Universidad Carlos III de Madrid

Bachelor's Degree in Data Science and Engineering

Machine Learning Applications Course

Leganés - May 2021

Final Project Report

Enrique Botía Barbera, David Méndez Encinas, Andrés Ruiz Calvo,
Simón E. Sánchez Viloria

Index

Introduction

For the course's final project, we were tasked to conduct an extensive exploratory, semantic, and predictive analysis of a text-based dataset of our choice. We relied on the Machine Learning and Natural Language Processing techniques that we learned along the course, as well as our research and experience to carry out the experimentation and implementation of this analysis.

The following report is meant to provide a detailed summary of the work done in this project and the methodology followed to accomplish each of the required tasks. The first section of this report describes how we selected and created the dataset that we used for the analysis. The second one describes the methods used to preprocess the dataset to perform semantic analysis with it. The third section details how vector representation and topic modeling was performed using this preprocessed data, while the fourth one describes how we obtained a semantic graph from it and what it represents. Finally, the fifth section explains the methodology used to train and evaluate various machine learning models to carry out the predictive analysis from the semantic representation of the data.

1. Dataset Creation

We initiated the first phase of our project by searching for an adequate dataset that would fit the requirements of the project. The dataset had to be appropriate for performing NLP tasks so largely text-based and additionally contain a variable that could be used for regression or classification.

After considerable research and experimentation, we decided to go with building our dataset from posts scraped from the website Reddit. Where our text-based variable would be the text content of a given post published to the website and as the variable to classify the "subreddit" where the post was submitted from.

To get a better idea of why we chose a dataset based on data gathered from Reddit, it is important to acknowledge that Reddit is a forum-based website. It allows users to post content in public communities or forums denominated *subreddits*[1] and encourages other users to discuss it between them.

Each subreddit normally represents a distinctive subject going from very specifics like "GameOfThrones", "DataScience" or "SpaceX", where the posts and discussion are very limited by the specificity of the topic to more generals like "CasualConversation" and "Funny" where the diversity of content is much higher. This natural clustering of highly text-based content in Reddit is what led us to the idea of basing our project on data obtained from this website.

To build the dataset we relied on Reddit's official API wrapper for python PRAW, which provides an easy-to-use interface to extract submissions and comments from the website as well as other automatization tasks. Based on this wrapper we built a *get_submissions_df* function that given the name of a subreddit it returns a given *k* number of posts from that community.

With the function defined above, all that it was left to do is selecting which subreddits we would use for our task. We went on with 10 different subreddits ranging from very specific to more diverse, we were also careful to select subreddits that would encourage textual submissions in order to avoid scraping image or video-based posts that would end up being irrelevant.

[1] Throughout our analysis we use the terms *forum, community* and *subreddit* interchangeably to refer to the same concept.

