0
RAKE (instance)	4.14	4.3	3.98	0
YAKE	1.85	1.89	1.82	23
TextRank	1.13	1.15	1.11	5

Table 4: Keyphrase analysis for dictionaries of 300 entries

6 Conclusions

Our project presents the application of machine learning algorithms that are traditionally used on numerical data on a text classification task for sentiment analysis. We discover the TF-IDF vectors are separable in the hyperspace defined by them, SVM performing very well. We not only report the results obtained with algorithms from different classes: probabilistic, linear, and neural networks, but also introduce a new evaluation technique for the unsupervised task of keyword extraction. We thus implement and discuss the results obtained using statistical approaches of text mining, combined with advanced deep learning techniques for text processing.

Our work can be further extended in the area of Explainable AI (XAI), where the mined key-phrases act as explanations of the deep learning classifier.

References

[1] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. A text feature based automatic keyword extraction method for single documents. 02 2018.

[2] Dimitrios Kotzias, Misha Denil, Nando de Freitas, and Padhraic Smyth. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 597–606, New York, NY, USA, 2015. Association for Computing Machinery.

[3] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[4] Bin Lu, Myle Ott, Claire Cardie, and Benjamin K. Tsou. Multi-aspect sentiment analysis with topic models. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ICDMW '11, page 81–88, USA, 2011. IEEE Computer Society.

[5] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[6] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[9] Juan Ramos. Using tf-idf to determine word relevance in document queries. 01 2003.

[10] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. *Automatic Keyword Extraction from Individual Documents*, pages 1 – 20. 03 2010.

[11] Ah-Hwee Tan, Kent Ridge, Digital Labs, and Heng Terrace. Text mining: The state of the art and the challenges. 11 2000.

Appendices

A MLP architecture

The MLP model architecture for the classification task can be visualized in Table 5.

Layer(type)	Output shape
Embedding	(None,74,32)
Flatten	(None,2368)
Dense	(None,250)
Dense	(None,1)

Table 5: MLP architecture diagram

B Receiver Operating Characteristic curve

According to the given metric of each model, we can plot the ROC curve. By observing the curve of each plot, both models have almost the same curvature

in the beginning of the curve. Every curve tends to have an dramatic increase in ROC curve between 0 and 0.1. The Area Under Curve and the curve of the plot support the claim that the SVM classifier is the best among our compared models, irrespective of the chosen metric of comparison.

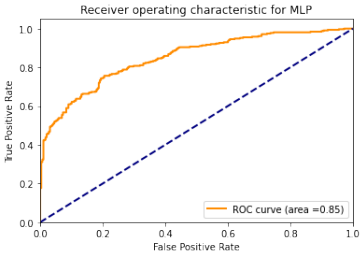

Figure 2: ROC - MLP

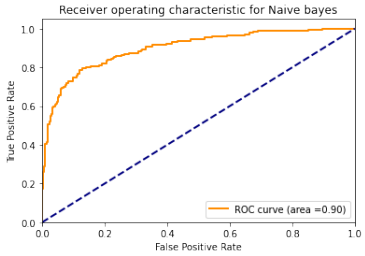

Figure 3: ROC - Naïve Bayes

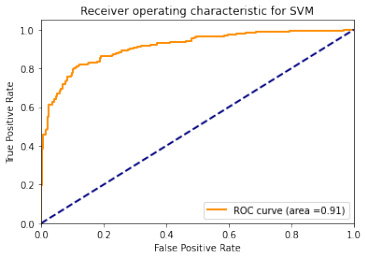

Figure 4: ROC - SVM

C Keyword generation - qualitative analysis

One of the visualizations options that can be exploited for a further qualitative analysis is the word-cloud technique. Figures 5 and 6 exemplify the results obtained using the YAKE text mining algorithm and our proposed generation model.

Figure 5: YAKE - positive

Figure 6: YAKE - negative