Individual Coursework Submission Form

Specialist Masters Programme

Surname: Sinkov	**First Name: Evgenii**
MSc in: Business Analytics	**Student ID number: 210046075/1**
Module Code: SMM799	
Module Title: Applied Research Project	
ARP tutor name: Dr. Rui Zhu and Dr. Ahmad Abu-Khazneh	**Submission Date: 25.08.2022**

Declaration:

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the coursework instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

We acknowledge that work submitted late without a granted extension will be subject to penalties, as outlined in the Programme Handbook. Penalties will be applied for a maximum of five days lateness, after which a mark of zero will be awarded.

Marker's Comments (if not being marked on-line):

Deduction for Late Submission:

Final Mark: **%**

Developing a Natural Language Processing Classifier as a Feature for an Assistive Device

Sinkov Evgenii

August 25, 2022

This report is submitted as part of the requirements for the award of the MSc Business Analytics

Bayes Business School

City University, London

Abstract

Since the emergence of AI and the recent increase in computational power of machines, the healthcare industry has quickly become one of the most forward-thinking fields of application. This work is intended to help the Rolls-Royce initiative in developing an assistive technology device for Motor Neuron Disease patients. This paper compares different Natural Language Processing and machine learning techniques for predicting the category of a user's question using a publicly available dataset - Yahoo Answers, which contains ten categories and 1,4 million rows. The methodology included careful consideration of various pre-processing pipelines, the definition of customized functions, and the comparison and selection of machine learning classifiers. The resulting performance was evaluated based on the accuracy capable of capturing category similarity and corresponding training time. The following settings were justified and chosen based on the constraints set by the Rolls Royce research team: Pre-processing layer containing regex cleaning; TF IDF approach as a vectorization method; and Naïve Bayes as a classifier. The resulting customized accuracy, capturing similarity between categories, trained on one hundred thousand questions is 68%. The various misclassification examples were explored, and potential extensions to the classification procedure were discussed. The team can either use the model with the suggested settings or run further experiments using the provided comparison pipeline.

Contents

List of Figures

1. Introduction

1.1 MND facts

Motor neuron disease (MND) causes the motor neurons to gradually stop sending messages to the muscles. This causes the muscles to deteriorate, impairing one's ability to walk, talk, eat, drink, and breathe. At any given time, up to 5,000 adults in the UK are affected by MND. One in every 300 people is at risk of developing MND during their lifetime. Adults of any age can be affected, but people over the age of 50 are more susceptible to the disease (MND Association, 2022).
The MND association recognizes five distinct types of MND with varying degrees of limb mobility, life expectancy, and speech ability. Providing a high-quality conversational device would greatly improve the quality of life of individuals affected by MND who can speak.

1.2 Assistive Device Description and Requirements

The NLP workflow for the potential device, provided by an industrial partner, is displayed in Appendix A. Instead of relying solely on single pre-trained Natural Language Processing (NLP) model, the presented approach to building conversational pipeline uses a variety of chatbots that have been trained on different datasets based on corresponding contexts, topics, emotionality. The method for selecting which chatbot to use for each user question has not yet been implemented, though. This paper's goal is to present a classification prototype that will allow intelligent chatbot selection based on the context of the user input.

When specifying requirements for the classification model, the industry partner mentioned that the device would not have a lot of computation power due to privacy issues of medical data, making the time-efficiency of the classifier very crucial. Moreover, the classifier will have to be executed after each user-query, putting even more strain on the training process's speed.

Another requirement raised by the client was making the accuracy of the classifier capable of taking into consideration the similarity of categories. In other words, misclassifications should not be treated equally for all categories. Instead, if the predicted versus true categories are similar (e.g., Health and Sports) the reward score should be above 0. If dissimilar (e.g., Business and Finance comparing to Family and Relationships) – below 0.

2. Literature review

2.1 Natural Language Processing (NLP) and its Applicability in Healthcare

“Collecting, understanding, processing, and generating insights out of text data is called natural language processing” (Kulkarni et al., 2022). It is considered to be one of the most rapidly growing technologies in AI. Well-established Python libraries and communities provide developers with an opportunity to solve a great variety of tasks ranging from text classification, topic modeling, sentiment analysis, text summarization, translation, and generation. NLP was developed at the interference between linguistics, mathematics, and machine learning (ML) concepts.

An assistive device for MND patients can be considered a smart chatbot. Soufyane Ayanouz et al (Ayanouz, Abdelhakim and Benhmed, 2020) give the following definition of a chatbot: “The chatbot is also known as chatter robots, are software agents that simulate human conversation via text or voice messages”. The article provides examples of chatbots that have already been put into use for specific cancer and insomnia patients as well as a more general chatbot created by Babylon Health, a partner of NHS, that makes recommendations based on textual input – symptom descriptions.

2.2 Pre-processing Tools and Techniques for NLP tasks

According to Akshay Kulkarni et al (Kulkarni, 2022) the raw data is crucial to preprocess before executing any NLP pipeline, because undesirable, junk text might influence the direction of the results produced by a machine learning algorithm. Thus, most NLP pipelines include a pre-processing layer that consists of a unique combination of text cleaning and normalization functions. The first step is typically a procedure for removing numbers, characters, uppercase words, and other manipulations based on the aim of an NLP task. Irfan Alghani Khalid (Khalid, 2020) has presented a comprehensive text cleaning workflow that uses the Python regular expressions (regex) library and is therefore very time efficient.

2.3 Vectorization

The next step toward developing a classifier is to transform each cleaned question into a vector of numbers that the machine can understand. “Any type of text analysis requires the transformation of unstructured documents into structured data. In machine learning, for supervised learning, it is necessary to create feature vectors in a specified feature space.” (Krzeszewska, Poniszewska-Marańda and Ochelska-Mierzejewska, 2022). There is a plethora of techniques available to complete this task. Some rely only on words within the dataset when creating vectors (Bag of Words, TF IDF) and are very fast; others – implement pre-trained linguistic models, containing vector representations of each word; and still others – require machine learning training (Doc2vec, Skip-gram bag of words) and are more computationally complex.

Mauro Di Pietro (Di Pietro, 2022) has provided a comprehensive comparison of 3 NLP classifiers, based on Term Frequency-Inverse Document Frequency (TF IDF), Word2Vec[1] and BERT[2] vectorization techniques applied to a News Category Dataset containing 200,000 news headlines and 41 corresponding categories. The accuracy score obtained by using TF IDF reached 0.85, outperforming Word2Vec (0.84) but slightly underperforming BERT (0.87). Given the much-increased number of examples and 4 times smaller number of categories the priority of TF IDF over other vectorization methods only grows when applying it to the Yahoo dataset. An article prepared by Gaurav (Gaurav, 2021) strengthens this intuition. The author provided a similar analysis of two NLP approaches: a classical TF IDF vectorization coupled with machine learning and BERT. Indeed, in that case, TF IDF vectorization provided a higher accuracy – 0.86 vs. 0.856.

2.4 Exploration of different Machine Learning Classifiers

There is a great variety of options for selecting the machine learning algorithm to enable supervised learning on a training dataset and acquire predictions on a test dataset using the already computed vectors. Turner (Turner et al., 2017) provides a comparison of four different ML classifiers (Neural Networks, Random Forests, Naive Bayes, and Support Vector Machines) when predicting the Systemic Lupus Erythematosus (SLE) diagnosis based on medical notes for 662 patients. Glaz (Le Glaz et al., 2021) adds decision trees to this list since they allow interpretability of the resulting predictions.

This work narrows the pool of potential classifiers and concentrates on the two "industry standards"—Naive Bayes and Logistic Regression. The simplicity of the Naive Bayes classifier is emphasized by V. Ratz (2021) as opposed to "heavy AI-based semantic analysis." Logistic regression, on the other hand, requires

[1] Word embedding technique. Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. (Karani, 2018)
[2] Bidirectional Encoder Representations from Transformers.

significantly more iterations due to its optimization nature. It is more complex in terms of the specification of several tuning parameters, which can dramatically influence the predictive power of the classifier and avoid overfitting[1]. Soner Yildirim (Yildirim, 2020) explains the nature of overfitting and explains how to address this issue with regularization techniques in Python. Scikit-learn library presents a variety of different tuning parameters to construct a reliable logistic regression, such as:

- Solver – the optimization algorithm.
- Max_iteration – the maximum number of iterations.
- Type of regularization (Ridge or Lasso) and the corresponding penalty score.

To reduce the training time of the logistic regression several alternatives were considered. First, the dimensionality of TF IDF vectors could be reduced by applying machine learning techniques. According to Kadhim et al. (Ismael Kadhim, Cheah, Abbas Hieder and Ahmed Ali, 2017), the primary objective of dimensionality reduction (DR) is to compress high-dimensional input to a lower-dimensional subspace while retaining as much variation as possible. There are multiple ways available to perform this task using concepts from linear algebra. Pranav Thaenraj (Thaenraj, 2021) demonstrates the practical implementation of three various methods – Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Second, the size of the resulting TF IDF matrix could also be regulated by limiting the vocabulary size when performing TF IDF vectorization. However, experimentation is necessary to understand which procedure to implement in the classification pipeline for the assistive device.

2.5 Assessing Semantic Similarity Between Words

By measuring the cosine of the angle between word vectors we could obtain semantic similarity. There are other similarity metrics, for example, Euclidean distance or Jaccard similarity. Singh et al. (Singh and Singh, 2020) made a comprehensive review of the 3 mentioned methods while estimating text similarities in news articles and highlighted the performance of cosine similarity, which gained the highest accuracy in that paper.

Cosine similarity is present in many pre-trained linguistic models available on the internet. However, these models differ depending on the context of textual data, vocabulary size, and dimensionality. The “glove-twitter-200” model (fse/glove-twitter-200, 2022) is trained on 2 billion tweets, that resemble Yahoo conversations and thus may be applicable in the context of the project. The model represents each word as a vector with 200 numerical features.

[1] Overfitting is a concept in data science, which occurs when a statistical model fits exactly against its training data. When this happens, the algorithm unfortunately cannot perform accurately against unseen data, defeating its purpose. (IBM Cloud Education (2021).

3. Methodology

3.1 Dataset Exploration

The Yahoo answers dataset contains 1,400,000 question – answers in the train set and 60,000 in the test set. Each observation is labelled with one of 10 categories (Society and Culture, Science and Mathematics, Health, Education and Reference, Computers and Internet, Sports, Business and Finance, Entertainment and Music, Family and Relationships, Politics and Government) and contains 3 other columns: The question title, question content and best answer. This dataset was not chosen at random; it contains a substantial amount of textual data from user-generated questions and answers on the 10 most popular topics, making it very useful for chatbot allocation. It depicts the concerns, interests, and questions people express online. The number of questions per category is evenly distributed, making the training process more reliable without additional need for data imputation or cut. The NLP classification model will be trained using the title and content of the questions without considering the answers. Intuitively, the model needs to be trained to identify the category of the question without using the answer as a source of additional vocabulary. On average, each question body and title combined contain 32 words. The maximum length of a question is 919 words, so the dataset represents both concise and detailed questions. The distribution of lengths of questions is presented in Figure 1. Assuming the patient can make a request with as many words as possible, the length of the given example is not limited.

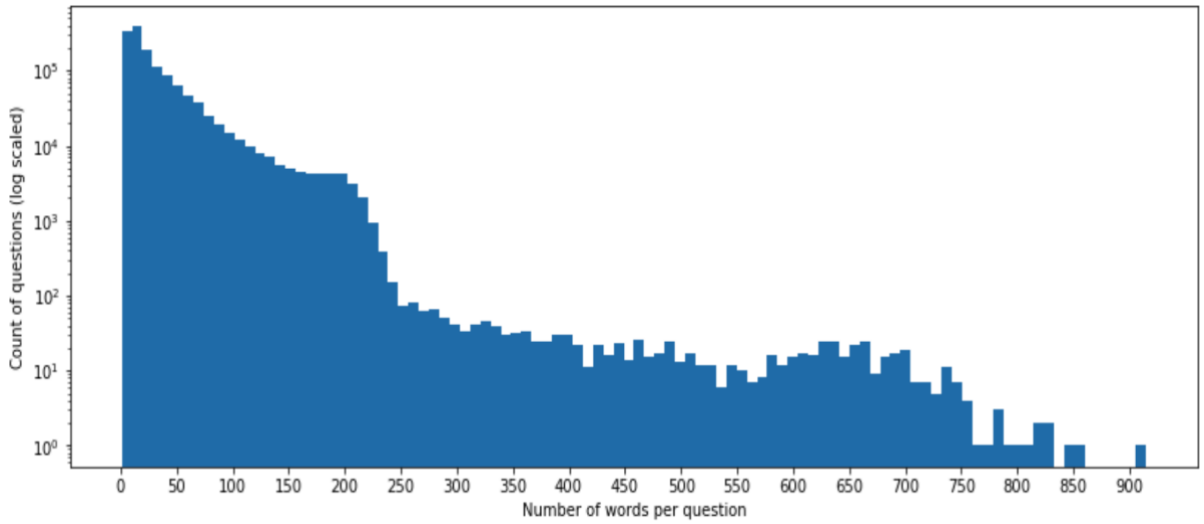


Figure 1: Distribution of number of words per question

3.2 Developing the Pre-processing Pipeline

The Yahoo textual data contains various site links, numbers, abbreviations, slang jargon, and misspellings. Furthermore, at least 1% of all questions are written in a language other than English. Attempting to pre-process the text perfectly, correcting all words, extracting dates and numbers, paraphrasing slang, translating non-English words, and lemmatizing[1] it can be exceedingly computationally difficult. Therefore, the lemmatization, language check, and spelling correction layers were each added separately, allowing for flexible activation based on the user's preferences. The complete pre-processing layer is displayed on Figure 2.