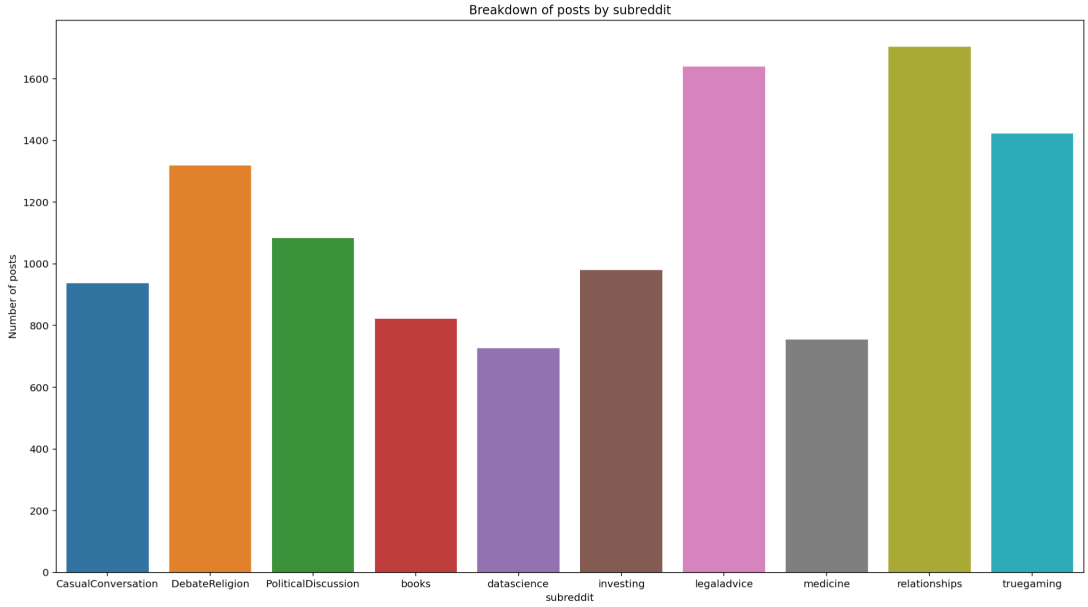
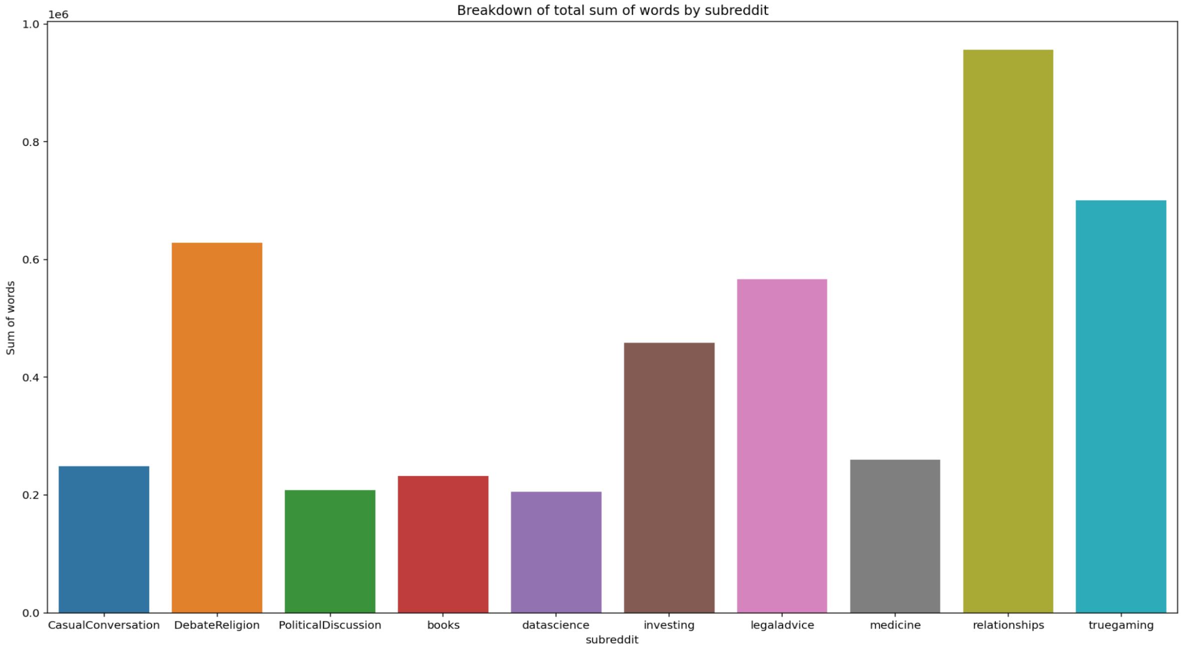


Image 1

The image above shows the 10 subreddits selected along with the number of posts obtained for each of them. Most of these communities have a distinct vocabulary specific to the topic they are about but others like “relationships”
and “CasualConversation” tends to be more diverse. We additionally filtered from the dataset those posts whose text had less than 25 words in them.



We’ve additionally made a **Tableau Workbook** showcasing some of our initial exploratory data analysis of this dataset that we think may bring further insights about this data.

2. Preprocessing

In the preprocessing, we deal with the conversion from the initial original Reddit posts to a format in which the data can be vectorized efficiently to later obtain optimal results in the classification task.

The Reddit posts have a messy original format. Text and links to other pages (in URL format) are mixed together and, since the content of Reddit tends to be informal, there are misspellings and all types of expressions. So, preprocessing is a specially important step for the data that we use in this project.

