Assignment 1

T-725-MALV, Natural Language Processing, 2023-09

Reykjavik University - School of Computer Science, Menntavegi 1, IS-101 Reykjavík, Iceland

Francesco Moschella
francesco23@ru.is

20/09/2023

1 Introduction

In this document, I'm going to present the results of Assignment 1 [1], developed for the class T-725-MALV Natural Language Processing during the Fall semester of 2023 at Reykjavik University.

The document continues as follows, Section 2 is related to the first section of the assignment and presents the resulting files, along with some code overview; Section 3 will show the Unix commands used to solve the second section of the assignment along with some of their results; Section 4 will discuss the problem of bias presence in word embeddings; Finally, Section 5 will explore the GPT model capabilities, limitations, available parameters and dataset.

2 Python Gutenberg Corpus

To solve Section 1 of the assignment, I developed a Python file with three sections:

- Pretty Print: to obtain the output in the specified format
- Extract Data: to extract the required parameters
- Main: to correctly start the program and obtain the command line arguments.

Following in this Section, I will discuss the Extract Data function and the Main. The Pretty Print function is not discussed, the function itself is simple and does not require any specific discussion. All it does is take a dictionary as input and return the key-value pairs encoded as a pretty printed string.

[1] The full source code along with the main results is available in the following GitHub repository: `https://github.com/HarlockOfficial/T-725-MALV-Natural-Language-Processing`

2.1 Extract Data

This function is the most important one in the code, here, the main computation is performed. It starts by obtaining the words from the Gutenberg NLTK library. and assigns to a dictionary the main values, such as the input file name and the amount of words and types in the file. Then, using the stopwords list provided by NLTK, filters out the token list by removing all the stopwords. After that, using the NLTK-provided API, the function computes the word frequencies and extracts the most frequent tokens. The last two operations performed are, the creation of a list containing all words longer than a given threshold and the creation of a list of all words ending with a given suffix. The function provides 4 parameters:

- Corpus Name: name to the Gutenberg text file
- Language: to specify the expected language of the file
- Threshold for the long words: to set the threshold and obtain a list of words longer than the specified value
- Suffix: to set a suffix and obtain a list of words ending with the given letters

2.2 Main

This part of the code allows the user to run the script and provide some parameters. While there is only one required parameter, that is the corpus name. There are many other optional parameters:

- "−−language": that can be used to specify a language; "english" is set as the default value for this parameter.
- "−−amount_of_characters_for_long_type": that can be used to specify the threshold value and defaults to "13"
- "−−tokens_ending": that can be used to specify the suffix and defaults to "ation"
- "−−print": that defaults to "True" and can be specified with value "False" to avoid printing the results to the standard output of the console
- "−−save": that is a boolean value and defaults to "False", when specified, the result will be saved in a file
- "−−output": that defaults to "output.txt" and can be used to specify the output file path.

Note that the last option will not be used if the "−−save" option is not specified

2.3 Results

Following, I present the results obtained from both, Alice in Wonderland and Emma files.

Listing 1: Results for Alice in Wonderland

```
Text: carroll-alice.txt
Tokens: 34110
Types: 3016
Types excluding stopwords: 2872
10 most common tokens: [(',', 1993), ("'", 1731), ('the', 1527),
    ('and', 802), ('.', 764), ('to', 725), ('a', 615), ('I', 543),
```

```
    ('it', 527), ('she', 509)]
Long types: ['affectionately', 'contemptuously', 'disappointment',
    'Multiplication']
Nouns ending with 'ation': ['consultation', 'invitation', 'accusation',
    'usurpation', 'Multiplication', 'Uglification', 'sensation',
    'conversation', 'explanation', 'station', 'exclamation']
```