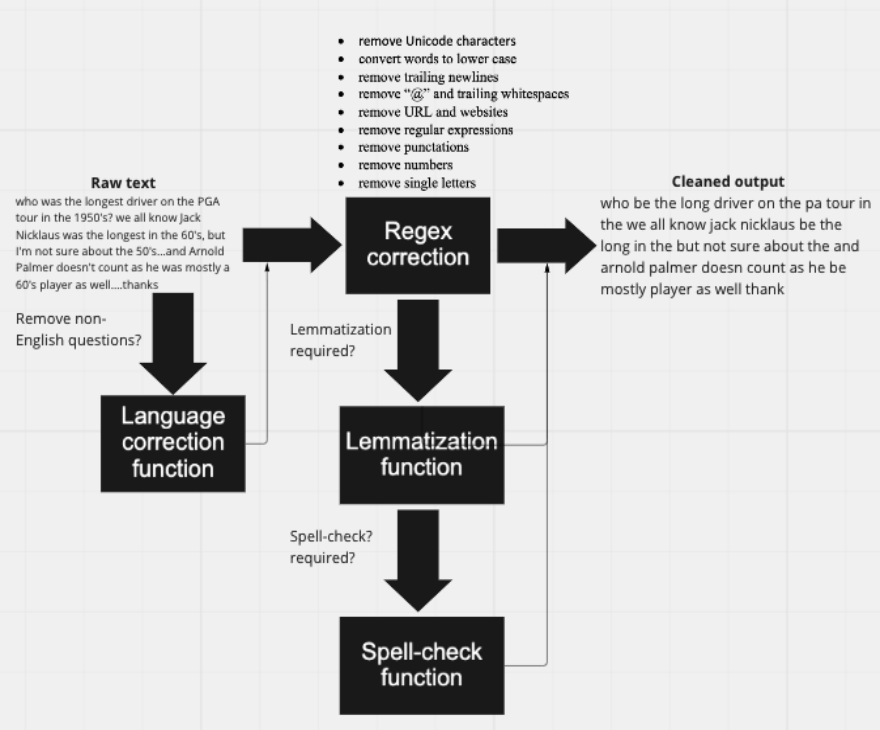


Figure 2: Pre-processing Pipeline

3.3 Vectorization Part

Based on the articles previously mentioned regarding vectorization techniques, TF IDF approach will be applied because it captures both how frequently a specific term appears in the given question and evaluates the semantic value of this word globally, across the entire dataset.

The built-in TfidfVectorizer function from the machine learning Python library – Scikit-learn (Machine Learning in Python, 2022) will be used to perform the vectorization of pre-processed questions. The function takes textual data and converts it into vectors – TF IDF features automatically. Among other inputs,

[1] Lemmatization gets to the root word, considering the tenses and plurality of a word. For example, best and good can mean the same thing. After lemmatization, the result for both words is good.

the function can specify a maximum number of features, which is the same size as a vocabulary. The potential bottleneck could arise when passing matrices with too large dimensions into the machine learning steps. For example, the corresponding TF IDF matrix for the whole train dataset has more than 400,000 features which could result in hours of training for logistic regression.

3.4 Creating a Custom Function to Obtain Prediction Accuracy

This challenge could be accomplished either by manually assessing semantic similarity between categories using one's subjective knowledge or by using the pre-trained linguistic model.

First, a function retrieving the average similarity score using the "glove-twitter-200" model between any set of words was created. This function was then applied to the Yahoo categories to obtain a similarity matrix containing corresponding similarity scores. Next, the resulting scores for any misclassification were scaled between -0.5 and 0.5. The resulting penalties and rewards matrix is visualized in Figure 3 using a heatmap.

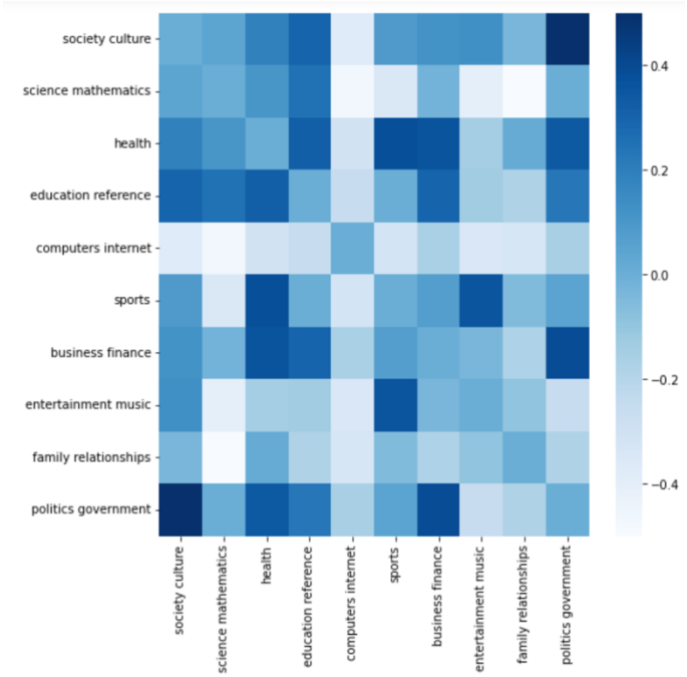


Figure 3: Penalty and Reward Scores for Categories in the Yahoo Dataset

The most semantically similar pair, having the highest reward score is Society and Culture against Politics and Government. The least similar – Family and Relationships against Science and Mathematics.

Finally, the function "mod_accuracy" outputs the customized accuracy score by calculating the mean of both correctly predicted (score of 1) and misclassified categories (-0.5 to 0.5 score is obtained from the similarity matrix).

3.5 Classification Layer Tuning

As previously mentioned in the literature review section, two ML classifiers, both widely utilized in solving NLP classification tasks – Multinomial Logistic Regression and Naive Bayes will be applied. However, logistic regression requires a much longer training time compared to Naive Bayes. For instance, training a sample of 100,000 questions with 83,180 TD IDF features would take more than 200 seconds, whereas Naïve Bayes would require less than a second. The following experiments will be carried out to select suitable classifier parameters:

1. Choosing the regularization strategy to avoid overfitting for the logistic regression by measuring the modified accuracy for different solvers ('saga', 'lbfgs'), penalty scores (0.1,0.5,1,5,10) and types of regularization (Ridge, Lasso) on the training dataset size of 100,000 and test size of 5,000.
2. Selecting appropriate dimensionality reduction strategy by measuring transformation and training time and calculating the corresponding modified accuracy for the following parameters:
    - Training size – 1,000; 10,000; 100,000
    - Test size – 5,000.
    - Maximum vocabulary for TF IDF – 100; 500; 1,000; 5,000; Unlimited
    - Number of features (components) for SVD – 100; 500; 1,000.
    - Pre-processing using only a regex layer.
3. Evaluating the performance of both ML classifiers by developing a comparison pipeline, capturing the trade-offs between the training and testing sizes, accuracy, and time-efficiency. The pipeline will enable the end user of the classification pipeline to make informed decisions about which classifier to utilize. The proposed workflow contains "fit_performance" function that takes arguments from the pre-processing pipeline and outputs train, test, modified accuracy and executing time. More precisely its functionality is displayed in Figure 4.

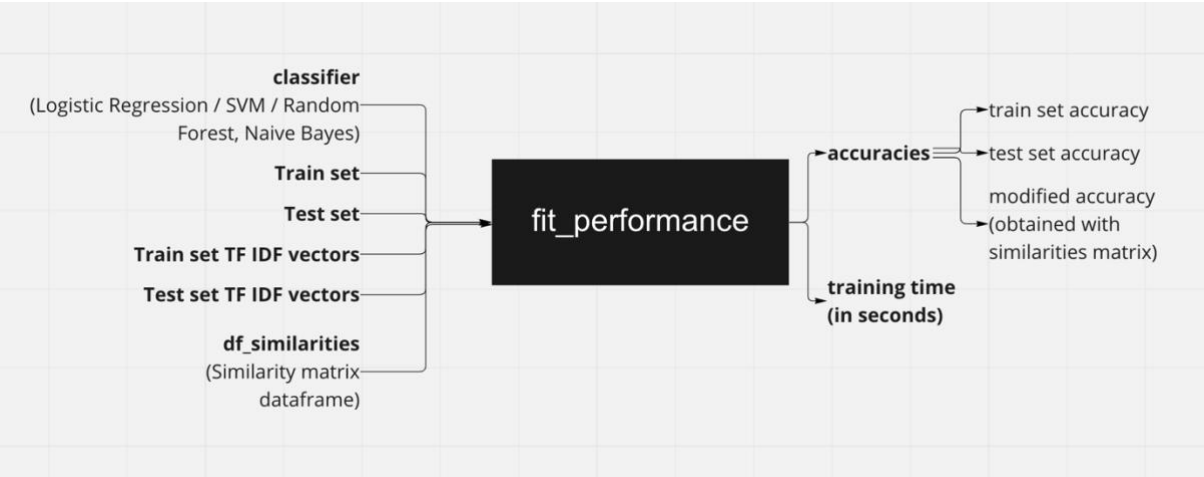
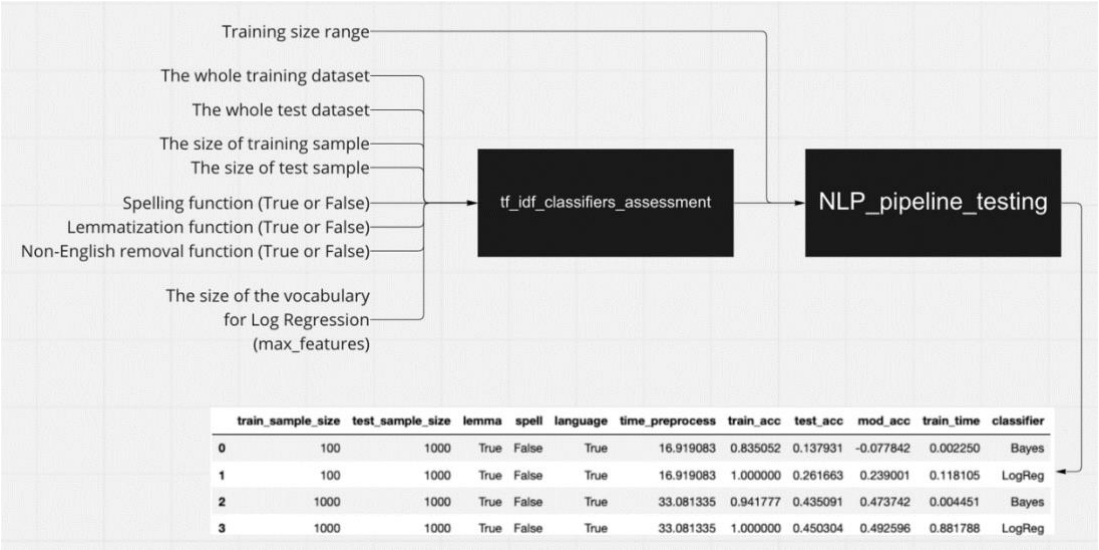


Figure 4: fit_performance Function Diagram

3.6 Completed Classifiers Comparison Pipeline

To assess the performance of classification pipelines the two functions were designed (Figure 5) and the following experimental settings will be applied:

- Training set range – 100; 1,000; 10,000; 100,000; 1,000,000.
- Lemmatization and Non-English removal functions are activated.
- Vocabulary size of 5,000 words for the logistic regression.
- Full test dataset – 60,000 examples.



	train_sample_size	test_sample_size	lemma	spell	language	time_preprocess	train_acc	test_acc	mod_acc	train_time	classifier
0	100	1000	True	False	True	16.919083	0.835052	0.137931	-0.077842	0.002250	Bayes
1	100	1000	True	False	True	16.919083	1.000000	0.261663	0.239001	0.118105	LogReg
2	1000	1000	True	False	True	33.081335	0.941777	0.435091	0.473742	0.004451	Bayes
3	1000	1000	True	False	True	33.081335	1.000000	0.450304	0.492596	0.881788	LogReg

Figure 5: tf_idf_classifiers_assessment and NLP_pipeline_testing Functions Diagram

Finally, a possible setting for a chatbot allocation problem will be applied as the project's last step based on the findings of the experiments. Additionally, the corresponding interpretation of results will be provided.

