Ashesi University | Natural Language processing | Fall 2018

WRITE UP

# PROJECT SUMMARY:

This project is an attempt to analyze various natural language implementations to see which worked better for different tests. This was done in two parts. Topic Modelling and Question Answering.

Topic Modelling involved the user giving a question to the program which in turn should be able to

print the topic associated with the question. Note that the question used to test your program may not have appeared in the training data exactly. For instance, the training data may contain “Who is the president of Ghana” but the test question might be “Who is Ghana’s head of state?”. If the topic for “Who is the president of Ghana?” is “Ghana government”, we expect the topic for “Who is Ghana’s head of state?” to be “Ghana government”.

Question Answering involved the user giving a question to the program which in turn should be able to print the answer associated with the question.The notice provided about the nature of the test questions under Topic Modeling would apply here. Thus, if the answer to the question “Who is the president of Ghana?” is “Nana Akufo Addo”, then the answer to “Who is Ghana’s head of state?” should be “Nana Akufo Addo”.

# DATA SET

The training dataset for this project were three files sourced from the Natural Language Processing Class at Ashesi University. One file consisted of Questions, another consisted of the answers for these topics these questions were under and the last contained the answers to these questions. These files contained 2608 examples in total.

80 % of the data was used to be used for training and 20% for validation. This equate to 2086 examples for training and 521 in the validation set. This was the same across all the various models in order to ensure we were best able to compare the performance of each model.

TOPIC MODELLING

Methodology:

As indicated in the project summary section there were two separate parts of our project. Topic Modelling and Question Answering. The first section tackled was topic modelling as it would better inform the ability a model to answer questions. The goal for this section was to find a model that would best illustrate the following: given a question, the program returns the topic associated with the question.

A range of algorithms have been developed for topic modelling approaches. The majority of these applications are done by detecting patterns in a corpus and grouping these words into a various topics.

Naive Bayes is the most commonly used algorithm for text classification its speed and ease of implementation make it incredibly desirable(CITE). However, there isn’t much knowledge about how it is used for multi-text classification and it can be prone to error if its parameter estimates are poor (Rennie, 2001). In fact, it has placed last numerous times in head-to-head classification problems which speak to the flaws that may arise from the severe assumptions it makes (Rennie, Shih & Karger, 2003). However, despite these flaws given its success in different in natural language processing even with small datasets we wanted to analyse its performance.

Logistic Regression, is similar to Naive Bayes in that it too is an algorithm used extensively in applications for natural language processing. Unlike Naive Bayes which is generative, it is a discriminative classifier. However this algorithm is very suited for discovering the link between features or cues and some particular outcome (Jurafsky). This knowledge brought the group to a realization that it would work best at choosing the most likely features in the questions and assigning them to a topic. It was this reason and the fact that logistic regression is so linked to supervised machine learning such as machine learning algorithms for classification that made us decide to use this as one of our approaches.

The Latent Dirichlet Allocation is the most popular topic modelling approach and is used extensively in the field. It is a generative model that assumes “each word in a document is generated from a topic that in turn is picked from the topic distribution for each document” (Risch, 2016(. The model is so popular it is used in the clustering of Google News articles and “describes a document in terms of a probabilistic distribution over words” ( De Smet & Moens, 2009). Risch used the LDA’s in detecting twitter topics, which involved detecting topics based on a short messages of less than 140 characters (Risch, 2016). This is a helpful as the questions, we seek to the model on are also short so we may expect similar results. Unfortunately, the main issue with this model is that the data given only contains 2608 samples which is too small to measure LDA’s. Despite this we wanted to better understand even with a small dataset the Latent Dirichlet Allocation would compare to other models. Despite this fact we felt that testing it out would help us to understand more on our data set.