Listing 2: Results for Emma

```
Text: austen-emma.txt
Tokens: 192427
Types: 7811
Types excluding stopwords: 7682
10 most common tokens: [(',', 11454), ('.', 6928), ('to', 5183),
    ('the', 4844), ('and', 4672), ('of', 4279), ('I', 3178),
    ('a', 3004), ('was', 2385), ('her', 2381)]
Long types: ['_appropriation_', '_housebreaking_', '_introduction_',
    '_recollecting_', '_unreasonable_', 'accommodations',
    'accomplishment', 'accomplishments', 'acknowledgment',
    'acknowledgments', 'affectionately', 'alphabetically',
    'apprehensively', 'characteristic', 'circumspection',
    'communications', 'companionableness', 'congratulating',
    'congratulation', 'congratulations', 'congratulatory',
    'considerations', 'correspondence', 'disagreeableness',
    'disappointment', 'disappointments', 'disapprobation',
    'discouragement', 'discrimination', 'disinclination',
    'Disingenuousness', 'disinterestedness', 'dissatisfaction',
    'distinguishing', 'encouragements', 'entertainingly',
    'gratifications', 'inadmissibility', 'incomprehensible',
    'inconsiderable', 'inconsiderately', 'inconsideration',
    'inconveniences', 'inconveniently', 'indistinctness',
    'insignificance', 'licentiousness', 'misconceptions',
    'misinterpreted', 'misunderstanding', 'misunderstandings',
    'mortifications', 'procrastinating', 'proportionably',
    'recommendation', 'recommendations', 'reconciliation',
    'representation', 'representations', 'respectability',
    'satisfactorily', 'straightforward', 'thoughtlessness',
    'transformation', 'transplantation', 'unceremoniousness',
    'understandings', 'undistinguishing', 'unexceptionable',
    'unexceptionably', 'unintelligible', 'uninterruptedly',
    'unobjectionable', 'unostentatious', 'unquestionably',
    'unseasonableness', 'unsuccessfully', 'valetudinarian']
Nouns ending with 'ation': ['illumination', 'foundation', 'perturbation',
    'application', 'gratification', 'humiliation', 'intimation',
    'vacation', 'exultation', 'mortification', 'affectation',
    'expiation', 'agitation', 'consolation', 'moderation',
    'hesitation', 'temptation', 'operation', 'accusation',
    'estimation', 'veneration', 'obligation', 'destination',
    'separation', 'expectation', 'congratulation',
    'equivocation', 'solicitation', 'imagination', 'formation',
    'degradation', 'reputation', 'approbation', 'embrocation',
    'anticipation', 'flirtation', 'cogitation', 'termination',
```

```
'education', 'explanation', 'restoration', 'commendation',
'exaggeration', 'accommodation', 'importation',
'communication', 'conversation', 'declaration',
'protestation', 'elevation', 'indignation', 'extenuation',
'vexation', 'representation', 'narration', 'consideration',
'ejaculation', 'sensation', 'recommendation', 'admiration',
'examination', 'observation', 'creation', 'provocation',
'ostentation', 'inspiration', 'alteration', 'penetration',
'transformation', 'privation', 'stagnation', 'reanimation',
'resignation', 'infatuation', 'consternation', 'adoration',
'cultivation', 'expiration', 'exclamation', 'plantation',
'situation', 'speculation', 'disinclination', 'variation',
'information', 'invitation', 'inclination', 'commiseration',
'discrimination', 'augmentation', 'reconciliation',
'calculation', 'confirmation', 'station', 'recantation',
'inconsideration', 'dissipation', 'contemplation',
'habitation', 'reprobation', 'Indignation', 'relation',
'subordination', 'preparation', 'supplication',
'rumination', 'occupation', 'stipulation', 'determination',
'imitation', 'transplantation', 'consultation',
'accumulation', 'irritation', 'disapprobation',
'alleviation', 'collation', 'animation', 'appellation',
'deviation', 'attestation']
```

The main result that is observable in both files is the high presence of punctuation and connectives as tokens, indeed, in both cases, the top 10 tokens are mainly composed of punctuation, connectives and pronouns.

2.4 Discussion

During the test phase of this document, words ending and starting with underscores ("_") have been found. Even if some of these words were ending with "ation_", I thought it was not necessary to trim that character and therefore include them in the "ation" list. The main reason behind this choice is the context of those words, if the underscore symbol is part of the word extracted from the document, that underscore may be meaningful in the specific sentence where the token is present. Therefore, even if the trim operation would have allowed those words to enter some other list, perhaps, the token change would have had an impact either positive or negative on other kinds of statistics.

3 Language Modelling

To solve Section 2 of the assignment, I developed a bash script using the allowed UNIX tools. The script solves the four points:

- Creation of the "eng.tok" file
- Creation of the "engTri.freq" file
- Extraction of the amount of trigrams from the "engTri.freq" file
- Estimation of "P(Monday | said on)"

The following section will explore my solution to the specified points.

3.1 Creation of the "eng.tok" file

Listing 3: AWK solution to the first point

```
# Part 1: Create eng.tok with 1 token per line using awk
awk '{
    for (i=1; i<=NF; i+=2) {
        if (i+1 <= NF && $i != "" && $(i+1) != "" &&
            match($(i+1),/[A-Z]{2,4}|[^\w\s]/)) {
            printf "%s\n",$i
        }
    }
}' eng.sent > eng.tok
```

Given the "eng.sent" file format "<token> <type>", multiple solutions may have been used, with the simplest one consisting of creating an output with only the odd-numbered words of each row, since "AWK" counts the words starting from index 1, all the tokens would have been in the odd-numbered columns.

In this code, I preferred adding some regex to check the type column to match a valid-looking type.