The objective is to remove all the irrelevant information from the original posts, trying to keep only the information that provides semantic content. To achieve this, we have designed a pipeline that contains the following steps:

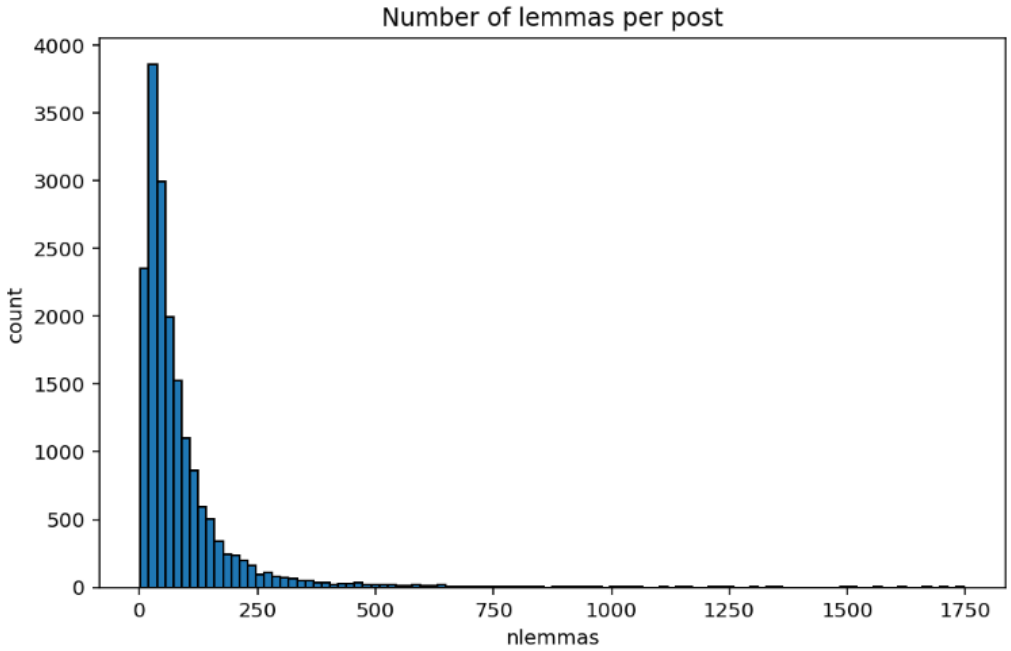
1. Tokenization: Segments the text into words (the tokens).
2. Non-alpha-numeric removal: removes all tokens that are not classified as alpha-numeric by Spacy.
3. Filtering by tag: keeps only nouns, verbs, adjectives, and prepositions.
4. Generic stopword removal: remove all words that are classified as stopwords.
5. Lemmatization: conversion of the tokens into their base form.

We have decided not to filter by English sentences because we have not considered it necessary, since in Reddit posts there are only terms or expressions in other languages, but it is uncommon to find entire sentences. And this step would add much computational time-consuming in the execution of the pipeline. As such, we've only included a step to remove words that are not included in *NLTK's corpora of English words*[2].

We add the "specific stopword removal" step to the pipeline after observing the results of the topic descriptions obtained by LDA. This step consists of removing the common words that don't provide semantic content information.

Because the accuracy of topic modeling in very short documents tends to be poor the next step in our preprocessing was to filter the posts by post size. We did this by analyzing the distribution of the number of lemmas and then keeping only the documents that have at least 20 lemmas.

[2] The NLTK data package includes a number of lexicons and word lists that are accessed just like text corpora. https://www.nltk.org/howto/corpus.html#word-lists-and-lexicons



In addition, before creating our topic model, we have applied another filter. It consists of removing the terms that appear less than five times in the dataset and the ones that appear in more than 40% of the posts.

We remove the low frequent tokens because they are useless if we want to find trends or repetitive patterns across the posts. This filter also will remove for us many misspellings, which are not strange to find in the Reddit posts. We also decided to delete the high-frequency words because they are probably generic words that don’t add semantic meaning.

