Augenstein (01607626), Haimbuchner (01612874), Ivanova (01153937), Jusufi (01626135)

1640 – Data Processing 2

COOL

Clustering On Online Linguistics

The main goal of this project is to get an overview of what people discuss when they tweet about Elon Musk. To accomplish this, first tweets about Elon Musk are streamed and collected. After applying natural language processing, the data is clustered using Word2vec to produce word embeddings and clusters to display the most frequent topics related to Elon Musk at one glance.

1. High Level Description and Data Sources

The project is divided in three main steps: the streaming, the processing of the tweets and the formation of clusters.

Streaming

The streaming was accomplished by using the API provided by Twitter. It was necessary to request a Twitter Developer Account in order to be able to use the Twitter API. For the streaming itself Spark and Kafka were combined. Dr. Sabrina Kirrane provided the code for a KafkaProducer as well as a KafkaConsumer.

By using a keyword in the code, it is possible to stream only tweets that are filtered by this keyword. In this case, “elon” served as the keyword as @elonmusk is the twitter username of Elon Musk.

Since there was some trouble with processing the streamed data directly, the tweets were stored in a large text file. This file serves as the input for the further processing activities.

Natural Language Processing

The code used in this project was presented in class by Dr. Sabrina Kirrane. The natural language processing of the unstructured tweets was accomplished using the Natural Language Toolkit, also known as NLTK. NLTK is a collection of libraries for natural natural language processing (NLP) in Python. The main objective using NTLK was to analyze the tweets and to extract only the keywords and the most important information from them to enable effective clustering of the tweets into groups.

Clustering

For the clustering itself Word2vec by Mikolov et al. was chosen. Word2vec is a two-layer neural net that processes text and is implemented into TensorFlow. It takes unstructured text like the tweets in this project as an input and delivers a set of vectors as its output. The algorithm assigns each word a vector based on its characteristics and creates word embeddings based on how close or distant the different vectors/words are to one another.

2. Code Description

Streaming

The TwitterProducer notebook streams the tweets via the Twitter API. For this purpose, Tweepy is used as an easy-to-use Python library for accessing the Twitter API. It does authentication via the OAuth which is the only way to use the Twitter API. That is where the consumer keys and access tokens come into place. This enables the user to make use of the methods and be able to get any object that the Twitter API offers. The StreamListener object enables the handling of the received tweets from the stream. In this case the stream uses filter() to track “elon” and English tweets only.

The KafkaTweetConsumer listens then to a particular Kafka topic. As mentioned before, having some trouble with processing the streamed data directly, the tweets were stored in a large text file, instead of directly processing the DStream.

Natural Language Processing

For the natural language processing, only the text of the tweets is needed. Each tweet is transformed to rows of a data frame. First, the stop words are removed, using the stop words list included in the Natural Language Toolkit. Subsequently, further non-essential words are excluded which can be chosen according to the purpose of the NLP.

The next step classifies words into their parts of speech and keeps only nouns, verbs and adjectives. The lemmatization is used to analyse the vocabulary and morphology of words. It removes endings and returns the base of a word.

As some tweets might have consisted only of words which have now been excluded, it is useful to once again check for missing values in the data frame.

Clustering with Word2Vec

In order to begin working with Word2Vec, the processed tweets need to be tokenized. This means that the text – set of terms – is taken and converted into fixed-length feature vectors. As the model provided by Kavita Ganesan requires a list of lists (the second list being the list of words that tweets consist of), the data frame, used for the NLP, has to be converted to the according format.

In the next step, we are training a neural network with a single hidden layer in order to predict the current word based on the context. The further used part of this is the resulting learned vector, which is also known as the embeddings. Lastly, one can look for words similar to whichever word is of interest, which in this case means that the words were used in similar context.

3. Licenses and Ethical Issues

To be able to work with the Twitter API and the Apache Framework, the Twitter Development Agreement and Policy as well as the Apache License (Version 2.0) were important to consider.

The Twitter Development Agreement and Policy states that Twitter grants a "revocable license to access, index, and cache by any means (...) Twitter Content using embedded Tweets or embedded timelines". Since the goal of this project was just to process tweets without having any intention to sell or publish non-public information, there were no issues with this licence.

We found out that we can use the Apache License as well, since Apache Licence Version 2.0 grants a "perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form".

The full content of the two licenses can be accessed through the following links:

- https://developer.twitter.com/en/developer-terms/agreement-and-policy.html (Twitter Development Agreement and Policy)
- https://www.apache.org/licenses/LICENSE-2.0 (Apache License - Version 2.0).

Aside from licenses, there are also potential ethical issues to consider. But as already mentioned above, this project only classifies tweets according to their similarity and no further analysis is done.

4. Credits

At this point it should be mentioned that a major part of the code that was used in this project was written by others. Therefore, this section shall be used to give credit where credit is due.

As already mentioned above, the code for Spark and Kafka Streaming was provided to us by Dr. Sabrina Kirrane. The code for the Natural Language Processing with Spark using the NLTK was found on https://github.com/dreyco676/nlp_spark/blob/master/nlp_with_spark.ipynb.

The Word2vec was created by a team of researchers led by Tomáš Mikolov. The implementation of Word2Vec in Python using TensorFlow can be found on https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/word2vec/word2vec_basic.py.

The concrete adaption of the Word2Vec for this project was accomplished using code and tutorials provided by and with explicit permission from Kavita Ganesan. This can be seen in the submitted file "email_response". The code was downloaded from the GitHub repository on https://github.com/kavgan/nlp-text-mining-working-examples/tree/master/word2vec. Further explanations can be found in Kavita Ganesan's blog post "Gensim Word2Vec Tutorial - Full Working Example (http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.XFHO8M9KhZJ) where she also provides a link to her Jupyter notebooks.

5. Limitations and Lessons Learned

Unfortunately, we were not able to fully realize the project the way it was originally planned. One of the issues we stumbled across was that we were unable to find a way to transform the DStream the Twitter Producer generated into a suitable format for the Natural Language Processing. Various approaches to solve this problem were tested until we realised that a potential approach would be working with micro-batches, distributing the stream and further processing it. However, in the remaining time after we discovered this solution we did not manage to fully understand how this process works.

The transition from one notebook to another caused a lot of trouble over the course of this project. This taught us that converting the data into the right format is crucial to achieving an automatic workflow across different algorithms and parts of the code.

We also noticed that not all methods and algorithms that are available are scalable and it is important to understand how they work in order to choose the best one for the specific purpose. Encountering this issue prompted us to make heavier use of the official websites, which explain the basic functionalities and available variables in detail unlike forums such as Stack Overflow.

A rather surprising limitation we came across is that there were not as many tweets about Elon Musk as we expected. However, the algorithm also works with only a few tweets.

In conclusion, this project showed us that distributed file systems are very complex. In order to be able to realize a project like this it is essential to understand the framework as a whole very well. Especially at the beginning of the project we had some difficulties dealing with this complexity. However, in the end things turned out quite well.