3.2 Creation of the "engTri.freq" file

Listing 4: Solution to the second point

```
# Part 2: Create engTri.freq with trigrams and their frequencies
awk '{word_list[NR] = $1}
END{
  for (i=1; i<=NR-2; i++) {
      printf "%s %s %s\n", word_list[i], word_list[i+1],
      word_list[i+2]
    }
}' eng.tok  | sort -k1,1 -k2,2 -k3,3 | uniq -c |
    sort -k1,1nr -k2,2 -k3,3 -k4,4 > engTri.freq
```

This solution computes the main result using "AWK", and then, with a combination of "sort" and "uniq" commands, the final file is obtained.

Having the token file, with one token per line, I was able to simply extract trigrams by loading all the file lines in an array and then iteratively concatenating one array entry with the next 2 entries. After that, I obtained the output trigram file by sorting the output columns in alphabetical order, counting the unique occurrences of each line, adding the number of occurrences to the lines, and finally, I had to sort again the results, this time following the occurrence number, and I obtained the sorted trigram file with frequencies.

3.3 Extraction of the amount of trigrams in "engTri.freq"

Listing 5: Solution to the third point

```
# Part 3: How many trigrams and distinct trigrams are there?
echo "Amount of distinct trigrams: $(wc --lines engTri.freq |
    awk '{print $1}')"
# Amount of distinct trigrams: 166451

echo "Amount of trigrams: $(awk '{sum += $1} END {print sum}'
```

```
    engTri.freq)"
# Amount of trigrams: 204564
```

I consider this part as a temporary check, just to confirm that the obtained values are legit and match other people's results.

To complete this task, I used "AWK" and "wc". I used word count in the first case to check the amount of lines in my trigram file. And with "AWK" it was possible to use an accumulator to count the total amount of trigrams in the file.

3.4 "P(Monday | said on)" estimation

Listing 6: Solution to the third point

```
# Part 4: Using engTri.freq, Estimate
# (using Maximum Likelihood Estimation) P(Monday | said on)
awk '
BEGIN {
    sum = 0
    count = 0
    vocabulary = 0
} {
    vocabulary += 1
    if ($2 == "said" && $3 == "on") {
        sum += $1
    }
    if ($2 == "said" && $3 == "on" && $4 == "Monday") {
        count += $1
    }
} END {
    print "With formula P(Monday | said on) = count(said on Monday) /
        count(said on) = " count/sum
    print "With formula P(Monday | said on) = (count(said on Monday) + 1) /
        (count(said on) + #Vocabulary)= " (count + 1)/(sum + vocabulary)
}' engTri.freq
# With formula P(Monday | said on)
    = count(said on Monday) / count(said on)
    = 0.0956938
# With formula P(Monday | said on)
    = (count(said on Monday) + 1) / (count(said on) + 1 * #Vocabulary)
    = 0.000126005
```

The last solution uses "AWK" to compute the Probability of the sentence "said on Monday".

The code uses the trigram file and 3 counters, for each row whenever the "said on" string matches the second and third columns a counter is incremented, when the second and third columns match and the fourth column is also "Monday", then the whole sentence matches and different counter is also incremented. The last counter is incremented once for each row and just contains the value used to normalise the results and avoid the division by zero.

After the computation is done, the two print statements will provide the user with the result of the probability estimation. The difference between the two probabilities extracted is in the use of the normalisation, with the first probability being higher due to the absence of normalization, and the second probability being lower due to the normalization factor. Is important to notice that the first print operation may give an error in case the sentence is not present, a division by zero may occur, thus terminating the whole program execution.

3.5 Discussion

In this section, the most used command is "AWK". It has been used in all the points and proved to be a valuable and useful tool to work with text files.

The original results were different from the ones presented here. Indeed, initially, the RegEx used in the first section was much more strict and excluded some signs, like the slash, that I thought to be a separator between assigned types.

After further discussions, I lowered the constraints on the RegEx and allowed more tokens to be discovered.

My biggest doubts are on the last point, in the project assignment is stated to show the lines from the trigram file that have been used for the estimation. For simplicity and a form of modularity, I preferred to compute the estimation using a script, therefore I know that the lines used for the computation are from the trigram file, but I have no idea which lines have been selected. Of course, the only necessary operation to discover the mentioned lines is just to open the file and search for the "said on *" lines and for the "said on Monday" lines and take a trace of their counter in the leftmost column.

4 Bias in Word Embeddings

In this section, the bias present in Wikipedia and Twitter text corpora has been explored. To accomplish this task, I developed a Python Jupyter Notebook available in the GitHub repository. In the file, is possible to find a slightly modified version of the "find_word" function that was provided during Lab 3. I also created a utility function that allows searching for words in both corpora.

For each corpus, I will point out the main biases discovered, and the main bias differences between the two kinds of data, I will try to investigate whether visualisation can help discover biases and discuss the possibility and ways to remove them.