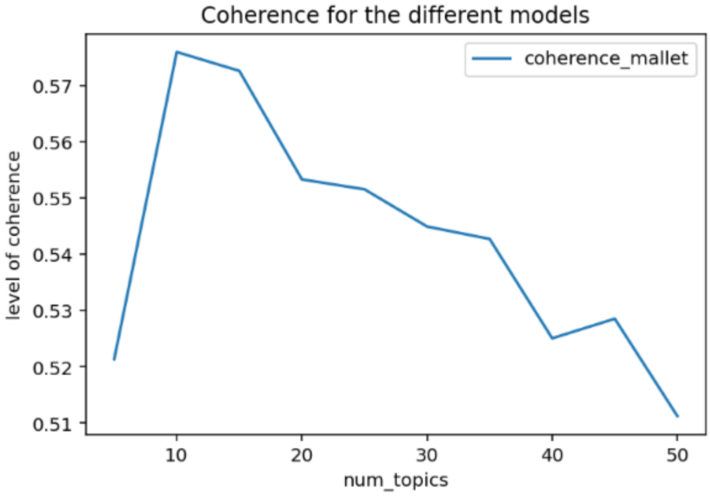
Using Gensim, we have also computed N-gram detection, which binds two or more tokens into a single one, based on the frequency that these tokens appear together in the posts. We have tended to be restricted in terms of the threshold to detect them because we prefer to not detect false n-grams than the contrary.

Finally, we have created a Dictionary object using Gensim and we have converted the corpus data into a bag of word representation, which is the format that is needed to later create the topic LDA model. The Dictionary stores all the words found with an assigned id and some other attributes and the bag of words representation contains the tokens of each post with their corresponding times that they have appeared in the post.

3. Topic Modeling and Vector Representation

Once the data had been properly preprocessed, it was time to apply a topic modeling algorithm, particularly the Latent Dirichlet Allocation algorithm. We have used Gensim's LDA Mallet wrapper to implement it.

To obtain the model, we first have to select the number of topics, which will affect if the topics found are meaningful or "garbage" (redundant). To choose it we have used the coherence metric that Gensim provides. So, we have created models with the Dictionary and the Bag of Words obtained and different numbers of topics, to later compute and store the coherence of each of them. The results of this analysis were the following:



We can observe that the highest level of coherence is achieved when the number of topics is 10. So, we have set this value as the final number of topics we will use. After obtaining the model we have plotted the result of the vector representations of the topics with their 25 most probable words.

Observing these results, we have detected a list of words that seem to be common in the topics but do not add semantic meaning, for example, the word "lot" is redundant and it is one of the most probable tokens in some of the topics. So, we have stored these words into a list and we have filtered them out from our Bag of Words. This step is called specific-word removal, and it is important to avoid the presence of garbage topics.

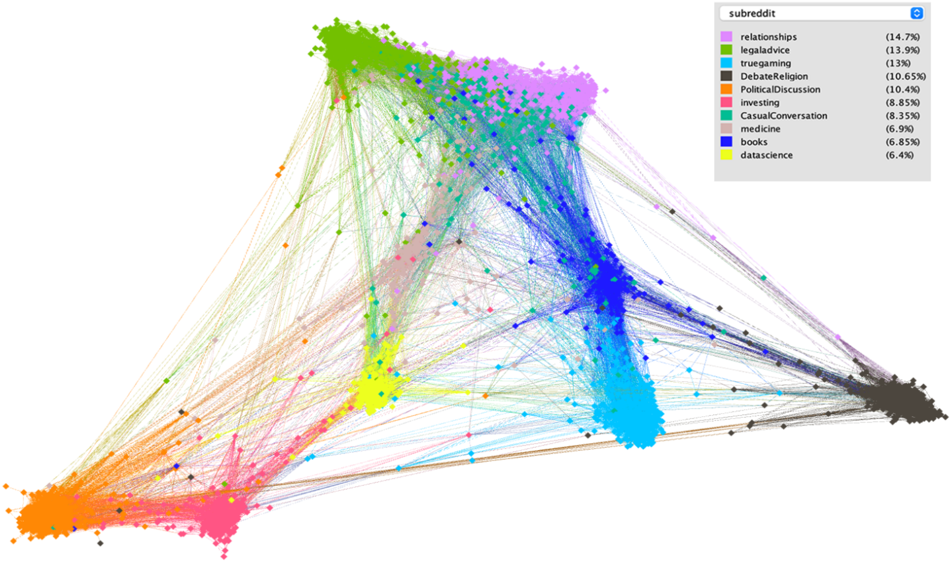
After cleaning the data, we have to repeat all the preprocessing, including the computation of the number of topics. We have stopped and got the final results when the number of garbage topics has been reduced.

