Text Classification

Udacity Machine Learning Nanodegree Capstone Proposal

By Prasann Pandya

Domain Background

Understanding context from a given document by a computer is an age old problem under the field of natural language processing. Since my interests lie in the field of natural language processing, I decided to choose one of the fundamental tasks of natural language processing: **Document classification** or **document categorization**. Document classification is an age-old problem in information retrieval, and it plays an important role in a variety of applications for effectively managing text and large volumes of unstructured information. Automatic document classification can be defined as content-based assignment of one or more predefined categories (topics) to documents. This makes it easier to find the relevant information at the right time and for filtering and routing documents directly to users. The task is to assign a document to one or more classes or categories.

Every news website, scientific journals, online digital library, etc. needs to categorize document so that it can be easily found by users. Usually, this is done by manually screening and tagging of documents by humans. This task is not only time consuming, it can also be error prone and subject to bias as one person’s categorization can be different from other person’s categorization. Also, the screener may not be familiar with a particular topic. Since this task is redundant, manual and labour intensive (requires lot of reading), I decided to use Machine Learning techniques to automate this task. Machine Learning for classifying and categorizing documents can save time, improve accuracy and also remove bias.

There are many other applications for a document classification software namely: spam email filtering, genre classification (automatically determine genre of text), sentiment analysis, help librarians to categorize scientific papers by providing additional information beyond authors' keywords, assigning disease code in medical documents, etc.

There are many different approaches that have been used previously to tackle this problem previously. However, there is not standardized approach as far as I have found to classify documents. Thus, my purpose for this project is to try different supervised and unsupervised learning approaches to find the best model to classify documents.

Problem Statement

The problem I will be working on is automatic document classification. Document classification is a problem faced by all scientific journals, news organizations and digital libraries. The software will use documents as input and the software will automatically recognize which category it is related to from a list of categories. The performance of the software will be measured by how many documents it classifies correctly (accuracy) and also F1 score (precision and recall).

Datasets and Inputs

The dataset I will be using to solve this is popular 20 Newsgroups dataset. It is one of the datasets available in sklearn. The 20 newsgroups dataset comprises around 18000 newsgroups posts on 20 topics. Each document is labelled to be in one of the 20 categories.

Information on how to fetch 20 Newsgroups dataset from sklearn: http://scikit-learn.org/stable/datasets/index.html#the-20-newsgroups-text-dataset

This is a list of the 20 newsgroups:

1. comp.graphics
2. comp.os.ms-windows.misc
3. comp.sys.ibm.pc.hardware
4. comp.sys.mac.hardware
5. comp.windows.x rec.autos
6. rec.motorcycles
7. rec.sport.baseball
8. rec.sport.hockey sci.crypt
9. sci.electronics
10. sci.med
11. sci.space
12. misc.forsale talk.politics.misc
13. talk.politics.guns
14. talk.politics.mideast talk.religion.misc
15. alt.atheism
16. soc.religion.christian

I will be training and testing my models using this labelled data.

Solution Statement

The solution to this problem of classifying document is a model that can take documents as input and automatically classify them according to their category. For example, it can take in any of the news document as input and label it as “electronics” or “space” or “politices”, etc.

Benchmark Model

The traditional way of performing text classification is to use a bag of words model, convert the words to vectors using TF-IDF and using a classifier mainly SVM. This usually gives an accuracy of around 90%.

Evaluation Metrics

Most text classification models are measured on accuracy. However, I want to use both Accuracy and F1 score as evaluation for the model since F1 score can provide better understanding of overall performance while taking into account the false positives and false negatives.

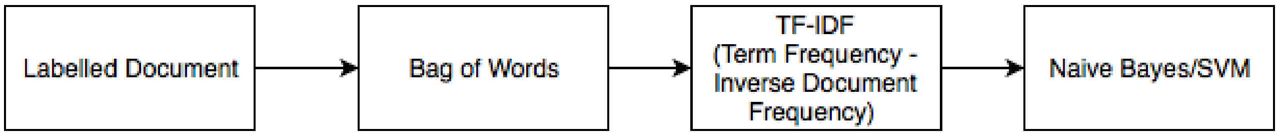
Project Design

The project design contains three steps:

1. Preprocessing (Feature Extraction)
2. Training Model (SVM/NB)
3. Classification

There are two different approaches that can be taken for extracting features from documents and converting them into word embedding.

First Approach: Using Bag of Words Model



All the words in a labelled document will be converted to bag of words. This means that each document will be segmented into words. Then all the words will be converted to vectors by finding occurrence of each word in the document. To normalize this vector so that the model is not biased towards longer documents (longer documents will have higher average count values than shorter documents), we will convert the counts into frequency of the word in the document. Another refinement on top of this is to downscale weights for words that occur in many documents in the corpus and are therefore less informative than those that occur only in a smaller portion of the corpus (words such as the, an, is, for, etc.) [2]. This whole process is known as finding "Term Frequency-Inverse Document Frequency (tf-idf)". Sklean has a in-built function to automatically convert words into tf-idf vectors.

Once all words in a particular document are converted to vectors, a classifier (NB/SVM) will be trained on the data. The data will be separated into training and testing sets using crossvalidation. Whichever classifier provides better performance in terms of Accuracy and F1 score will be chosen.

Second Approach: Using Doc2Vec Model

Bag of Words model has many disadvantages. The word order is lost, and thus different sentences can have exactly the same representation, as long as the same words are used [3]. To combat this, a new approach called Doc2Vec (also called paragraph vector) was introduced. In this approach, word order is taken into consideration. The vectors of words with similar context (surrounding words) are similar and thus the word meanings are taken into account in this approach. This Doc2Vec provides numerical representation of a document which represents concept of the document. More details on Doc2Vec approach are in [3].

Doc2Vec can be implemented using a library called Gensim. This library provides an efficient way to convert each document to a vector. Once trained on each category, Doc2vec gives the probability of how likely a new document belongs to a particular category.

For every test document, the test vector can be compared to the doc2vec model of each category. Whichever category gives the highest probability, the test document belongs to that category.

Conclusion

After trying out all the approaches, Accuracy and F1 scores will be used to decide the best model.

References

1. Evaluation of Text Classification (Stanford NLP Group): https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-text-classification-1.html
2. Working with text data: http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
3. Distributed Representations of Sentences and Documents: https://cs.stanford.edu/~quocle/paragraph_vector.pdf
4. Gensim Doc2Vec tutorial: https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/doc2vec-IMDB.ipynb
5. Document Classification Wikipedia: https://en.wikipedia.org/wiki/Document_classification