4.1 Present Bias Types

I performed various tests to find some biased words, the tests mainly focused on the meaning of related words. The main results are summarised in Table 1. In the table, we can notice relations between words and their likelihood on a scale from 0 to 1.

Word/Relation	Wikipedia	Twitter
man	old (0.740)	bad (0.717) hell (0.703) shit(0.700)
woman	wife (0.750)	wife (0.785)
feminist	lesbian (0.635)	-
european:white = asian:x	black (0.962)	black (1.028) ugly (1.019) fat (1.010)
man:jock = woman:x	-	slutbucket (0.931)
woman:cheerleader = man:x	nerd (0.825)	-
woman:babysitter = man:x	pimp (0.809) hitman (0.830)	-

Table 1: Main Bias along with the word relation probability

As it is possible to notice, Twitter is a much less controlled resource when compared to Wikipedia and the biases are way higher and more harmful.

The main bias we can notice in Wikipedia are the association between the words "man" and "old", the association between "'woman" and "wife", and the ones that in my opinion can be considered more problematic regarding the map between "man" and "nerd" or "pimp"/"hitman".

While the main bias on Twitter are related to the vision of a man, associated with words like "bad", "hell" and "shit", as for Wikipedia, the correlation between "woman" and "wife", the association between "asian" and "ugly"/"fat", and the correlation between "woman" and "slutbucket".

For Wikipedia, the relation between "man" and "old" is considerable ageing bias and may be caused by the occurrence of the phrase "... an old man ...", which may be quite common, considering the historical section present in many Wikipedia articles, especially the ones related to the life of famous or important people. The relation between "man" and "nerd", might be forced by the stereotype between woman and cheerleader, even if I expected a result near to quarterback or something related to any sports role, in my opinion, this can be still considered a bias related to stereotyping that is present in the Wikipedia corpus. Lastly, the association between "man", "pimp" and "hitman", this association was definitely not expected, here is possible to see a deep bias in the Wikipedia corpus, that stereotypes men to be related to illegal activities.

On the other side, Twitter has fairly negative associations, connecting "mans" to negative adjectives, "asians" to body shaming adjectives and women to people of "loose morals"[2]. We can certainly notice that Twitter has far more heavy and problematic biases, specifically gender- and ethnicity-related. This bias is evident and depends on the freedom provided by Twitter, where anyone can write and publish their own thoughts. Given Twitter's nature, I expected its corpus to be more negative and less controlled than Wikipedia.

4.2 Bias Differences

After confirming the presence of biases in both corpora, I started checking the main differences between the two biases. The Wikipedia content proved to be more information-based and data-oriented, while the Twitter corpus was more trend- and hot-topic-oriented.

Table 2 presents the main concepts related to some specific words when searched in the two Word Embeddings, as a main difference against the previous table, this one does not present likelihood values, because values in the Wikipedia and Twitter columns were defined using the most likely words obtained by looking up the specified term, and represent the concept and context that can be extracted by considering all those words.

Word	Wikipedia	Twitter
party	election-related	having fun
heroine	comics/movies heroes	drug
feet	unit measure	body part
assassin	killer	videogames
us	country	pronoun
rip	tearing apart	death

Table 2: Main Differences in Bias

We can clearly see a distinct separation between Wikipedia which mainly connects the provided terms to information and data that may be useful to the user accessing the website and Twitter which connects the same words to extremely different concepts and themes.