4. Semantic Graph Representation with Gephi

To create the graph shown below, we created two CSV files. The first one contains information about the nodes of the graph such as the id, the title of the documents, and the subreddit to which it belongs. The number of nodes included, 2000, were chosen so that the graph was representative enough and the computational time of Gephi was feasible. The second CSV file contains information about the edges such as the weight of the links. We decided to filter links that may have a lower relevance, so we applied a threshold of 0.85 to the weight attribute. The higher the weight of the edge that connects two nodes, the more similar the two nodes are.

Once we have the two CSV files, we use Gephi to obtain the graph visualization. As you can see in the legend, we assigned one color per subreddit. The size of the nodes of the graph was increased and we decided to apply an edge filter of 0.9, so in the graph, there are only edges with a weight equal to or larger of 0.9, both with the idea of improving the aesthetics of the graph. The layout chosen is Force Atlas 2.

By looking at the graph, we can see that it intuitively makes sense because topics like Data Science (yellow) are very far away from others like DebateReligion (brown) but are closers to medicine (clear pink), books (dark blue), and investing (red).



5. Classification

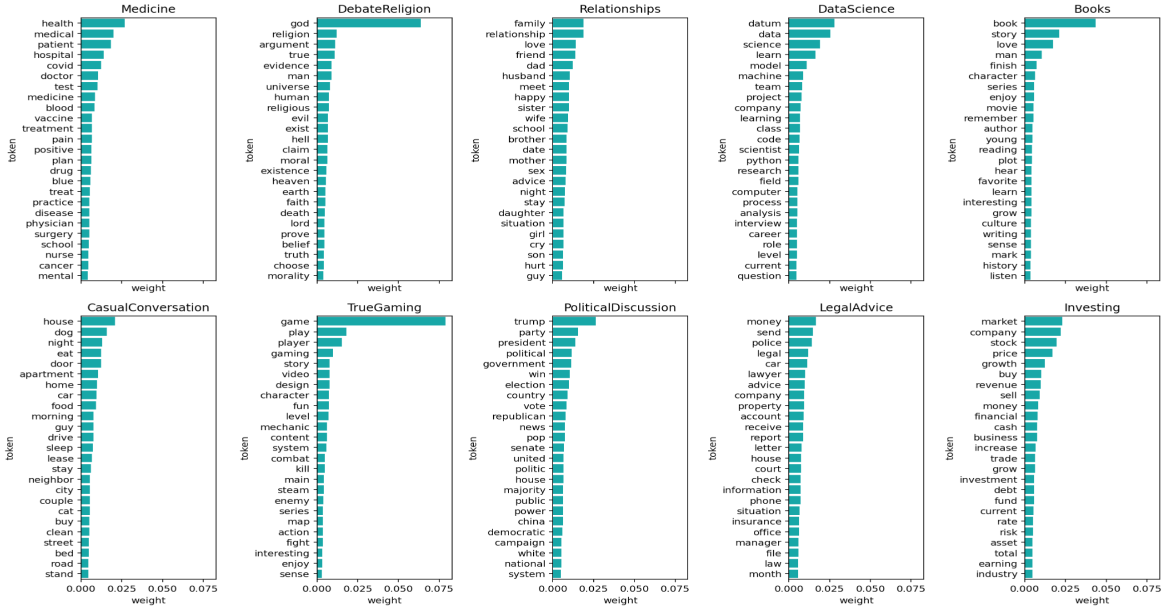
In this section, we compare the performance of using LDA and TF-IDF vector representation techniques to predict which subreddit a post came from given the semantic representation of its content and give a comprehensive description of the methodology used in our implementation and evaluation of the Machine Learning and feature engineering techniques to improve, obtain, and assess the results of each model.