Latent Semantic Analysis (LSA for short), is a technique which is part of the foundation in topic modeling. In short, the main idea is to take a matrix documents and terms and decompose it into a separate document-topic matrix and a topic-term matrix. This model, like the others is built around the idea that the semantics of our document are governed by some hidden variables that we are not observing directly. The aim of the model is to uncover these latent variables, which are topics, which shape the meaning of our document and corpus.

Next, we will give a brief idea on how we decided to implement the three different ideas.

Implementation:

**Naive Bayes implementation:**

In our implementation, we used the sklearn naive Bayes classifier as the core of our code. Using the questions as our X input and the topics as our Y classes, we tackled the problem using what we learnt from sentiment analysis. As such, our classifier used a TFIDF vectoriser to clean and vectorise the data (converted to UTF-8 during the file reading to normalise our input data). The data was first read line by line into 3 different arrays. Each line was then concatenated with its corresponding question, answer and topic, separated by a tab “/t” character, and written to a new file. This file was then read by a pandas CSV reader and split into 3 columns using the tab character. This was done to make manipulation simple given that pandas’ CSV reader puts the data onto a pandas dataframe which is compatible with the TFIDF vectoriser. Each column was assigned to a representative variable, allowing us to use the data with ease. The data was randomised using a random seed generator and split into training and testing data. Once done, it was passed into the NB classifier for training.

**Logistic Regression**

Our logistic regression implementation used sklearn’s Logistic Regression classifier to perform the regression. We started first cleaning the question data with nltks’ built-in methods of Lancaster stemming and wordnet lemmatization. Vectorization was also done on the question data set using the Count Vectorizer, a model to identify features or occurrences. The data was trained with the questions as X data and the topics as Y data. Using panda to read csv files we had to specify an encoding type .This was so the data passed could be in a .txt format. We put the two file together ie the questions and topics file. We then split it 80, 20 with the training set having 80 percent of the total and the remaining 20% being used for the test set. In order to measure the performance of the classifier

**Latent Semantic Analysis:**

For the LSA first step is generating the document-term matrix. Given y documents and n words in the vocabulary, we construct a y × n matrix A in where each row represents a document and each column represents a word. LSA models replace raw counts in the document-term matrix with a tf-idf score. Tf-idf, or term frequency-inverse document frequency, assigns a weight for term j in document i as follows:

The more the term appears in the document, the smaller its weight, and the more infrequently it appears across the corpus, the greater its weight.

Moving on to the latent topics in all likelihoods, *A* is sparse, noisy, and almost redundant across its many dimensions. As a result, to find the few latent topics that capture the relationships among the words and documents, perform dimensionality reduction on A.

The reduction is performed by using a truncated singular value decomposition. It factorizes any matrix into the product of 3 separate matrices: M=U\*S\*V, where S is a diagonal matrix of the singular values (non-negative real numbers) of M. Truncated SVD reduces dimensionality by selecting only the t largest singular values, and only keeping the first t columns of U and V.

This makes matrix A≈ UtStVt . This is like only keeping the *t* most significant dimensions in our transformed space. (Landauer, T. K., McNamara, D. S., Dennis et al, 2013)

Using these document vectors and term vectors, we apply cosine similarity to evaluate:

* the similarity of different documents
* the similarity of different words
* the similarity of terms and documents (which becomes useful in information retrieval, when we want to retrieve passages most relevant to our search query).

Results:

**Naive Bayes implementation:**

Our classifier achieved a highest accuracy of 60% in our testing phase. We consider this an acceptable result given that it performs better than a random or static baseline classifier. Considering that the classifier has to learn the difference between almost 100 different classes, we believe it far outperforms baseline functions.

Although this performs better than baseline estimates, we consider this only to be a good starting point for our classification model as 60% is still not good enough for a production ready model.

**Logistic Regression Implementation:**

After using the logistic regression for topic modelling, our accuracy was ~0.78, we think it could have been made better if we wrote a method to clean the data ourselves. Data on which we trained could have been larger to increase accuracy.