4. Experiments

4.1 Selecting Regularization Strategy

Ridge regression regularization ('l2') with a penalty term of 5 is the combination that produces the highest testing accuracy. Thus, it will be implemented in the next experiments. Figure 6 shows how overfitting is changing for different parameters specified in the methodology section.

	Type_of_Regularization	Penalty_term	Training_accuracy	Testing_accuracy
0	l2	0.1	0.6058	0.517607
1	l2	0.5	0.7483	0.566026
2	l2	1.0	0.8238	0.583633
3	l2	5.0	0.9629	0.584634
4	l2	10.0	0.9871	0.576831

Figure 6: Performance of the Logistic Regression for Different Regularization Parameters

4.2 Selecting Appropriate Dimensionality Reduction Strategy

Figure 7 displays the performance of dimensionality reduction of both approaches (Limiting TF IDF vocabulary and reducing dimensionality via SVD)

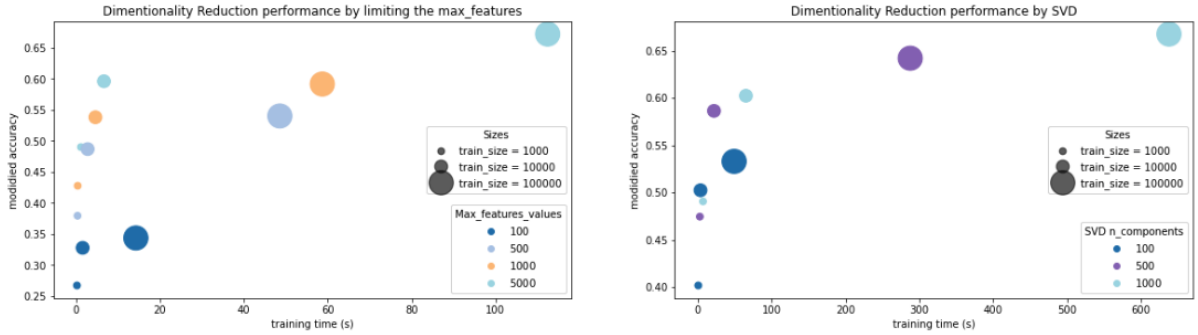


Figure 7: The Comparison of Limiting Vocabulary Size (max_features) and Performing SVD as a Means to Reduce Dimensionality of TF IDF Vectors

The visualization reveals that the SVD approach requires significantly more time to perform the reduction and does not increase the modified accuracy compared to the second method. Thus, reducing 83,180 features from a training set of 100,000 questions to 1,000 features took 650 seconds. The final accuracy was 66.75%. In contrast, it took 111 seconds to train the classifier with a vocabulary limit of 5,000 words and the same number of training questions, achieving a higher accuracy (67.21%). Hence, the preference will be given to the vocabulary reduction method. The end-user of the classifier, given the scarcity of the computation time, can also decide on the train sample size and the corresponding extent of accuracy loss.

For example, the classifier with the vocabulary size of 5,000 trained on 10,000 questions demonstrates 59.61% accuracy and requires only 6.6 seconds.

Figure 8 shows the performance of the classifier without any dimensionality reduction. The highest modified accuracy score achieved in this experiment is 70%. However, it required a significant amount of the training time (262 seconds). Thus, in the following stages of the classification workflow, the vocabulary size will be limited to 5,000 words when vectorizing processed text for logistic regression.

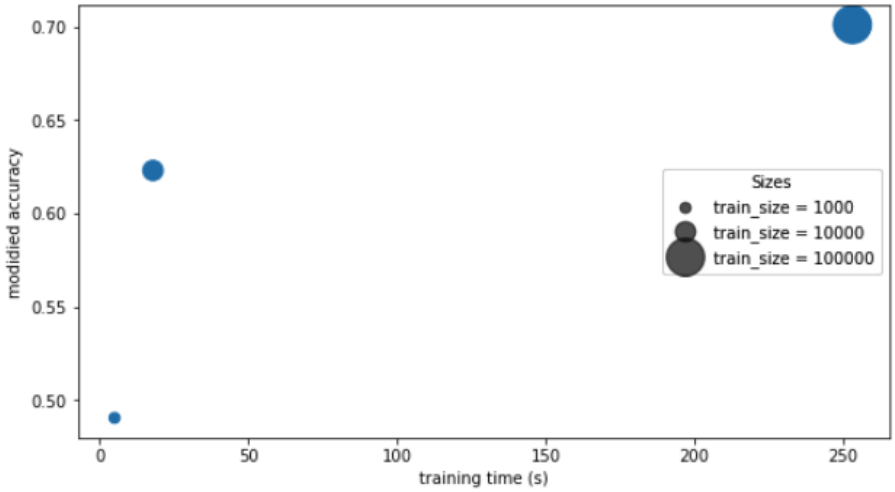


Figure 8: The Performance of the Logistic Regression Classifier for the Full Dimensionality

As a second step in selecting the most suitable classifier, let's determine if limiting the vocabulary for the Naive Bayes classifier could shorten training time without compromising accuracy. Figure 9 demonstrates that the training time for the Naive Bayes classifier ranges between 3 and 5 seconds, depending very slightly on the TF IDF dimensions.

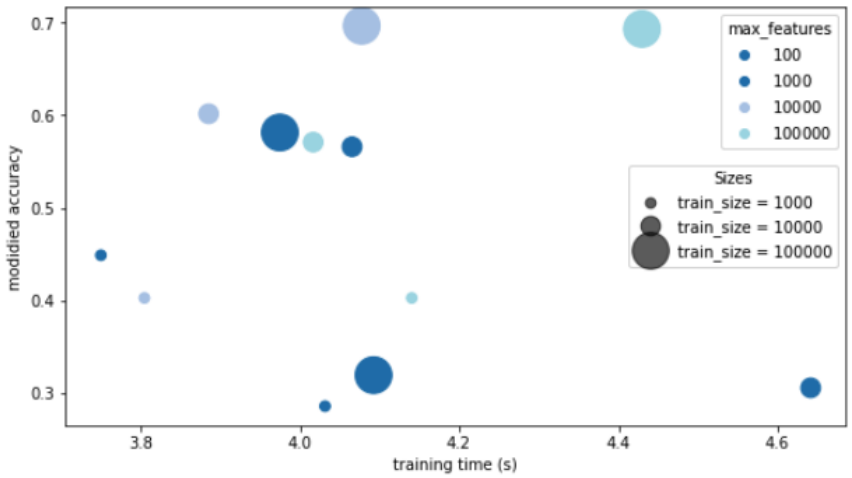


Figure 9: Performance of the Naive Bayes classifier for Different Vocabulary Sizes

Hence, vocabulary reduction is unlikely to be beneficial for this classification model and it will not be implemented.

4.3 Executing Classifiers Comparison Pipeline

The results of the experiment set in section 3.6 of this work are displayed in Figure 10.

	train_sample_size	test_sample_size	lemma	spell	language	time_preprocess	train_acc	test_acc	mod_acc	train_time	classifier
0	100	60000	True	False	True	920.344807	0.835052	0.141413	-0.093221	0.002794	Bayes
1	100	60000	True	False	True	920.344807	1.000000	0.260259	0.203981	0.105788	LogReg
2	1000	60000	True	False	True	918.039529	0.941777	0.449905	0.477958	0.004657	Bayes
3	1000	60000	True	False	True	918.039529	1.000000	0.462196	0.487794	0.869801	LogReg
4	10000	60000	True	False	True	1059.202837	0.788033	0.585478	0.599361	0.014007	Bayes
5	10000	60000	True	False	True	1059.202837	0.928259	0.573458	0.601871	6.050484	LogReg
6	100000	60000	True	False	True	2451.271801	0.731646	0.663185	0.677372	0.111297	Bayes
7	100000	60000	True	False	True	2451.271801	0.739010	0.649353	0.673740	89.439498	LogReg
8	1000000	60000	True	False	True	15834.114975	0.718886	0.693692	0.711923	0.937131	Bayes
9	1000000	60000	True	False	True	15834.114975	0.691913	0.682332	0.704451	1580.845654	LogReg

Figure 10: Results of the Comparison Pipeline for Different Classifiers and Training Sizes

The modified accuracy of the algorithm was not affected when activating the spelling correction layer on small training and test samples. However, it took an average of one second per question, which is too long given the size of the dataset and therefore spelling correction was not included in the comparison experiment.

Lemmatization does not show any substantial increase in the predictive power of the model either and required 5 hours to execute. Since the pre-processing of the whole dataset should not be repeated after each user iteration, lemmatization layer could still be activated as 'a rule of thumb'. Logistic regression classifier makes more accurate predictions on smaller training sizes (less than 100,000 examples). For larger training sizes Naive Bayes Classifier demonstrates the highest modified accuracy score and at the same time the shortest training time. Hence, it is suggested to be implemented in the assistive device.

4.4 Applying the Classification Pipeline with Suggested Settings

Finally, the classifier with the following settings was fitted to provide visualizations and interpret the results:

- Train size – 100,000
- Test size – 10,000
- ML classifier – Naïve Bayes
- Lemmatization, spelling and language correction functions deactivated

Confusion matrix is one of the most popular methods for evaluating the accuracy of predictions made by the classification pipeline. A framework provided by Dennis T (T, 2019) was applied to produce the corresponding confusion matrix (Figure 11) and to create user-friendly visualization.

Figure 11: Confusion Matrix

It reveals the following insights:

- The most frequently correctly predicted categories are Family and Relationships and Computers and Internet (8.66% and 8.48% out of all predictions). However, as shown in Figure 12, the algorithm predicted the greatest number of them – 1,795 and 1,111, respectively.

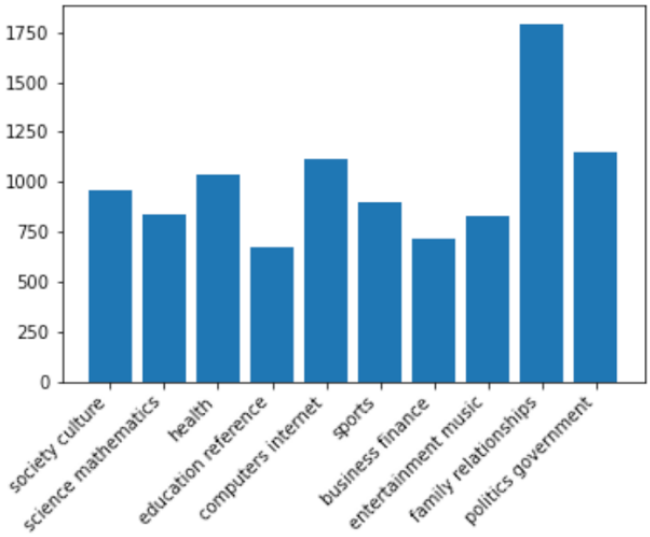


Figure 12: Number of Predictions per Category

- The hardest categories to identify appeared to be Education and Reference and Business and Finance (4.11% and 4.42%)
- The most misclassified pairs are Society and Culture (true labels) versus Family and Relationships (predicted labels), that account for 2.13% of all predictions. The second pair, with 1.61%, is Entertainment and Music (true labels) against Family and Relationships (predicted).

Misclassified questions for some of the categories were printed out to get a deeper understanding of the classifier's reasoning logic. After reading through them, the following patterns emerge:

- Personal rhetorical questions with few specific contexts such as ‘What is happiness?’ or ‘Do others see you the same as you see yourself in the mirror?’ are often misclassified.
- Naïve Bayes does not consider the meaning and instead simply counts important words. In the sentence presented in Figure 13, for example, both music-related words such as ‘drummer’, ‘band’, and relationships-related ‘relationships’ are present. Thus, the classifier labelled this question as Entertainment and Music category instead of Family and Relationships (true label).

```
Relationships? I am the only girl in my band and I like my drummer and he likes me but my new
band manager said that relationships inside the band could destroy the band. Is this true?
```

Figure 13: Example of Misclassification (“Music and Entertainment” instead of “Family and Relationships”)

- The presented pipeline does not support Named-entity recognition (NER)[1] (Marshall, 2019). This results in ignorance of names of the movies, artists, companies, and organizations. For example, in the sentence (Figure 14) the predicted label is Family and Relationships, although the question was obviously about the well-known TV-series ‘Lost’.

```
What do you think will happen in Lost? I know at the end of this series we will not find out the meaning of the islan
d and why the plane really crashed, but I just wanted to know if anna and the other girl who was with her die as a ma
te told me they get shot!!
```

Figure 14: Example of Misclassification (“Family and Relationships’’ instead of “Music and Entertainment”)

[1] Named entity recognition (NER) – sometimes referred to as entity chunking, extraction, or identification – is the task of identifying and categorizing key information (entities) in text (Marshall, 2019).

Conclusion and Future Work

To summarize, the modified accuracy score of around 68% predicting 10 categories with Naive Bayes classifier trained on 100,000 questions requiring less than 0.2 seconds makes the algorithm both accurate and efficient. Given enough of specific context in a question, the classifier would demonstrate an even greater accuracy. The client can either implement the pipeline with the proposed settings, or choose an alternative one, based on the comparative analysis presented throughout this project.