5.1 LDA-based Unsupervised Classification

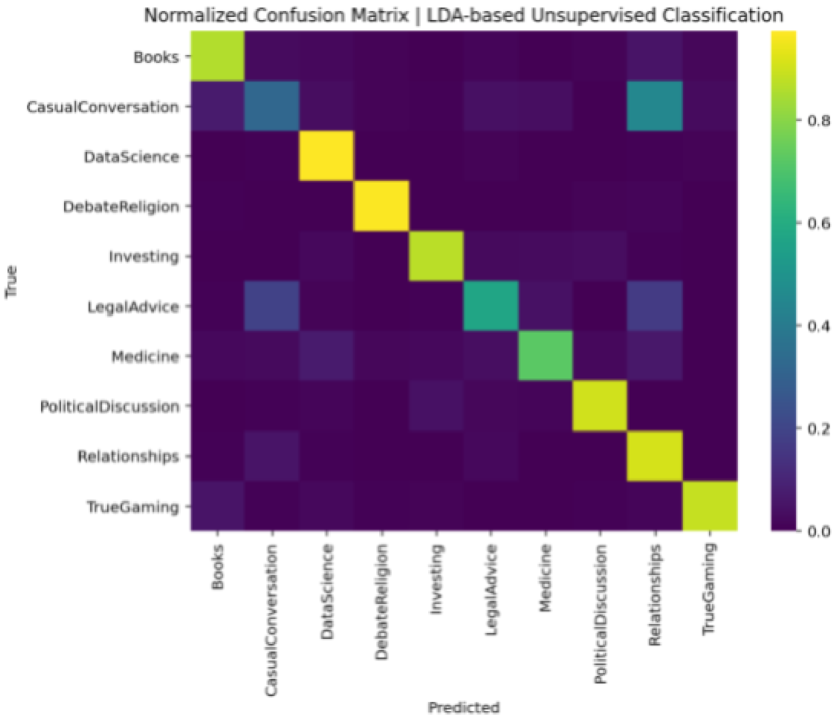
Since Reddit forums are often about a certain subject there's a clear relationship between them and traditional topics obtained from doing semantic modeling. For this reason, we've considered a Latent Dirichlet Allocation model with ten topics to model in the assumption that the topics obtained are related to the subjects of the ten respective subreddits of the dataset.

We verified our assumptions by training and analyzing a new LDA model using only a fixed number of 10 topics, the same number of subreddits to classify, such that it is possible to label a post based on the most relevant topic as assigned by the LDA model. This could be seen as an unsupervised approach to solving the classification task as the LDA model only receives as input the bag of words obtained from the text in each post of the dataset.

As such, to obtain the classifications with this approach we manually labeled each of the 10 topics obtained from the aforementioned model with the name of the subreddit that we think is more related to it based on the relevant words for that topic. These results more obvious when we visualize the most relevant words in each topic as can be seen in the figure below, it can be concluded by close inspection of the words that there is indeed a clear relationship between the topics obtained and the subject of the subreddits that we are attempting to classify in this task.

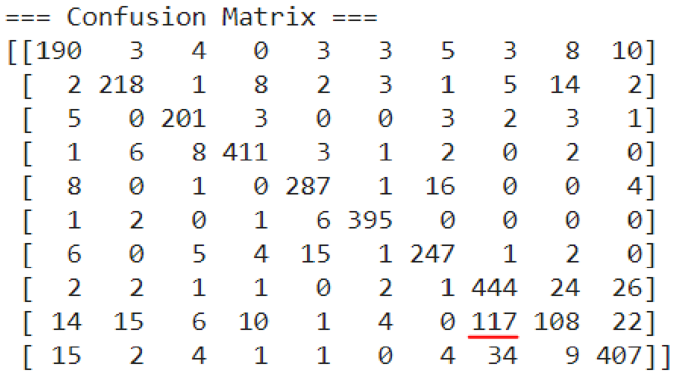


We evaluated this approach with a test set consisting of 30% of the documents (posts) in the dataset and obtained around 80% of accuracy. However, breaking down the results for each label through a confusion matrix revealed a significant percentage of prediction error for subreddits like “r/CasualConversation”, “r/LegalAdvice” and “r/Relationships”. The reason for this is likely because these communities tend to be more general-oriented and have a less distinct vocabulary than others such as “DataScience” and “DebateReligion”.