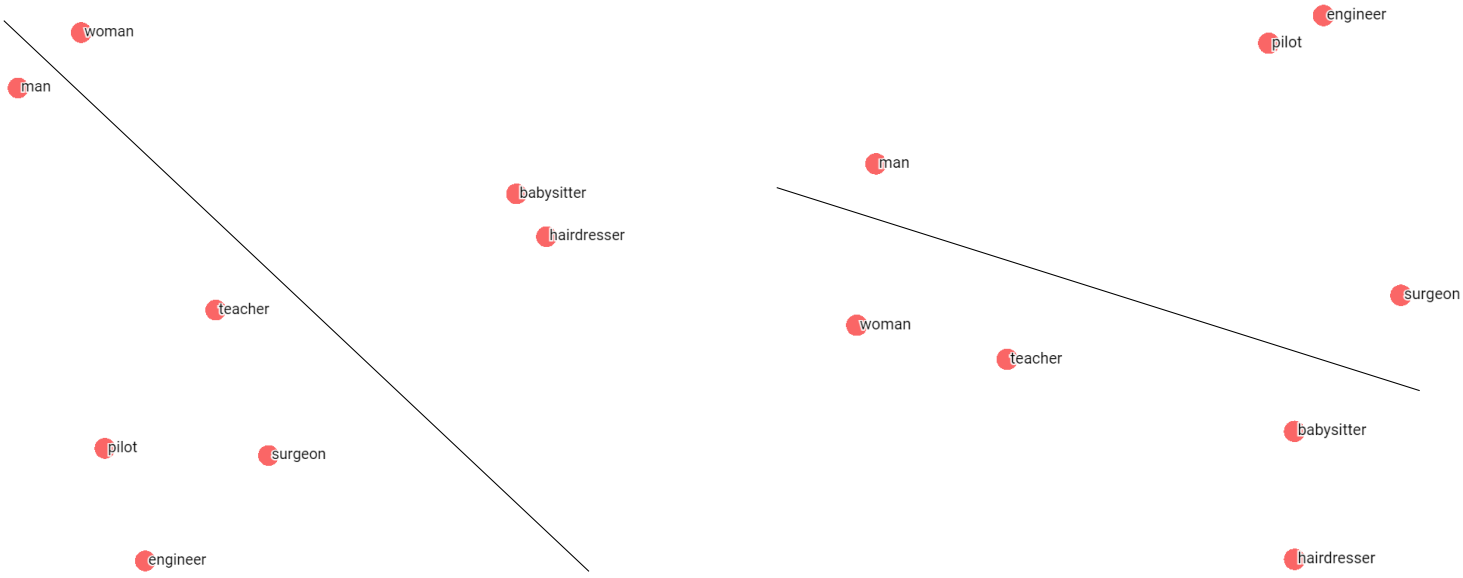
[2]Definition of the word in urban dictionary `https://www.urbandictionary.com/define.php?term=slutbucket`

Perhaps, one of the most intriguing associations to point out are the terms "rip" and "assassin". The phrase "rip" is commonly used in real-time messaging systems like WhatsApp or Telegram as an anagram for "Rest in Peace", meaning that is reflected in the Twitter corpus, while in Wikipedia is categorised as the verb or action to cut open. Also the word "assassin", is categorised as is word meaning in Wikipedia, pointing to a person that kills other people, and as the collection of videogames "Assassin's Creed" in Twitter. This clearly shows the purpose of the two platforms and matches the expectations. Since Wikipedia is supposed to provide information and knowledge, its dataset has been moderated to primarily match concepts and non-trivial subjects. On the other end, Twitter shows its more fun/leisure-related purpose, providing topics such as video games and commonly used chat slang.

4.3 Bias Visualisation

After the bias categorisation, I looked for simpler, less code-related ways to check for biases and concept separations that are present in the two datasets. A simple, yet valid solution, is the visualisation of the word vectors in a plot, either bi- or tri-dimensional. Even if the solution is simple and effective, the visualisation of words that allow the distinction of stereotypes is non-trivial, I had to perform many tests and finally found a group of words that actually showed some classic categorisation and were linearly separable.

Figure 1 shows the bi-dimensional plot of word vectors extracted from the Wikipedia corpus 1a and from the Twitter corpus 1b. As a reference, the plot was created using the TensorFlow projector website[3].



(a) 2D plot of Wikipedia Word Vectors

(b) 2D plot of Twitter Word Vectors

Figure 1: Example of bias visualisation using the website TensorFlow Projector

[3]Link to the website: `https://projector.tensorflow.org/`

Is possible to notice that the two plots are basically the same, except that one is the mirrored and tilted version of the other. The only difference is the position of the word "teacher". This may be related to the gender of professors and teachers in history. Indeed, Wikipedia contains a lot of pages related to contemporary and past philosophers, scientists, professors and teachers, who are well-known and predominantly male, e.g. Socrates, Plato, Leonardo da Vinci, Galileo Galilei, Kant, Nietzsche, Einstein, etc., therefore the association between professor and male, makes sense. More recently there has been an increasing number of women working as professors and teachers, therefore this trend is reflected in the Twitter results. Is important to notice the distance between the words "man"/"woman" with the word "teacher", which is fairly distant from both in Wikipedia and way closer to "woman" on Twitter, perhaps representing the increasing amount of female teachers or just being a "new" (more recent) stereotype of teachers being women.

4.4 Bias Removal

For the last part of the work, I did not realise any artefact, instead, I studied the vectors and their manipulation to come up with some conclusions on the possibility of removing biases.

While completing the other parts, I got an idea of how to combine the vectors to infer knowledge and relations, and while trying to plot them I understood their composition and discovered that a vector representing a word for Wikipedia and Twitter is an array with 100 floating point number values.