Further extensions of the classification model:

- The NER technique could be applied to use it as a predictor feature for the logistics regression. Each category contains entities, that can easily be spotted. For example, Entertainment and Music contains titles of movies, names of cast, musicians; health – names of deceases, Education and Reference – universities, schools, and colleges; Business and Finance – companies. Spacy EntityRecognizer (EntityRecognizer, 2022) library has some of pre-defined entities. However, the majority of specific categories will have to be defined manually. Also, running logistic regression with the resulting matrix as an input would require significant amount of time making this idea less viable.
- Instead of predicting one category, the classifier could output multiple relevant categories based on probabilities obtained from logistic regression. This 'multiple choice' feature would enable patients to activate the right chatbot more efficiently.
- Pre-trained word embedding linguistic models (for example "glove-twitter-200") could be used to capture more context by performing vectorization based on pre-trained word vectors. Next, either logistic regression or neural networks could be trained as a classification layer. This approach also has its drawbacks, however. First, there may be many out of vocabulary words, that would result into informational loss. Second, the training time would explode in the case of neural networks capturing each word vectors separately.
- As an alternative to training the classifier after each user interaction the training stage could be scheduled daily or weekly, when the patient is not using the device. The computational demand would drastically decrease, and the training time constraint would be eliminated, allowing implementation of more sophisticated NLP classification techniques.

References

Ayanouz, S., Abdelhakim, B. and Benhmed, M., 2020. A Smart Chatbot Architecture based NLP and Machine Learning for Health Care Assistance. Proceedings of the 3rd International Conference on Networking, Information Systems & Security.

Di Pietro, M., 2022. Text Classification with NLP: Tf-Idf vs Word2Vec vs BERT. [online] Medium. Available at: <https://towardsdatascience.com/text-classification-with-nlp-tf-idf-vs-word2vec-vs-bert-41ff868d1794> [Accessed 18 August 2022].

fse/glove-twitter-200. [online] Available at: <https://huggingface.co/fse/glove-twitter-200> [Accessed 18 August 2022].

Gaurav, M., 2021. Comparing performance of a modern NLP framework, BERT, vs a classical approach, TF-IDF, for document classification with simple and easy to understand code.. [online] Datagraphi.com. Available at: <https://datagraphi.com/blog/post/2021/9/24/comparing-performance-of-a-modern-nlp-framework-bert-vs-a-classical-approach-tf-idf-for-document-classification-with-simple-and-easy-to-understand-code> [Accessed 18 August 2022].

IBM Cloud Education (2021). What is Overfitting? [online] www.ibm.com. Available at: https://www.ibm.com/cloud/learn/overfitting.

Ismael Kadhim, A., Cheah, Y., Abbas Hieder, I. and Ahmed Ali, R., 2017. Improving TF-IDF with Singular Value Decomposition (SVD) for Feature Extraction on Twitter. IEC2017 Proceedings Book.

Karani, D., 2018. Introduction to Word Embedding and Word2Vec. [online] Medium. Available at: <https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa> [Accessed 18 August 2022].

Khalid, I., 2020. Cleaning Text Data with Python. [online] Medium. Available at: <https://towardsdatascience.com/cleaning-text-data-with-python-b69b47b97b76> [Accessed 18 August 2022].

Krzeszewska, U., Poniszewska-Marańda, A. and Ochelska-Mierzejewska, J., 2022. Systematic Comparison of Vectorization Methods in Classification Context. Applied Sciences, 12(10), p.5119.

Kulkarni, A., Shivananda, A. and Kulkarni, A. (2022). Natural Language Processing Projects. Berkeley, CA: Apress. doi:10.1007/978-1-4842-7386-9.

Le Glaz, A., Haralambous, Y., Kim-Dufor, D., Lenca, P., Billot, R., Ryan, T., Marsh, J., DeVylder, J., Walter, M., Berrouiguet, S. and Lemey, C., 2021. Machine Learning and Natural Language Processing in Mental Health: Systematic Review. Journal of Medical Internet Research, 23(5), p.e15708.

Marshall, C., 2019. What is named entity recognition (NER) and how can I use it?. [online] Medium. Available at: <https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d> [Accessed 18 August 2022].

MND Association. 2022. What is MND? [online] Available at: <https://www.mndassociation.org/about-mnd/what-is-mnd/> [Accessed 18 August 2022].

Scikit-learn.org. 2022. Machine Learning in Python. [online] Available at: <https://scikit-learn.org/stable/> [Accessed 18 August 2022].

Singh, R. and Singh, S., 2020. Text Similarity Measures in News Articles by Vector Space Model Using NLP. Journal of The Institution of Engineers (India): Series B, 102(2), pp.329-338.

Spacy.io. 2022. EntityRecognizer. [online] Available at: <https://spacy.io/api/entityrecognizer> [Accessed 18 August 2022].

T, D., 2019. Confusion Matrix Visualization. [online] Medium. Available at: <https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea> [Accessed 18 August 2022].

Thaenraj, P., 2021. PCA, LDA, and SVD: Model Tuning Through Feature Reduction for Transportation POI Classification. [online] Medium. Available at: <https://ethaenra.medium.com/pca-lda-and-svd-model-tuning-through-feature-reduction-for-transportation-poi-classification-8d20501ee255> [Accessed 18 August 2022].

Turner, C., Jacobs, A., Marques, C., Oates, J., Kamen, D., Anderson, P. and Obeid, J., 2017. Word2Vec inversion and traditional text classifiers for phenotyping lupus. BMC Medical Informatics and Decision Making, 17(1).

V. Ratz, A., 2021. Multinomial Naïve Bayes' For Documents Classification and Natural Language Processing (NLP). [online] Medium. Available at: <https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6> [Accessed 22 August 2022].

Yıldırım, S., 2020. L1 and L2 Regularization — Explained. [online] Medium. Available at: <https://towardsdatascience.com/l1-and-l2-regularization-explained-874c3b03f668> [Accessed 18 August 2022].]

Appendix A

Enhancing Personalisation of MND Assistive Technology using NLP

Tatjana Shigailow, Supervisor: Dr. Ahmad Abu-Khazneh, Co-Supervisor: James Arney (Rolls Royce, AI Hub)

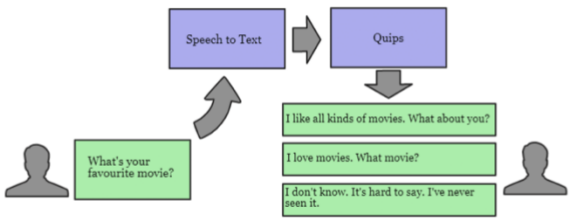
Imperial College London

1. Quips: Problem Overview

Motor Neurone Disease (MND) is a fatal disease that causes rapid loss of muscle movement. Eventually this can lead to an almost **total loss of communication**. While assistive technologies for this exist, the long delays associated with spelling out each word causes frustration, to the point where many people begin limiting their social interactions.

In partnership with the Motor Neurone Disease Association, Roll's Royce AI hub is leading a project that aims to enhance such assistive technologies. So far the project has implemented a prototype - '**Quips**', which is based on a Sequence to Sequence architecture commonly used in Neural Machine Translation (NMT) using PyTorch.

2. Quips Prototype



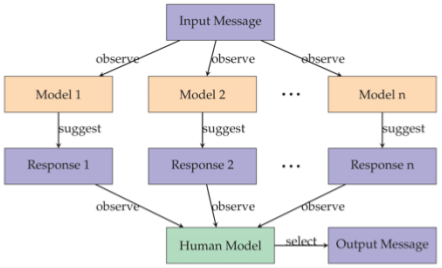
3. ImperialQuips

ImperialQuips is a prototype implemented (since May/June) in this project which is an **incremental upgrade** of Quips built by Rolls Royce. This is a joint work with MSc student Xin Li Huang.

In particular ImperialQuips enhances the architecture of Quips by making each interaction generate possible answers from multiple chatbots (or NLP models) rather than just implementing one chatbot.

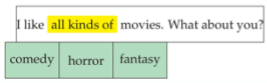
The models available in ImperialQuips include numerous dialogue systems implemented in Facebook's ParlAI framework (see Panel 7).

4. ImperialQuips Prototype



5. ImperialQuips 2.0

It is unlikely that any of the suggested responses in ImperialQuips 1.0 will exactly match what a user would like to say. So the next step is to implement a **personalisation** layer that helps the user to efficiently modify one of the suggested responses from ImperialQuips 1.0.

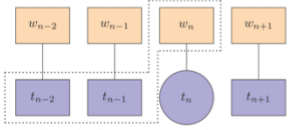


6. N-Gram Tagging

The modification of the responses can be done using **Part-of-speech** (POS) **tagging** - the process of classifying words (**tokens**) into their parts of speech (**tags**), such as nouns or adjectives.

An **n-gram tagger** picks the tag t_n for a token w_n that is most likely in the context of the $n-1$ preceding tags. Thus we are looking for t_n that maximizes

$$P(t_n|t_1, ..., t_{n-1}).$$



Example of n-gram tagger with n=3

7. ParlAI

ParlAI is a python framework for sharing, training and testing dialogue models developed by Facebook. It contains:

- 80+ popular datasets
- a set of reference models
- a zoo of pretrained models

Amongst those **pretrained models** there is:

Wizard Of Wikipedia (WoW)
Dataset: Conversations discussing a randomly chosen topic using passages from Wikipedia articles
Theoretical Novelty: Generative transformer memory network selects the most relevant piece of knowledge and performs an encoding step by concatenating it with the dialogue context
Overview: This model usually offers accurate responses to factual questions.

Convai
Dataset: Conversations between paired crowdworkers getting to know each other, where each is given a role to play based on their persona
Theoretical Novelty: Generative Profile Memory Network stores profile of persona
Overview: This model usually shows a consistent personality.

Empathetic Dialogues (ED)
Dataset: Conversations grounded in an emotional situation related to a given feeling
Theoretical Novelty: Training over this dataset can improve the performance of an end-to-end dialogue system on empathetic dialogue
Overview: This model is usually able to display empathy.

Blended Skill Talk
Dataset: Instances from the three previous datasets
Theoretical Novelty: Existing Retrieval and Generator models are fine-tuned
Overview: This model aims to combine the skills from the models above seamlessly during dialogue.

References

[1] Facebook AI Research. ParlAI Documentation. https://parl.ai/docs/index.html, 2020.
[2] Roler, Dinan, Goyal, Ju, Williamson, Liu, Xu, Ott, Shuster, Smith, Boureau, Weston. Recipe for building an open-domain chatbot. Facebook AI Research, arXiv:2004.13637, 2020.

Appendix B: Classification Pipeline Jupyter Notebook

Import libraries

In [321…

```
import numpy as np
import pandas as pd
#viz
import matplotlib.pyplot as plt
import seaborn as sns
#for text pre-processing
import re, string
import spacy
import langid

#spell-check
from textblob import TextBlob


#TF IDF vectorization
from sklearn.feature_extraction.text import TfidfVectorizer

#for logistic regression and Naive Bayes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import TruncatedSVD
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, f1_score, accuracy_score, confusion_mat
#linguistic models
import gensim.downloader as api
#time performance
import time
```

Openning the file. Performing EDA.