5.2 LDA Supervised Classification

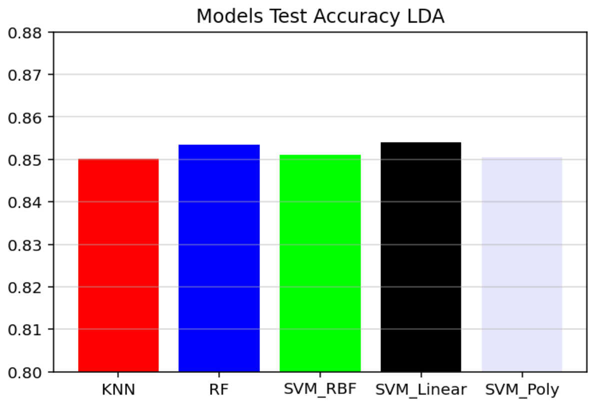
For this supervised task, we are going to use the “subreddit” label we have on the dataset to perform the classification. The LDA representation gives us a total of 10 variables with ranges between 0 and 1. This makes no need to reduce the dimension, scale or search mutual information as basically, LDA is telling the weights of being part of each topic. This first hypothesis is confirmed once we enter the cross-validation part. Here we can see that all the models are giving the same neighborhood of accuracy being the most difficult topics to classify “casual conversation” being confused with “relationships”.

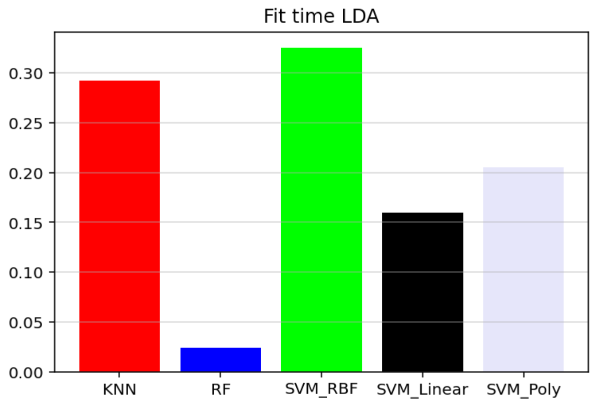


```
=== Confusion Matrix ===
[[190   3   4   0   3   3   5   3   8  10]
 [  2 218   1   8   2   3   1   5  14   2]
 [  5   0 201   3   0   0   3   2   3   1]
 [  1   6   8 411   3   1   2   0   2   0]
 [  8   0   1   0 287   1  16   0   0   4]
 [  1   2   0   1   6 395   0   0   0   0]
 [  6   0   5   4  15   1 247   1   2   0]
 [  2   2   1   1   0   2   1 444  24  26]
 [ 14  15   6  10   1   4   0 117 108  22]
 [ 15   2   4   1   1   0   4  34   9 407]]
```

On this confusion matrix, obtained from the RBF SVM classifier, we can see underline in red the problematic topics we have mentioned before.

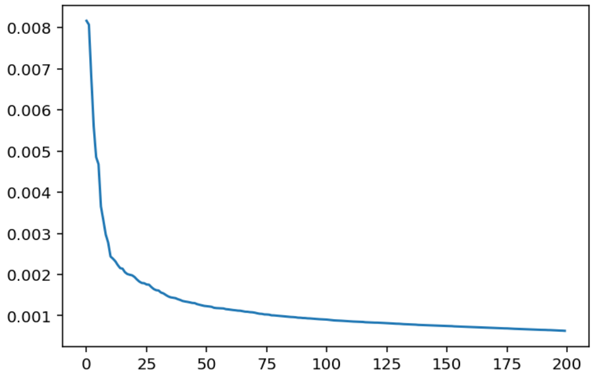
The fitting time is also almost instant for every model (the graph measurements below are taken in seconds). So the conclusion will be that with LDA the accuracy does not fully depend on the classifier used but mostly on how well this LDA vector is constructed via all the NLP pipelines. The unsupervised accuracy we just get before the only thing it does is to assign a label depending on the higher column on the LDA and it gets around 80% so the models are improving the accuracy but not much.