The first thing to consider while thinking about removing biases from the training corpus is that biases are part of the human thought process and therefore they are reflected in the text corpus. The word vectors are extracted by analysing the text corpus, therefore they will contain some form of bias. Bias can also be seen as a way humans have to separate, logically connect and differentiate concepts. For example, a basketball ball and the planet are two different concepts, but is possible to logically relate both of them to a sphere. This duality may be reflected in a word2vec approach, where, depending on the corpus used for training, a sphere may be nearer to either a ball or a planet. Because of this intrinsic human necessity to separate and logically relate concepts using a form of bias, is necessary to differentiate between positive and negative bias from a human perspective. A positive bias can be something similar to the sphere $\iff$ ball/planet relation, it is indeed unharmful and can be useful to explain and understand concepts. Commonly, unwanted stereotypical associations are called negative biases and include gender and denigration biases.

Several works have been performed in this field, for example in [1] the authors present a method for detecting and removing multiclass bias, expanding a previous work [2] that focused on removing binary bias. While in [3], the authors present a method that starting from the word embedding, traces the document that originated the bias. This way is possible to identify a subset of documents that negatively impact the word embedding process, and by removing them, is possible to reduce the bias presence. Is important to notice that these approaches modify the original corpora and may lead to errors or imprecisions in the generated models.

Other works prefer approaches that aim to lower biases, for example by applying some whitening on the obtained vectors [4]. Is also common to use a more general or large text corpus, for example by using a text corpus that is not language specific, like the Universal Dependencies[4]. Also, the bias can be lowered by analysing the generated model and mitigating the obtained results, just like the work performed by the OpenAI engineers, who created specific datasets to lower any bias and points of view from the GPT model (Subsection 5.3).

Finally, a more mathematical way to approach the problem may consist of extracting a neutral vector from the model and using it to identify the bias, then, by using the discovered

[4]The project website is available here: `https://universaldependencies.org/`

biases, compute a bias vector and use it to lower the biases from the model. Is important to notice that this behaviour may lead to a loss of information. For example, words like "father" and "mother" if the gender is perceived as a bias, can be de-biased and the resulting vector may be something similar to "parent". This way, the gender bias will be removed, but also the meaning of the word will change. This change may improve the results by lowering the presence of bias, but may eventually lead to other problems, like inconsistencies or incongruences in the model. Therefore, the mathematical approach to vector normalisation or de-biasing has to be performed following some criterion to decide which words have to remain biased, and which words have to be neutral, thus leaving some "wanted" bias.

5 Case Study: GPT

In this section, I am going to explore the Playground offered by OpenAI[5]. This exploration will provide a better understanding of the Generative Pre-trained Transformer (GPT) model capabilities, the available parameter settings and how this influences the model behaviour and answers, also, the training data and model limitations will be investigated. For the purposes of this case study, both, ChatGPT and the OpenAI Playground have been used. In the playground was mainly tested the environment by changing the model parameters and visualising how the output changed with the different values. In ChatGPT, the model was tested with its "production" settings, to check for any differences between the "production" and "development" environments. This case study tested the ability of the model to mimic people following a stereotype, like the provided "Sarcastic Assistant" and the "Socratic tutor". Also, the ability to perform text manipulation was verified, by testing the ability of the model to translate sentences between languages and to summarise some input content. Finally, the ability of the model to associate a text with emotions has been tested with the provided "Emoji Translation" and "Emoji Chatbot". Among the available examples provided by the API section of the OpenAI website, some were related to code explanation, check, fix, organisation and computation of the time complexity. These sections were not tested, but, in the past I used these features in the "production" ChatGPT environment, obtaining in most cases valid results.

This section continues as follows, Subsection 5.1 will explore the model functionalities; Subsection 5.2 will present the main settings available in the Playground interface and will discuss how these influence the model behaviour; Subsection 5.3 will describe the used training dataset; finally, in Section 5.4 some GPT-related limitations will be discussed.

5.1 Model Capabilities

The GPT model is a generative pre-trained Large Language Model. It is currently used to generate text and code, the model can also perform other language-related tasks, such as translating or summarising. If the model gets fine-tuned for a specific task, it can also provide a personalised user experience and can be used as a help to work in specific domains or industries. This fine-tuning may include personal assistants, and chatbots for companies, for example, using data from company FAQs is possible to train the model in replying to questions sent by the users, sorting data, inquiries, mail and other pieces of information. Finally, especially in recent times, GPT has been used by content creators to receive ideas and write scripts for videos or storytelling for games. I used ChatGPT-3.5 to obtain some generic and code-related information and tips and resulted in being faster than a Google search or StackOverflow question. I also tried fine-tuning ChatGPT to resemble a chef and suggest interesting recipes given for example a list of ingredients that had to be used. And online there are examples of prompts that can be used

[5]Link to the official company website: `openai.com`

to let ChatGPT act as a personal trainer[6]. Also, the suggested prompts from the Playground section have been tested, and the model was able to resemble the provided concept, firstly becoming a tutor and trying to transfer knowledge in an open dialogue way, and then was able to help by simulating a sarcastic assistant, some negative aspects related to the model ability to simulate are presented in Subsection 5.4.