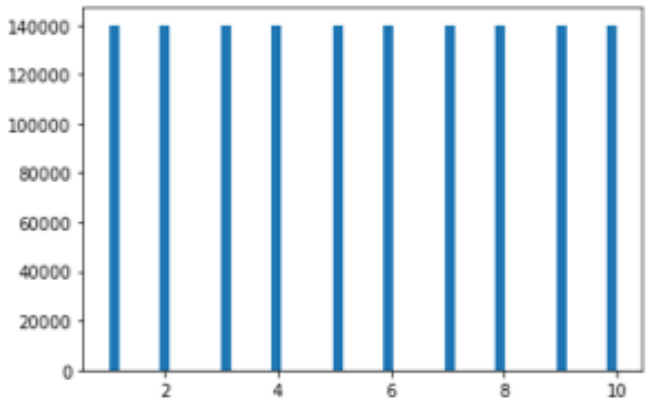
Display distributions of categories for both train and test datasets

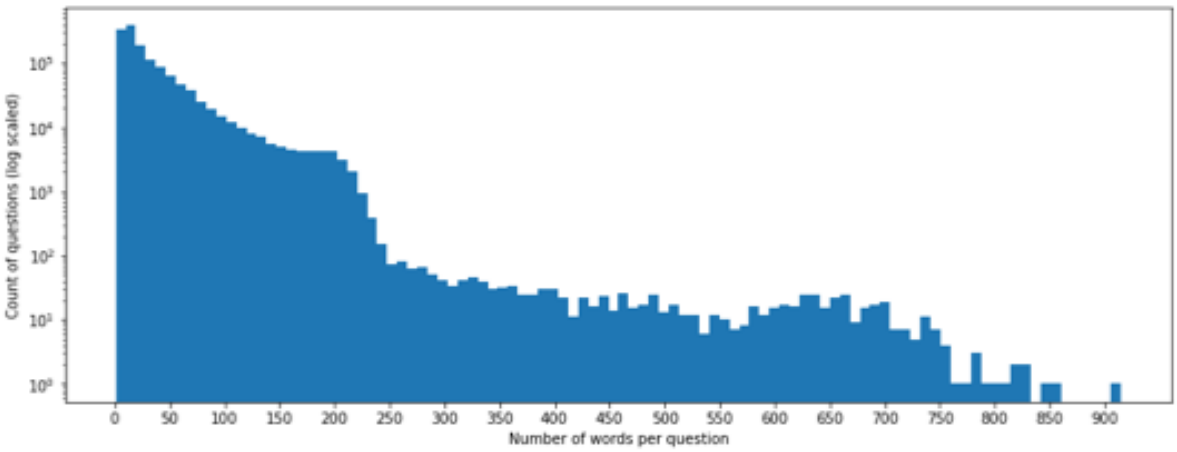
In [225…

```
#load dataset
test = pd.read_csv('test.csv', delimiter=',', header = None)
train = pd.read_csv('train.csv', delimiter=',', header= None)
```

In [472…

```
plt.hist(train[0], bins=50)
plt.show()
```



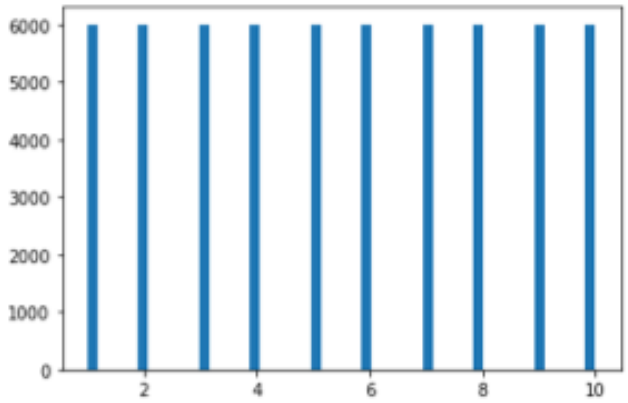


In [31]:
```
#print the longest question in the dataset
for i in train.loc[train['question_len']>900,'question']:
    print(i)
```

```
why me why my life? hello i am a mother ofa 10 week and 3 day old son and i am abused by m
y husband it all started before we got married he get drunk and would tell me he use to be
at his first and 2nd wife but i still married thanking it be diffrert for me but i was wro
ng, first it was yelling, then he would tell me he would hit me he would say stuff like i
was lucky he not hit me because he could really hurt me then the grabbing started he would
grab me by the arm so hard it would leaves marks he would pin me to the wall so i could no
t move and yellthen one day he got so drung and grabed my arm telling me he was going hit
me and i told him i was not scared of him so he told me i batter be hecause he could kill
me then he pushed me so hard a almost fall i finel thongh he calm down so i went to bed he
would come in and yell call me names then he set on the bad and puch me so hard i fell of
the bed, then i got pranget we got in to it he was so drunk he grab my arms and puch in to
the door and for the first time i was scared i though he was going hit me and all i could
thank of was not to let him so i slaped him because i though he would hit my tummy and mak
e me lose my baby then the same nigh he trow rock at me and he would hold his fist up and
tell me he wish i would lose the baby but he never did touch me because he had is drug bud
dy over he told my husband if he wanted any drug he would not touch me so he stoped he fin
el calm down, then when i was 4 months prangent we got in to it he wanted money for the dr
ugs and i wouldnot give it to him so he hit me 2 times on the arm telling me next time it
be my face the he grabed a baseball bat and told if i not give him the money he hit me wit
h it so i gave him the money then he lift but he come back with the stuff and telling me h
was sorry he hit me he was crying he said it never happen again and i told him i for give
him i belive he would not do it again but i was wrong because when i was 6 months prangent
we had no money and no food because he took all the money for the drugs and you can't get
food without money. finely we got a call telling us we could move in with my family so we
could get my husband clean the day befor the move he was getting high on that stuff his li
ghter was out so he was useing a candle and i not want him doing them drugs so i blow out
the candle and he slaped me in the face and all i did was go to bed telling my self it all
stop when we get to texas but i was wrong he started telling me who i can and cn't talk to
where i can and can't go he had to know where i was at ever min even when i went to the ba
throom he would give me money and i have to tell him how much i spent, where and what i go
t finely on march 31 2006 i had my son he was so sweet but that only was for 2 weeks then
the yelling started back he would yell at me at welmart where people could hear he would c
all me names he would tell me woman know nothing and men knew it all then on june 3 2006 w
e went to eat he got drunk he went 6 monthsno dranking so when he got drunk he got mean an
d started yelling at me we was on the side of the road walking he would tell me i was a ba
d mother the he grabed me so hard i though he pull my clothes off he let go of me we got b
ehind this church and for some reason i can't remember why set down on the ground by then
the yelling got worse my son started crying soi was feeding him but he was still in his st
roller my husband we tell me he would not hit me he say he could go to jail if he did then
he hit me so hard on the arm it turn purple it as the first time ever he lift a mark and a
gain he told me he hit my face next time and i told himi was not scared so he started come
ing to me with his fist up telling me i batter be scared. then he put his hands in my face
```

In [473…

```
plt.hist(test[0], bins=50)
plt.show()
```



Manipulating the datasets

Filling the NA values with empty strings. Combining the question body and title in one column – "question"

In [226…

```
train.fillna('', inplace=True)
train["question"] = train[1] +' '+ train[2]

test.fillna('', inplace=True)
test["question"] = test[1] +' '+ test[2]
```

How many words do questions contain?

In [478…

```
train['question_len']=train["question"].apply(lambda x: len(x.split()))
test['question_len']=test["question"].apply(lambda x: len(x.split()))
```

In [507…

```
#lenght distribution
f = plt.figure(figsize=(30,5))
ax = f.add_subplot(121)

ax.hist(train['question_len'],bins=100)
ax.set_xlabel('Number of words per question')
ax.set_ylabel('Count of questions (log scaled)')
plt.xticks(np.arange(0, 915, 50))
plt.yscale('log')
plt.show()
```

```
i was so scared he would hit me again i could not look at him he then knid of man if you d
on't look it makes him worse finel he calm down, the 5 days ago i not fold up the baby dia
per up so he slaped me . i alwa
```

In [59]:
```
#what is the average lenth of question?
np.average(train['question_len'])
```

Out[59]:
```
32.39525785714286
```

Building the pre-processing Pipeline

Removing the non-English questions

In [474…
```
# Define a function
def lng(text):
  a = langid.classify(text)[0]
  return a
```

In [476…
```
# Identify the language
# Estimate the proportion of non-English questions and execution time
sample_size=5000
lang_experiment = train.sample(n = sample_size, random_state = 234)

start=time.time()

lang_experiment['lang'] = lang_experiment["question"].apply(lambda x: lng(x))

# Remove non-English words
lang_experiment_en = lang_experiment.loc[lang_experiment["lang"]=='en' ]

stop=time.time()

lang_execution=stop-start
print('Running time is ',lang_execution, 'secons for 5,000 questions')
print('The proportion of non-English questions is ',(sample_size-len(lang_experiment_en)),
```

```
Running time is  17.91603183746338 secons for 5,000 questions
The proportion of non-English questions is  0.0158
```

In [257…
```
#what languages are present?
lang_experiment.lang.value_counts()
```

Out[257…
```
en    4921
fr      17
pl       8
pt       6
de       6
it       5
es       5
da       3
nl       3
ms       3
br       3
cy       2
af       2
mt       2
no       2
wa       1
ca       1
```

```
vo       1
id       1
eo       1
sv       1
sw       1
hr       1
tl       1
oc       1
fi       1
sl       1
Name: lang, dtype: int64
```

Lemmatization function

In [6]:

```
#download the language model Spacy pipeline
nlp = spacy.load('en_core_web_sm')
def lemmatizeSentence(sentence):
    doc = nlp(sentence)
    lemma_sentence=[]
    for token in doc:
        lemma_sentence.append(token.lemma_)
        lemma_sentence.append(" ")
    return "".join(lemma_sentence)
```

```
/Users/nikigelkott/opt/anaconda3/envs/value/lib/python3.8/site-packages/spacy/util.py:837:
UserWarning: [W095] Model 'en_core_web_sm' (3.0.0) was trained with spaCy v3.0 and may not
be 100% compatible with the current version (3.3.0). If you see errors or degraded perform
ance, download a newer compatible model or retrain your custom model with the current spaC
y version. For more details and available updates, run: python -m spacy validate
  warnings.warn(warn_msg)
```

Spelling correction

In [7]:

```
#function to correct spelling mistakes by words
def correct_word_spelling(sentence):
    raw=TextBlob(sentence)
    corrected=str(raw.correct())
    return corrected
correct_word_spelling('NLP is veery excitiing')
```

Out[7]:

```
'NLP is very exciting'
```

Combining defined functions and Regex pipeline

In [10]:

```
#data cleaning
def preprocess(text,lemma=False,spell=False):
    text = text.lower() #convert words to lower case
    text = text.replace(r'\n', "") #remove trailing newlines
    text = text.replace(r'(@\[A-Za-z0-9]+)|(\r)|(\t)|(<br />)', '') #remove @ and trailing
    text = re.sub(r'\w+:\/{2}[\d\w-]+(\.[\d\w-]+)*(?:(?:\/[^\s/]*))*', '', text) #remove
    text = text.replace(r'([^0-9A-Za-z \t])', "") #remove regular expressions
    text = re.compile('[%s]' % re.escape(string.punctuation)).sub(' ', text) #remove punct
    text = re.sub("\'\w+", '', text) # Remove ticks and the next character
    text = re.sub(r'\w*\d+\w*', '', text) #remove numbers
    text = re.sub('\s{2,}', " ", text) # Replace the over spaces
    text = text.encode('ascii', 'ignore').decode() # Remove unicode characters
    text=re.sub('(\\b[A-Za-z] \\b|\\b [A-Za-z]\\b)', '', text) #remove single letters
    if spell is True:
        text=correct_word_spelling(text)
```

```
    if lemma is True:
        text = lemmatizeSentence(text)
    return text
```

In [11]:

```
#Pipeline in action
print(train["question"][543])
print('#####')
print(preprocess(train["question"][543],lemma=True,spell=True))
```

```
who was the longest driver on the PGA tour in the 1950's? we all know Jack Nicklaus was th
e longest in the 60's, but I'm not sure about the 50's...and Arnold Palmer doesn't count a
s he was mostly a 60's player as well....thanks
#####
who be the long driver on the pa tour in the we all know jack nicklaus be the long in the
but not sure about the and arnold palmer doesn count as he be mostly player as well thank
```

Developing classification pipeline

Customizing the accuracy calculation

Downloading similarity matrix. Please refer to Similarity_matrix.ipynb.

In [375…

```
df_similarities=pd.read_csv('Similarity_df.csv',index_col=0)
```

In [13]:

```
#function to calculate accuracy of prediction based on similarity matrix
def mod_accuracy(Y_pred,Y_true,df_weights):
    scores=[]
    for i,j in zip(Y_pred,Y_true):
        score=float(df_similarities[(df_weights['0'] == i-1) & (df_weights['1'] == j-1) ]
        scores.append(score)
    scores=np.array(scores)
    scores = np.where(scores ==0, 1, scores)
    return np.mean(scores)
```

Helper function for assessing the performance of the classifier

In [270…

```
def fit_performance(classifier,
                    X_train_tf,
                    X_test_tf,
                    train_sample,
                   df_similarities,
                    test_sample):
    fit_start=time.time()
    classifier.fit(X_train_tf, train_sample[0])

    print('Training finished')
    ######TIME MONITORING
    train_time=time.time()-fit_start

    #Training accuracy
    train_acc=classifier.score(X_train_tf,train_sample[0])

    #Predicting
    y_pred = classifier.predict(X_test_tf)


    #Measuring simple accuracy on the test set
```

```
    test_acc=accuracy_score(y_pred,test_sample[0])

    #Measuring custom accuracy on the test set

    mod_acc=mod_accuracy(y_pred,test_sample[0],df_similarities)

    return train_acc,test_acc,mod_acc,train_time
```

In [ ]:

```
test_sample_size=5000
#pre-processing
train["clean_text"]=train["question"].apply(lambda x: preprocess(x))
#remove empty questions
train_cleaned=train[train['clean_text'].apply(lambda x: len(x.split()) > 0)]
# same procedure for test set
train_sample_size=100000
train_sample = train_cleaned.sample(n = train_sample_size, random_state = 234)

test_sample = test.sample(n = test_sample_size, random_state = 234)
test_sample["clean_text"]=test_sample["question"].apply(lambda x: preprocess(x))
test_sample=test_sample[test_sample['clean_text'].apply(lambda x: len(x.split()) > 0)]
tf_idf = TfidfVectorizer()
X_train_tf = tf_idf.fit_transform(train_sample["clean_text"])
X_test_tf = tf_idf.transform(test_sample["clean_text"])

train_vs_test_acc=[]
for pen in [0.1,0.5,1,5,10]:
    for type_ in ['l2',
                  #'l1'
                  ]:
        log_performance=fit_performance(LogisticRegression(multi_class='multinomial',
                                                           solver='lbfgs',
                                        #solver='saga',
                                        C=pen, penalty = type_,
                                        max_iter = 20000),
                                        X_train_tf,
                                    X_test_tf,
                                        train_sample,
                                        df_similarities,
                                        test_sample)
        train_vs_test_acc.append([type_,pen,log_performance[0],log_performance[1]])
```