5.3 TF-IDF Supervised Classification

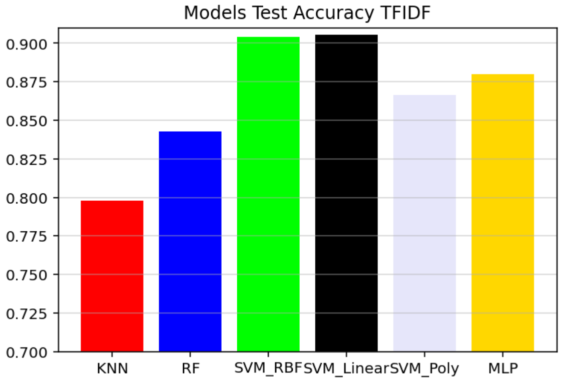
Now instead of having 10 variables given by the LDA representation, we have more than 9200 from TFIDF representing the frequencies for which each token appears in a document. This part has a more interesting preprocessing as now with such a high number of dimensions, the models take too long to cross-validate. To fix that, first, we have applied PCA to significantly reduce the dimensions.

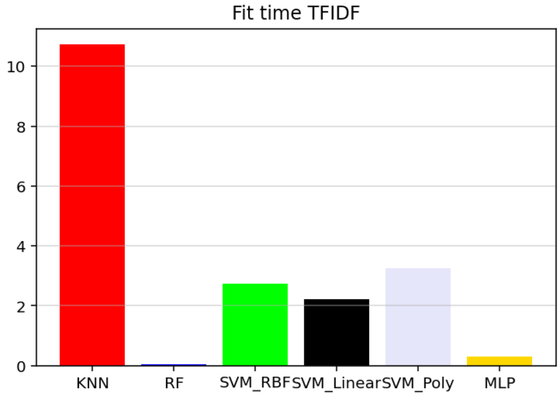


On the graph above, we can see that the greatest variance changes on components stop around component 100. By making some cross-validation we got that the perfect balance between accuracy and computational time was to take 200 components. The total variance of the model was around 24% which sounds pretty low but, compared with the results taking 3000 components (explaining around 88% of variance), are completely equal but with much longer computational time.

We have also tried some mutual information techniques that don't help us much more than enlarging the computational times. Also, we fail in attempting some supervised reductional techniques such as Linear Discriminant Analysis as they crashed or take way too long to compute.

Once we got the PCA projection to our train and the same projection to our test, we started the model cross-validation part. For this, we have additionally implemented a Multi-Layer Perceptron. With this MLP model, we can use the whole 9200+ variables in a pretty fast way thanks to the utilization of GPUs to compute operations in parallel. The negative part of this last approach is that as we don't have many training observations, the MLP overfits a lot even when we have applied all the regularization techniques we have learned in that course. Despite that, the fit time and the accuracy are quite good compared with other models. This time we can see some differences between the principal models we have used and we have two candidates for the best TF-IDF model.





As we can see, the Linear SVM offers not only a reasonable fitting time for the whole test set but also an accuracy of above 90%. But, if we want a model that performs reasonably well at around 88% of accuracy but with a much higher speed, the MLP model will be our choice. Furthermore, with more training instances and maybe with a better tuning of the hidden layers, we will have the perfect model to face this NLP classification task.

Code User's Manual

Our submission of this project includes three ipython notebooks that can be used to reproduce the exploratory and scientific analysis that we've summarized in this document. These notebooks are the following:

1. ***reddit_data_acquisition.ipynb***: Notebook used to acquire and export the dataset used in this project. To run this notebook to scrape posts from Reddit and build the dataset, some environmental variables defining the keys and user ID of the Reddit API need to be set. We've included a cell to automatically download the .env file and load these environmental variables in the notebook included in the submission *but not in the notebook available through the public git repository.*

2. ***task_1_reddit_topic_modeling_and_graph.ipynb***: Notebook used to process the textual information obtained from the dataset, do topic modeling, and finally obtain the weights and node files to build the semantic graph.

3. ***task_2_subreddit_classification.ipynb***: Notebook used to train, evaluate, and compare models used to predict the subreddit of a post given only its semantic representation. Performance with both LDA and TFIDF representations is compared among each other.

The three of them should be capable of being executed entirely in a Google Colab kernel. We warn, however, that because the last notebook implements a lot of memory and performance-intensive tasks such as training and cross-validating many different models it might take several hours to finish its execution.

We've additionally made our code available through a GitHub repository that we think can be better used for reproducibility.

Some additional material that is not in the Github repository is the Exploratory Analysis Story of the dataset that we've made with Tableau and is available on this Tableau Public page.