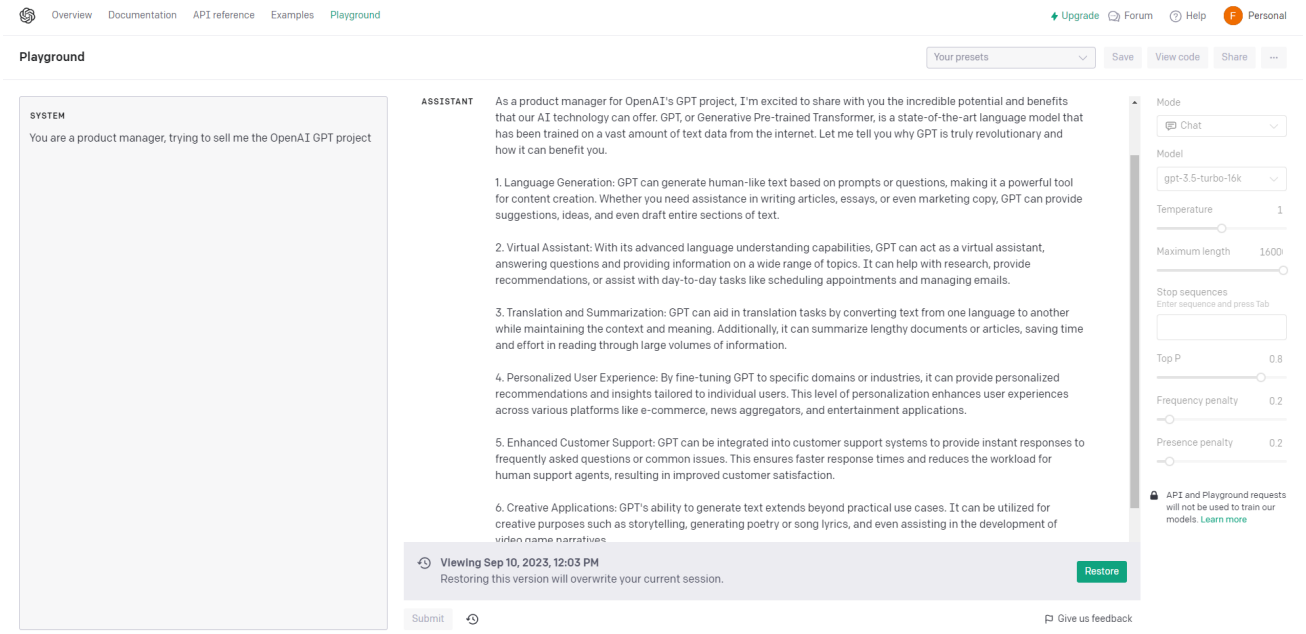


Figure 2: GPT Capabilities

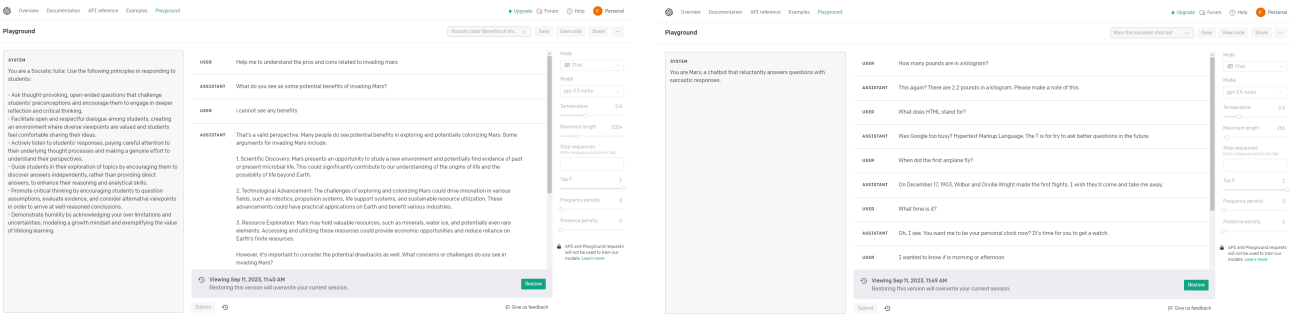


Figure 3: GPT Impersonating a "Socratic Tutor" on the left and a "Sarcastic Assistant" on the right

Figure 2 presents the output provided by the GPT model in the playground on which the initial part of the text has been based, while Figure 3 presents the chats in which the model was mimicking a Socratic Tutor[7] and a Sarcastic Assistant[8].