In [318…

```
results_tuning=pd.DataFrame(train_vs_test_acc,columns =['Type_of_Regularization',
                                                        'Penalty_term',
                                                        'Training_accuracy',
                                                        'Testing_accuracy'])

results_tuning
```

Out[318…

	Type_of_Regularization	Penalty_term	Training_accuracy	Testing_accuracy
0	l2	0.1	0.6058	0.517607
1	l2	0.5	0.7483	0.566026
2	l2	1.0	0.8238	0.583633
3	l2	5.0	0.9629	0.584634
4	l2	10.0	0.9871	0.576831

Logistic Regression optimization via dimensionality reduction

Comparing SVD and vocabulary limitation for the classification with different parameters

In [ ]:

```
test_sample_size=5000
#lists to store results
max_features_stats=[]
svd_stats=[]

#training sizes range (picked by a rule of thumb)
for i in [1000,10000,100000]:
    train_sample_size=i
    train_sample = train_cleaned.sample(n = train_sample_size, random_state = 234)
    ################### NO DIMENSIONALITY REDUCTION
    tf_idf = TfidfVectorizer()
    X_train_tf_full = tf_idf.fit_transform(train_sample["clean_text"])
    dim=X_train_tf_full.shape[1]
    X_test_tf_full = tf_idf.transform(test_sample["clean_text"])
    log_performance_full=fit_performance(LogisticRegression(multi_class='multinomial',
                                  solver='lbfgs',
                                  C=10, penalty = 'l2',
                                  max_iter = 20000),
                                  X_train_tf_full,
                                X_test_tf_full,
                                  train_sample,
                                  df_similarities,
                                  test_sample)

    ################### VOCABULARY LIMIT DIMENSIONALITY REDUCTION
    #max_features range (picked by a rule of thumb)
    for k in [100,500,1000,5000]:
        tf_idf = TfidfVectorizer(max_features=k)
        X_train_tf = tf_idf.fit_transform(train_sample["clean_text"])
        X_test_tf = tf_idf.transform(test_sample["clean_text"])

        log_performance=fit_performance(LogisticRegression(multi_class='multinomial',
                                  solver='lbfgs',
                                  C=10, penalty = 'l2',
                                  max_iter = 20000),
                                  X_train_tf,
                                X_test_tf,
                                  train_sample,
                                  df_similarities,
                                  test_sample)

        max_features_stats.append([log_performance_full,#performance of the large model
                                   log_performance[3],  #train_time
                                   i,                   #train_size
                                   k,                   #max_features
                                   log_performance[2]]) #mod accuracy
    #print('max_features_stats')
    #print(max_features_stats)
    ################### SVD DIMENSIONALITY REDUCTION
    #n_components range (picked by a rule of thumb)
    for r in [100,500,1000]:
        time_reduction=time.time()
        svd = TruncatedSVD(n_components=r, random_state=42)
        train_reduced=svd.fit_transform(X_train_tf_full)
        time_reduction=time.time()-time_reduction #duration of dim reduction for training
        print(time_reduction)
        test_reduced=svd.transform(X_test_tf_full)
        log_performance=fit_performance(LogisticRegression(multi_class='multinomial',
                                  solver='lbfgs',
                                  C=10, penalty = 'l2',
```

```
                                    max_iter = 20000),
                                    train_reduced,
                                  test_reduced,
                                    train_sample,
                                    df_similarities,
                                    test_sample)
        time_train=log_performance[3]+time_reduction
        print(log_performance[3])
        #print(svd.explained_variance_ratio_.sum())
        #print('#########log_performance_svd is ',time_train,i,r,log_performance)
        svd_stats.append([log_performance_full,dim,time_train,i,r,log_performance[2]])
    #print('svd_stats')
    #print(svd_stats)
```

Results visualization

In [275…

```
cols=['train_size',
      'method',
      'dimension',
      'reduced_dim',
      'model_acc',
      'model_time']
```

In [276…

```
DR_results_svd=pd.DataFrame(columns=cols)
full_results=pd.DataFrame(columns=['train_size',
      'method',
      'dimension',
      'model_acc',
      'model_time'])
DR_results_mf=pd.DataFrame(columns= ['method',
      'reduced_dim',
      'model_acc',
      'model_time'])
```

In [277…

```
train_size=[i[3] for i in svd_stats ]
dimension=[i[1] for i in svd_stats ]
model_acc=[i[5] for i in svd_stats ]
model_time=[i[2] for i in svd_stats ]
reduced_dim=[i[4] for i in svd_stats ]
DR_results_svd['train_size']=train_size
DR_results_svd['dimension']=dimension
DR_results_svd['model_acc']=model_acc
DR_results_svd['model_time']=model_time
DR_results_svd['reduced_dim']=reduced_dim
DR_results_svd['method']='SVD'

train_size=[i[2] for i in max_features_stats ]
model_acc=[i[4] for i in max_features_stats ]
model_time=[i[1] for i in max_features_stats ]
reduced_dim=[i[3] for i in max_features_stats ]
DR_results_mf['train_size']=train_size
DR_results_mf['model_acc']=model_acc
DR_results_mf['model_time']=model_time
DR_results_mf['reduced_dim']=reduced_dim
DR_results_mf['method']='max_features'

train_size=[i[2] for i in max_features_stats ]
train_size.extend([i[3] for i in svd_stats ])
model_acc=[i[0][2] for i in max_features_stats ]
model_acc.extend([i[0][2] for i in svd_stats ])
model_time=[i[0][3] for i in max_features_stats ]
```

```
model_time.extend([i[0][3] for i in svd_stats ])

dimension=[[i]*4 for i in [5376,21877,83180]  ]
dimension2=[[i]*3 for i in [5376,21877,83180]  ]
dimension.extend(dimension2)
out = []
for sublist in dimension:
    out.extend(sublist)

full_results['train_size']=train_size
full_results['model_acc']=model_acc
full_results['model_time']=model_time
full_results['dimension']=out

full_results['method']='full_features'
```

In [281…

```
DR_results_svd
```

Out[281…

	train_size	method	dimension	reduced_dim	model_acc	model_time
0	1000	SVD	5376	100	0.401787	0.588181
1	1000	SVD	5376	500	0.474551	2.717817
2	1000	SVD	5376	1000	0.490636	7.011820
3	10000	SVD	21877	100	0.502446	3.629432
4	10000	SVD	21877	500	0.586359	21.886729
5	10000	SVD	21877	1000	0.602314	65.104183
6	100000	SVD	83180	100	0.532952	49.156057
7	100000	SVD	83180	500	0.641980	287.629889
8	100000	SVD	83180	1000	0.667520	638.017513

In [284…

```
DR_results_mf
```

Out[284…

	method	reduced_dim	model_acc	model_time	train_size
0	max_features	100	0.267146	0.169139	1000
1	max_features	500	0.379455	0.308537	1000
2	max_features	1000	0.427773	0.347864	1000
3	max_features	5000	0.490067	1.123761	1000
4	max_features	100	0.327765	1.526420	10000
5	max_features	500	0.486812	2.726712	10000
6	max_features	1000	0.538095	4.569572	10000
7	max_features	5000	0.596103	6.607484	10000
8	max_features	100	0.343622	14.228024	100000
9	max_features	500	0.540179	48.557874	100000
10	max_features	1000	0.591649	58.690612	100000
11	max_features	5000	0.672124	112.431715	100000

In [286…

```
full_results=full_results.drop_duplicates(keep='first')
```

In [287…

```
full_results
```

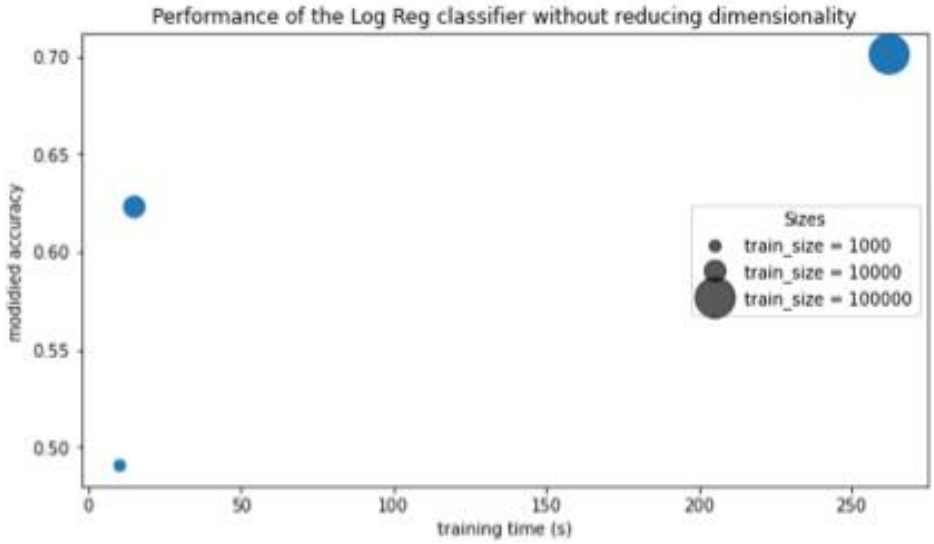
Out[287…

	train_size	method	dimension	model_acc	model_time
0	1000	full_features	5376	0.490636	10.257880
4	10000	full_features	21877	0.623073	15.016202
8	100000	full_features	83180	0.701160	262.399051

In [288…

```
f = plt.figure(figsize=(20,5))
ax = f.add_subplot(121)
scatter=ax.scatter(full_results.model_time,
            full_results.model_acc,
           s=(full_results.train_size)**0.55,
           cmap='tab20')
ax.title.set_text('Performance of the Log Reg classifier without reducing dimensionality')
ax.set_xlabel('training time (s)')
ax.set_ylabel('modidied accuracy')


handles, labels = scatter.legend_elements(prop='sizes', alpha=0.6)
legend = ax.legend(handles, ['train_size = 1000','train_size = 10000','train_size = 100000
ax.add_artist(legend)
plt.show()
```



In [289…

```
f = plt.figure(figsize=(20,5))
ax = f.add_subplot(121)
ax2 = f.add_subplot(122)
scatter=ax.scatter(DR_results_mf.model_time,
            DR_results_mf.model_acc,
           s=(DR_results_mf.train_size)**0.55,
            c=DR_results_mf.reduced_dim,
           cmap='tab20')
ax.title.set_text('Dimentionality Reduction performance by limiting the max_features')
ax.set_xlabel('training time (s)')
ax.set_ylabel('modidied accuracy')
```

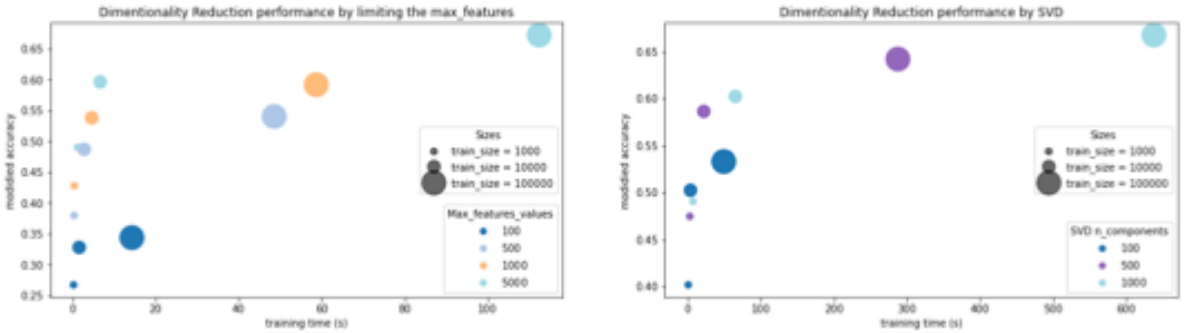
```
legend1 = ax.legend(*scatter.legend_elements(), title="Max_features_values")

ax.add_artist(legend1)
handles, labels = scatter.legend_elements(prop='sizes', alpha=0.6)

legend2 = ax.legend(handles, ['train_size = 1000','train_size = 10000','train_size = 10000

plt.ylabel('Mod_accuracy')
#ax.title.set_text('Model Accuracy for different train sizes')
#ax.set_xticks(DR_results_mf.train_size.values)
#ax.set_xticklabels(DR_results_mf.train_size.values)
scatter1=ax2.scatter(DR_results_svd.model_time,
            DR_results_svd.model_acc,
           s=(DR_results_svd.train_size)**0.55,
            c=DR_results_svd.reduced_dim,
           cmap='tab20')
legend3 = ax2.legend(*scatter1.legend_elements(), title="SVD n_components")

ax2.add_artist(legend3)
handles2, labels2 = scatter.legend_elements(prop='sizes', alpha=0.6)
legend4 = ax2.legend(handles2, ['train_size = 1000','train_size = 10000','train_size = 100
ax2.title.set_text('Dimentionality Reduction performance by SVD')
ax2.set_xlabel('training time (s)')
ax2.set_ylabel('modidied accuracy')
plt.show()
```



Assessing Naive Bayes classifier

Is it possible to reduce the training time of Naive Bayes classifier by reducing the dimensionality?