**Latent Semantic Analysis:**

Using the Latent Semantic Analysis classifier, the highest accuracy score was 60% in our testing phase. This result is similar to the naïve bayes classifier so is satisfactory for the time being. As the classifier learnt the difference between approximately 125 different classes, it is clear that it outperforms baseline functions once again.

QUESTION ANSWERING

Methodology

As indicated in the project summary section there were two separate parts of our project. Topic Modelling and Question Answering. The first section tackled was topic modelling as it would better inform the ability a model to answer questions. The goal for this section was to find a model that would best illustrate the following: given a question, the program returns the answer associated with the question.

We decided to use the same algorithms as described in the topic modelling section: These were Naive Bayes, Latent Semantic Analysis and Logistic Regression. This was done to see if we’d get similar performance and reduce complexity.

Implementation

**Naive Bayes classification:**

Here we implemented Naive Bayes again as a starting function to see how well a classifier could perform. In doing so, we utilised the same function used for topic modeling to reduce complexity. In this implementation, we used questions as the X value and answers as the Y classes. The processing procedure was the same as in the topic modeling.

**Latent Semantic Analysis:**

Similar to the implementation of the topic model, the data was vectorised into a three dimensional matrix and the the layout of the matrix constituted of the words in the training data set, the documents they fall under and further, the words were laid out in the matrix with similar words being close to each other. How often a word or phrase appeared was recorded at the intersection of the words and the documents in the matrix. The algorithm, after doing this uses singular-value decomposition which creates a factorized form of the matix to reduce the matrix. The results from these were used in creating a question and answering model.

**Logistic Regression:**

As in the implementation of the topic modelling for logistic regression that for the question and answering model was similar with the questions and answers serving as our training data X and Y respectively.

## 

## Results

**Naive Bayes Implementation:**

Our naive Bayes classifier achieved am accuracy of 14% in question answering. We believe this is as a result of the number of classes being increased from almost 100 to as many answers as there are questions given that each question has a unique answer. This greatly reduced its ability to generate unique class models for each answer since the questions were nearly identical as far as it was concerned, with only a few key words per question to suggest it belonged to a given question. This perhaps could be mitigated if cosine similarity is used to find the most similar question in the corpus and that question is used to predict the best possible answer.

Another odd behaviour this classifier has was that it predicted the answer “yes” for each question asked. We assume that this is because “yes” was the simplest answer and the easiest one to predict for any question. It is however peculiar and was quite a surprise to see during testing.

**Logistic Regression Implementation:**

Using logistic regression for our question and answering model, the dataset trained on was too small thus, our model’s accuracy was only up to 0.21.We inferred this could also be due to the fact that we used built in library to clean our data.In future works we hope to get better results if we train our data on a wikipedia article or a passage.In conjunction with cosine similarities and euclidean distance to be able to capture the variants of the questions our model was trained on better.

**Latent Semantic Analysis Implementation:**

Latent Semantic Analysis model predicted the most likely question in the training data instead of returning an answer to an input question. The accuracy when the model was tested on the training data was 98%, meaning that the model was able to recognize majority of the questions it was familiar with. A second test was done on the testing data using the criteria that if the cosine similarity between the test question and the predicted question was greater 0.5 it would be considered a correct prediction. The accuracy for that was 20%.

|  |  |  |
| --- | --- | --- |
|  |  |  |

References

De Smet, W., & Moens, M. F. (2009, November). Cross-language linking of news stories on the web using interlingual topic modelling. In *Proceedings of the 2nd ACM workshop on Social web search and mining* (pp. 57-64). ACM.

Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (Eds.). (2013). *Handbook of latent semantic analysis*. Psychology Press.

Risch, J. (2016). Detecting Twitter topics using Latent Dirichlet Allocation.

Rennie, J. D. (2001). Improving multi-class text classification with naive Bayes.

*Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. In Proceedings of the 20th international conference on machine learning (icml-03) (pp. 616-623).*