In [243…

```
# same procedure for test set
test_sample = test.sample(n = test_sample_size, random_state = 234)
test_sample["clean_text"]=test_sample["question"].apply(lambda x: preprocess(x))
test_sample=test_sample[test_sample['clean_text'].apply(lambda x: len(x.split()) > 0)]

#lists to store results
max_features_stats=[]

#training sizes range (picked by a rule of thumb)
for i in [1000,10000,100000]:
    train_sample_size=i
    train_sample = train_cleaned.sample(n = train_sample_size, random_state = 234)
    ################### NO DIMENSIONALITY REDUCTION
    ################### VOCABULARY LIMIT DIMENSIONALITY REDUCTION
    #max_features range (picked by a rule of thumb)
    for k in [100,1000,10000,100000]:
        tf_idf = TfidfVectorizer(max_features=k)
        X_train_tf = tf_idf.fit_transform(train_sample["clean_text"])
        X_test_tf = tf_idf.transform(test_sample["clean_text"])
        log_performance=fit_performance(MultinomialNB(),
```

```
                                  X_train_tf,
                                X_test_tf,
                                  train_sample,
                                  df_similarities,
                                  test_sample)
        max_features_stats.append([i, #train size
                                   k, #max_features
                                   log_performance[2],#mod acc
                                  log_performance[3]]) #time
```

```
Training finished
Training finished
Training finished
Training finished
Training finished
Training finished
Training finished
Training finished
Training finished
Training finished
Training finished
Training finished
```

In [244…

```
df_bayes=pd.DataFrame(max_features_stats,columns=['Train_size', 'max_features','mod_acc','
```

In [245…

```
df_bayes
```

Out[245…

	Train_size	max_features	mod_acc	Time
0	1000	100	0.285589	4.031584
1	1000	1000	0.448697	3.750131
2	1000	10000	0.402516	3.804764
3	1000	100000	0.402516	4.140498
4	10000	100	0.305441	4.641404
5	10000	1000	0.565914	4.065534
6	10000	10000	0.601539	3.885395
7	10000	100000	0.570879	4.016715
8	100000	100	0.319140	4.092558
9	100000	1000	0.581309	3.974872
10	100000	10000	0.696614	4.077611
11	100000	100000	0.693195	4.429498

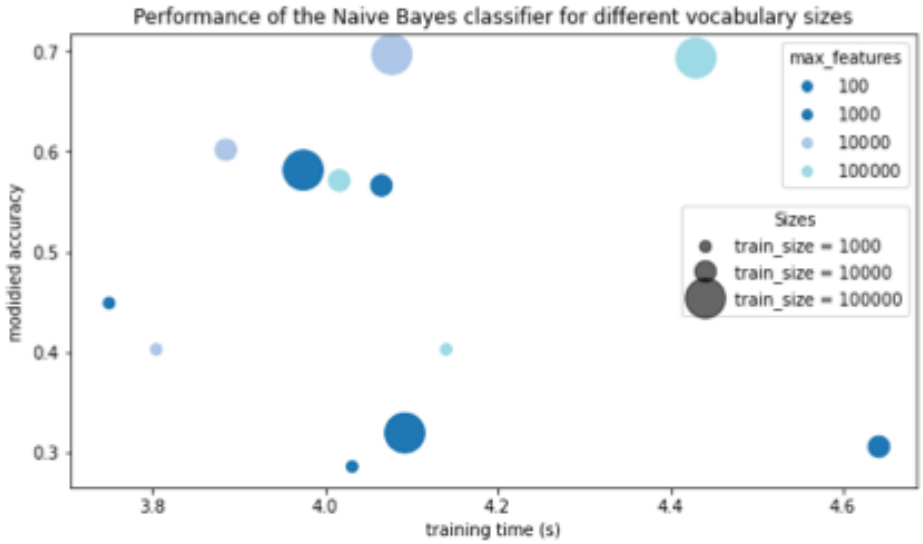
In [253…

```
f = plt.figure(figsize=(20,5))
ax = f.add_subplot(121)
scatter=ax.scatter(df_bayes.Time,
           df_bayes.mod_acc,
          s=(df_bayes.Train_size)**0.55,
            c=df_bayes.max_features,
          cmap='tab20')
ax.title.set_text('Performance of the Naive Bayes classifier for different vocabulary size
ax.set_xlabel('training time (s)')
ax.set_ylabel('modidied accuracy')
```

```
legend1 = ax.legend(*scatter.legend_elements(), title="max_features")

ax.add_artist(legend1)
handles, labels = scatter.legend_elements(prop='sizes', alpha=0.6)

legend2 = ax.legend(handles, ['train_size = 1000','train_size = 10000','train_size = 10000
plt.show()
```



Putting together the pre-processing, TF-IDF vectorization and custom accuracy

In [383…

```
#INPUTS: train and test dataframes and corresponding sample sizes
#OUTPUTS: custom accuracy,preprocessing time,training time,prediction&accuracy accessment

def tf_idf_classifiers_assessment(train_df,
                                  test_df,
                                  train_sample_size,
                                  test_sample_size,
                                 lemma=False,
                                  spell=False,
                                  language=False,
                                 vocab_size=5000):
    #TIME MONITORING
    prep_start=time.time()

    train_sample = train_df.sample(n = train_sample_size, random_state = 234)
    test_sample = test_df.sample(n = test_sample_size, random_state = 234)

    #PRE-PROCESSING


    if language is True:
    # Remove non-English questions
    #identify language
        train_sample['lang'] = train_sample["question"].apply(lambda x: lng(x))
        test_sample['lang'] = test_sample["question"].apply(lambda x: lng(x))
        train_sample = train_sample.loc[train_sample["lang"]=='en' ]
        test_sample = test_sample.loc[test_sample["lang"]=='en' ]

    print('Language check is finished')
```

```
    train_sample["clean_text"]=train_sample["question"].apply(lambda x: preprocess(x,lemma
    test_sample["clean_text"]=test_sample["question"].apply(lambda x: preprocess(x,lemma,s

    #Remove empty questions
    train_sample=train_sample[train_sample['clean_text'].apply(lambda x: len(x.split()) >
    test_sample=test_sample[test_sample['clean_text'].apply(lambda x: len(x.split()) > 0)]

    #Vectorization for Naive Bayes
    tf_idf = TfidfVectorizer()
    X_train_tf = tf_idf.fit_transform(train_sample["clean_text"])
    X_test_tf = tf_idf.transform(test_sample["clean_text"])

    #Vectorization for Log Regression
    tf_idf_reduced = TfidfVectorizer(max_features=vocab_size)
    X_train_tf_reduced = tf_idf_reduced.fit_transform(train_sample["clean_text"])
    X_test_tf_reduced = tf_idf_reduced.transform(test_sample["clean_text"])

    print('Pre-processing finished')

    ######TIME MONITORING
    time_preprocess=time.time()-prep_start

    #TRAIN and predict

    bayes_performance=fit_performance(MultinomialNB(),
                                      X_train_tf,
                                      X_test_tf,
                                      train_sample,
                                      df_similarities,
                                      test_sample)

    log_performance=fit_performance(LogisticRegression(multi_class='multinomial',
                                      solver='lbfgs',
                                      C=10, penalty = 'l2',
                                      max_iter = 20000),
                                      X_train_tf_reduced,
                                    X_test_tf_reduced,
                                      train_sample,
                                      df_similarities,
                                      test_sample)
    results=(train_sample_size,
             test_sample_size,
             lemma,
             spell,
             language,
             time_preprocess,
         bayes_performance,
            log_performance)
    return results
```

In [381…

```
def NLP_pipeline_testing(train_size_range,
                         test_size,
                         language=True,
                         lemma=True,
                         spell=False):
    runs=[]
    for i in train_size_range:
        res=(tf_idf_classifiers_assessment(train,
                                      test,
                                      train_sample_size=i,
                                      test_sample_size=test_size,
                                      language=language,
                                      lemma=lemma,
                                    spell=spell))
```

```
        lst1=list(res[:6])
        lst1.extend(list(res[6]))
        lst1.append('Bayes')
        runs.append(lst1)
        lst2=list(res[:6])
        lst2.extend(list(res[7]))
        lst2.append('LogReg')
        runs.append(lst2)
    dff = pd.DataFrame(runs)
    dff.columns=['train_sample_size',
             'test_sample_size',
             'lemma',
             'spell',
             'language',
             'time_preprocess',
                 'train_acc',
                 'test_acc',
                 'mod_acc',
                 'train_time',
                'classifier']
    return dff
```

In [384…

```
df_results=NLP_pipeline_testing([100,1000,10000,100000,1000000],60000)
```

```
Language check is finished
Pre-processing finished
Training finished
Training finished
Language check is finished
Pre-processing finished
Training finished
Training finished
Language check is finished
Pre-processing finished
Training finished
Training finished
Language check is finished
Pre-processing finished
Training finished
Training finished
Language check is finished
Pre-processing finished
Training finished
Training finished
```

In [385…

```
df_results
```

Out[385…

	train_sample_size	test_sample_size	lemma	spell	language	time_preprocess	train_acc	test_acc	mod_
0	100	60000	True	False	True	920.344807	0.835052	0.141413	-0.093
1	100	60000	True	False	True	920.344807	1.000000	0.260259	0.203
2	1000	60000	True	False	True	918.039529	0.941777	0.449905	0.477
3	1000	60000	True	False	True	918.039529	1.000000	0.462196	0.487
4	10000	60000	True	False	True	1059.202837	0.788033	0.585478	0.599
5	10000	60000	True	False	True	1059.202837	0.928259	0.573458	0.601
6	100000	60000	True	False	True	2451.271801	0.731646	0.663185	0.677
7	100000	60000	True	False	True	2451.271801	0.739010	0.649353	0.673

	train_sample_size	test_sample_size	lemma	spell	language	time_preprocess	train_acc	test_acc	mod_
8	1000000	60000	True	False	True	15834.114975	0.718886	0.693692	0.711
9	1000000	60000	True	False	True	15834.114975	0.691913	0.682332	0.704

In [377…
```
df_results_error
```

Out[377…

	train_sample_size	test_sample_size	lemma	spell	language	time_preprocess	train_acc	test_acc	mod_a
0	10000	1000	False	False	True	67.539774	0.802890	0.570994	0.5834
1	10000	1000	False	False	True	67.539774	0.936196	0.570994	0.6013

Executing classification pipeline to interpret the results

- Train size = 100,000
- Test size = 10,000
- Classifier = Naive Bayes
- Lemmatization = False
- Spell-check = False
- Language function = False

In [393…
```
test_sample_size=10000
test_sample = test.sample(n = test_sample_size, random_state = 234)
test_sample["clean_text"]=test_sample["question"].apply(lambda x: preprocess(x))
test_sample=test_sample[test_sample['clean_text'].apply(lambda x: len(x.split()) > 0)]

train_sample_size=100000
train_sample = train_cleaned.sample(n = train_sample_size, random_state = 234)
tf_idf = TfidfVectorizer()
X_train_tf = tf_idf.fit_transform(train_sample["clean_text"])
X_test_tf = tf_idf.transform(test_sample["clean_text"])

classifier=MultinomialNB()
classifier.fit(X_train_tf, train_sample[0])

print('Training finished')

#Predicting
y_pred = classifier.predict(X_test_tf)


#Measuring simple accuracy on the test set
test_acc=accuracy_score(y_pred,test_sample[0])


#Measuring custom accuracy on the test set

mod_acc=mod_accuracy(y_pred,test_sample[0],df_similarities)
print(test_acc)
print(mod_acc)

cf_matrix = confusion_matrix(test_sample[0], y_pred)
```

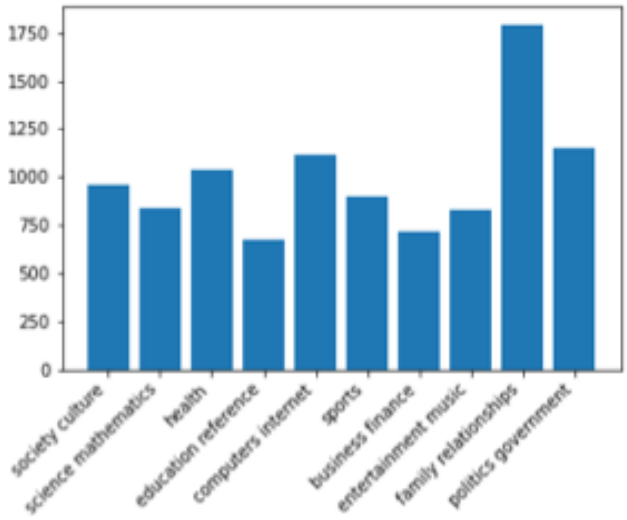
```
Training finished
0.6581974592377713
0.6723798464223887
```

In [339…
```
categories=['society culture','science mathematics',
            'health','education reference',
            'computers internet','sports',
            'business finance','entertainment music',
            'family relationships','politics government']
```

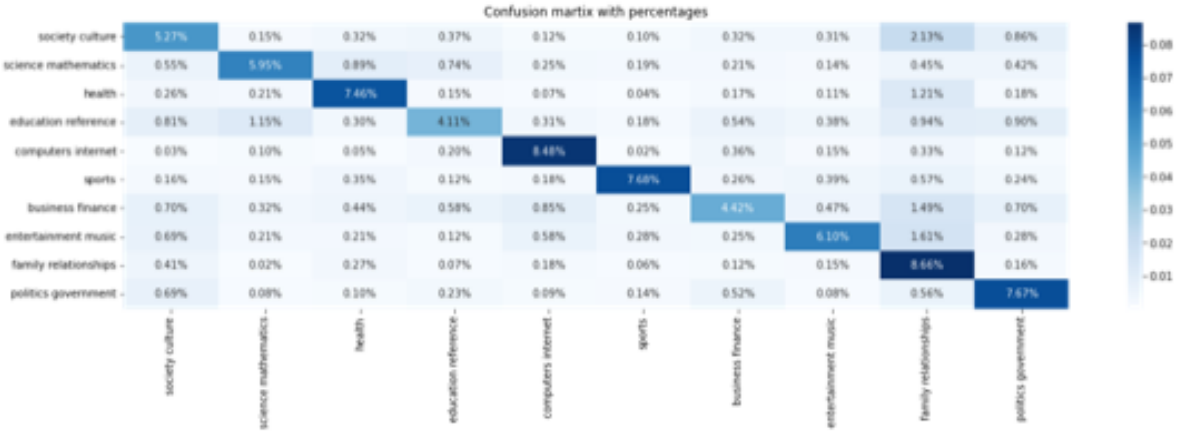
In [463…
```
[[list(y_pred).count(x)][0] for x in set(y_pred)]
```

Out[463…
```
[957, 834, 1039, 669, 1111, 894, 717, 828, 1795, 1153]
```

In [460…
```
#How predicted categories are distributed?
plt.bar(categories,[[list(y_pred).count(x)][0] for x in set(y_pred)])
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [437…
```
f = plt.figure(figsize=(20,5))
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,
            fmt='.2%', cmap='Blues',
            xticklabels=categories,
            yticklabels=categories).set_title("Confusion martix with percentages")
plt.show()
```



Although the accuracy is far from perfect (69%), it is obvious after analyzing different questions that most

of them provide ambibious context or they are just too short. Examples:

'is every girl the only girl if i am not near the girl i love do i love the girl i am near will this mean i have to chose every moment freedom is gone'

The question relates to HEALTH, but I would classify it as a RELATIONSHIP question.

Another reason of missclaffication: Naive Bayes does not take into account meaning but simply counts important words. This raises problem, when keywords from multiple categories are present. for example jobs and universities in the sentence below

'do u know about any part time jobs i am a first year student at the university of technology utech and i need a job to help pay my school fees please help me find a job as soon as possibe'

Explore the missclassidied questions

Predicted – 'family relationships'. Actual – 'society culture',

In [466…

```
n=0
for i in range(len(test_sample)):
    if list(test_sample[0])[i]==1 and y_pred[i]==9 :
        n+=1
        print(list(test_sample["question"])[i],'\n')
        if n==10:
            break
```

```
What do you think about Russians?Are you prejudiced when you meet them?

What should mothers do if their child keeps giving them worms or snakes they find outside?


How to be a Lady? I am wondering... what are the ettiquettes of being a lady...  I love it
when you are at a restaurant and a lady comes in and all the gentlemen stand up. please. i
f you know of any "lady ettiquettes" like for example: ladies should not say bad words...
etc etc etc. it would be really helpfull. THANK YOU !

I recently had a talk with god, and he told me to do terrible things!? Should I do them ju
st because he is the all-powerful one?

What should a fat woman do if a person opens the door when she's on the toilet?

Is it appropriate to kiss on the 1st date?

what do baptisms do?

Christmas presents? okay, there are tons of things guys can get for girls for X-mas. you k
now, jewelry, candy, flowers, stuffed animals, ANYTHING. but what kind of stuff do girls g
et for guys??

Homosexuality to heterosexuality? Can a person who has had sex with the same sex ever enjo
y sex with the opposite sex?if that person wants to change his/her way of life?...is there
any website on this kind of info....homosexualtiy to heterosexuality?

a guy is always starring and talking about me whenever am around am kind of feeling guilty
can i face him .? can i face him and ask him why he make me feel at ease.
```

Predicted – 'family relationships'. Actual – 'entertainment music'

In [469…

```
n=0
for i in range(len(test_sample)):
    if list(test_sample[0])[i]==8 and y_pred[i]==9 :
        n+=1
        print(list(test_sample["question"])[i],'\n')
        if n==10:
            break
```

```
anyone!!?!!??!?!?!!?!? hey if there is anyone out there that likes metal, that worships di
mebag darrell, that thinks that zakk wylde is the best, you need to add me to your friends
list on IM. I am bored. and i like to talk music to people.

Have anyone done a prank on you? If so, what have they done?\nIf you done something, what
have you done to them?\n\n\nThey dumped water all over me and I did it back. And of course
I did some phone pranks (who hasn't? lol)

Scenes from a  hat, What do you say to your dog you could never say to your husband or wif
e? From Who's line is it anyway

Do you want him as your lap-dog? http://static.flickr.com/113/288712560_1fa7718a6b_o.jpg <
----He is soo cute

Alien love? A pair of newly weds win a trip to mars....\nIn the hotel bar they meet an ali
en couple who suggest a partner swap...\nThey decide to go for it!\nThe woman takes her al
ien to bed, and he when he pats his head, his willy gets longer...\nAs he rubs his tummy,
his willy gets wider and harder...\nShe has the time of her life.\nIn the morning she meet
s up with her husband, who is extremely grumpy...\nShe asked if he had a good time (being
as he was so keen)...\nHe said "NO -  the stupid thing just patted my head and rubbed my t
ummy all night..."

What do you think will happen in Lost? I know at the end of this series we will not find o
ut the meaning of the island and why the plane really crashed, but I just wanted to know i
f anna and the other girl who was with her die as a mate told me they get shot!!

I have been invited to many partys this year but end up just sitting on side and not danci
ng, any suggestions? You can call me a shy guy but once i feel comfortable with people i'l
l blaber the whole night. Im just not good at small tallk or picking up girls. I've alread
y had 2 GF but it took along time to finally ask them out. I Just want to go out and have
a fun time without worryinmg what other people think of me.\nThanks for your help

When guys take showers together, do they secretly check each other's "stuff"? I've always
wondered...coz guys are usually so squimish about anything that could show intimacy with a
nother guy (when they're straight of course)

do these lyrics sometimes describe the way you feel? http://lyrics.astraweb.com/display/22
2/tool..aenima..stinkfist.html\n\nsorry i couldnt fit all of them on here\n\nthey often de
scribe the way i feel and i was wondering if i am alone

Want to laugh see details.(I will choose good answer you will get points.)? If you laugh j
ust say yes if not give reason. So here are some jokes which are not mine.\n--------------
----------------------------------------\nThey say that when a man holds a woman's hand
\nbefore marriage, it is love; \nafter marriage it is self-defense \n----------------------
----------------------------------\nA man said his credit card was stolen \nbut he deci
ded not to report it\nsince the thief was spending much less than his wife did.\n---------
----------------------------------------------------\nA little boy asked his father,\n "Dad
dy, how much does it cost to get married?"\nThe father replied, \n"I don't know son, I'm s
till paying."
```

Predicted – 'science mathematics'. Actual – 'education reference'

In [471…

```
n=0
for i in range(len(test_sample)):
    if list(test_sample[0])[i]==4 and y_pred[i]==2 :
```

```
        n+=1
        print(list(test_sample["question"])[i],'\n')
        if n==10:
            break
```

```
What are the factors causing inflation?

do you know something about physics experiment?about "BEDS OF NAIL"? how to explain that e
xperiment?

What is the most common metal in the human body?  ;P?

what is the square root of 28?

convert 905(base10) to its octal equivalent?

how many amps can kill a human?

How much energy does it take to melt a 16.87g icecube? My choices are 102 kj, 108 kj, 936
J, 5.64 kj

non chromosomal genetic element?

What is the difference between ideal gases and real gases?

What cdan you do to slow the rate of global warming?
```

Appendix C: Similarity Matrix Jupyter Notebook

Importing packages and downloading Twitter language model

In [150…
```
import gensim.downloader as api
from nltk.tokenize import word_tokenize
import numpy as np
import pandas as pd
#downloading the glove twitter language model: will take couple minutes
model = api.load("glove-twitter-200")

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```
#display similarity for a random pair of words
model.similarity('dog','cat')
```

Out[2]:
```
0.83243024
```

Creating the customized similarity matrix with corresponding penalties / rewards

Define a function to calcualte pair similarity between a number of words

In [35]:
```
def measure_similarity(list_of_words_1,list_of_words_2):
    similarities=[]
    for i in list_of_words_1:
        for j in list_of_words_2:
            print(i,j)
            similarity=model.wv.similarity(i,j)
            similarities.append(similarity)
            score=np.mean(similarities)
            #mean because some categories contain 2 words (Culture&Society)
    print(similarities)
    return np.mean(similarities)
```

In [36]:
```
#test the function on a given sets of words
measure_similarity(['society', 'culture'],
 ['science', 'mathematics'])
```

```
society science
society mathematics
culture science
culture mathematics
[0.5503056, 0.3251309, 0.5756938, 0.30077365]
/var/folders/lp/kfk0rkcj6bj31b_4y46369qr0000gn/T/ipykernel_17469/3290809302.py:6: Deprecat
ionWarning: Call to deprecated `wv` (Attribute will be removed in 4.0.0, use self instea
d).
  similarity=model.wv.similarity(i,j)
```

Out[36]:
```
0.437976
```

In [46]:
```
measure_similarity(['health'],
 ['science', 'mathematics'])
```

```
health science
health mathematics
[0.5921795, 0.3220128]
/var/folders/lp/kfk0rkcj6bj3lb_4y46369qr0000gn/T/ipykernel_17469/3290809302.py:6: Deprecat
ionWarning: Call to deprecated `wv` (Attribute will be removed in 4.0.0, use self instea
d).
  similarity=model.wv.similarity(i,j)
0.45709616
```

Out[46]:

Apply the measure_similarity function to Yahoo categories

In [22]:
```
categories=['society culture','science mathematics',
            'health','education reference',
            'computers internet','sports',
            'business finance','entertainment music',
            'family relationships','politics government']
```

In [23]:
```
#tokenize each word
categories_processed=[word_tokenize(i) for i in categories]
categories_processed
```

Out[23]:
```
[['society', 'culture'],
 ['science', 'mathematics'],
 ['health'],
 ['education', 'reference'],
 ['computers', 'internet'],
 ['sports'],
 ['business', 'finance'],
 ['entertainment', 'music'],
 ['family', 'relationships'],
 ['politics', 'government']]
```

In [ ]:
```
#measure similarity
pair_similarities = [(str(p1), str(p2),measure_similarity(p1, p2)) for p1 in categories_p
df_similarities=pd.DataFrame(pair_similarities)
```

In [153…
```
#for same pairs of words similarity ==1
df_similarities.loc[df_similarities[0]==df_similarities[1],2]=1
```

In [154…
```
#prettify the dataframe
for i in range(len(df_similarities[0].unique())):
    df_similarities.replace(df_similarities[0].unique()[i], i,inplace=True)
```

Obtain penalties / rewards from similarity scores

In [156…
```
#remove pairs with same categories to scale them later
w_arr=df_similarities[2].to_numpy()
w_arr=list(w_arr)
w_arr=np.array([i for i in w_arr if i!=1]).reshape(-1, 1)
```

Scale the rewards / penalties from -0.5 to 0.5

In [158…
```
scaler = MinMaxScaler((-0.5,0.5))
mod=scaler.fit_transform(w_arr)
```

Plot the resulting rewards / penalties

In [165…
```
#insert 0 scores for same categories  previoulsy removed
mod=list(mod.flatten())
for i in range(0,100,11):
    mod.insert(i,0)
```

In [166…
```
#what is the mean reward / penalty
np.mean(mod)
```

Out[166…
```
-0.014527781009674072
```

In [167…
```
df_similarities[2]=mod
df_similarities.to_csv('Similarity_df.csv')
```

Visualize the resulting penalties / rewards

In [168…
```
sim_matrix=df_similarities[2].to_numpy()
sim_matrix=sim_matrix.reshape(10,10)
f = plt.figure(figsize=(8,8))
sns.heatmap(sim_matrix,xticklabels=categories,
            yticklabels=categories,cmap='Blues')
plt.show()
```