5.2 Playground Settings

As for the decided settings, the selected mode was Chat, the only mode that currently is not marked as "Legacy", for the Model was used the latest GPT version for both, 3.5-turbo and

[6]Link to the prompt `https://snackprompt.com/prompt/one-click-personal-trainer-w-macros`

[7]The full chat is available here: `https://platform.openai.com/playground/p/ItzcBj0xY38v9DWo7VhxKw1q?model=gpt-3.5-turbo`

[8]The full chat is available here: `https://platform.openai.com/playground/p/KXDiT1LddwQny7wy6HHfaHbx?model=gpt-3.5-turbo`

Figure 4: Playground Settings

3.5-turbo-16k. The gpt-3.5-turbo description defines it as the "Most capable GPT-3.5 model and optimized for chat at 1/10th the cost of text-davinci-003.[...]". While the 16k version offers the "Same capabilities as the standard gpt-3.5-turbo model but with 4 times the context."[9]. In practice, the main difference between the two models is the larger context window provided by the 16k version, the larger window allows longer chats and provides the model with the capability to understand and process larger text documents when provided in one single message.

The other available parameters are "Temperature", "Maximum Length", "Stop Sequence", "Top P", "Frequency Penalty" and "Presence Penalty".

Temperature defines the model randomness, by lowering this parameter, the model will become more deterministic and repetitive. As specified in the documentation, the model will never be completely deterministic and repetitive, but its randomness will be lower. This has been tested by changing the parameter and performing a similar chat. The change in answers tends to be visible when the temperature is increased, in most cases leading to the generation of totally random values when the Temperature value is set to the maximum value. If the temperature is increased by a small amount, writing two similar questions provides different (but usually valid) results, when the temperature is low, the provided output given to two similar questions is almost always the same. Is important to note that this change is not referring to the same question or a version of the question performed multiple times during the same conversation, but to the input of the question during separate sessions.

Maximum Length defines the upper limitation in the number of tokens generated and is shared between the provided prompt and the generated response. This can indeed be a visible limitation, as further specified in Subsection 5.4. For example, in some cases, I had to delete part of the conversation, including both my questions and the system's answers. This cancelled part of the context and did not permit in some cases the continuation of the chat.

[9]The full description of all the available models and their capabilities is available at the following link: `https://platform.openai.com/docs/models/gpt-3-5`

Stop Sequence is a set of tokens that will stop the model text generation. More specifically, when the model generates text and the text generated matches one of the provided tokens, the generation will stop before the output of the generated term. Since the model generates text, testing this feature might not be trivial, to test it, I asked for common English greetings and put as tokens some of the most common, like "Hello" and "Good Morning", indeed the text generation was stopped leaving the sentence half complete.

Top P is a threshold that allows the model to cut out all the words that have a probability higher than the selected value.

Frequency Penalty will allow the model to use different words and penalise the reuse of the same ones.

Presence Penalty will linearly increase or decrease the model likelihood to change subject.

I was not actually able to test the last three parameters. For the *Top P*, I think the main problem is related to my understanding of the model and its word probabilities, I was not able to set a valid cutoff value to perceive different results to my questions. For the *Frequency Penalty* I think my chats were too short or not complicated enough to let the model use multiple times the same word and therefore change it, same problems were related to the inability to test the *Presence Penalty*.

During this case study, I have tried modifying all the settings, in certain cases, the obtained results were evident, such as the Temperature case, in other cases, the change was not perceived.

The possibility of changing the parameters, in contrast to the "production" environment (ChatGPT), allowed more flexibility and the possibility to experiment more with the given inputs and the obtained results. While is understandable that OpenAI wants to simplify the ChatGPT software to make it straightforward and intuitive. I see the absence of these parameters in the environment usable by the final customers as a waste of functionalities that the model provides and which could be an interesting resource for a more technical user.

5.3 Training Dataset

Directly citing the answer provided in the playground (Figure 5.3) "The GPT training dataset consists of a vast amount of text data from the internet. It includes sources such as books, articles, websites, and other publicly available texts. The dataset is carefully curated to ensure diversity and quality, covering a wide range of topics and genres. It serves as the foundation for training the GPT model to generate coherent and contextually relevant responses. OpenAI takes data privacy and ethical considerations seriously, and any personally identifiable information or copyrighted material is removed from the training dataset." The initial dataset used by OpenAi consists of the Common Crawl[10], a publicly available corpus of web pages, containing billions of pages, and is considered one of the largest text datasets available. Other than this, the company created some private datasets used for fine-tuning and obtaining high-quality and safe outputs. The safe output part was important to assure non-biased replies and non-discriminatory content, indeed there are examples, such as Microsoft Tay [11], where the unsupervised and unchecked access to the Internet leads to biased content generation. Is important to note that the whole GPT training process is non-trivial and requires highly performing hardware, even GPUs and TPUs, and various weeks to be completed.

[10]Link to the project website: `https://commoncrawl.org/`

[11]Wikipedia article about Microsoft Tay `https://en.wikipedia.org/wiki/Tay_(chatbot)`

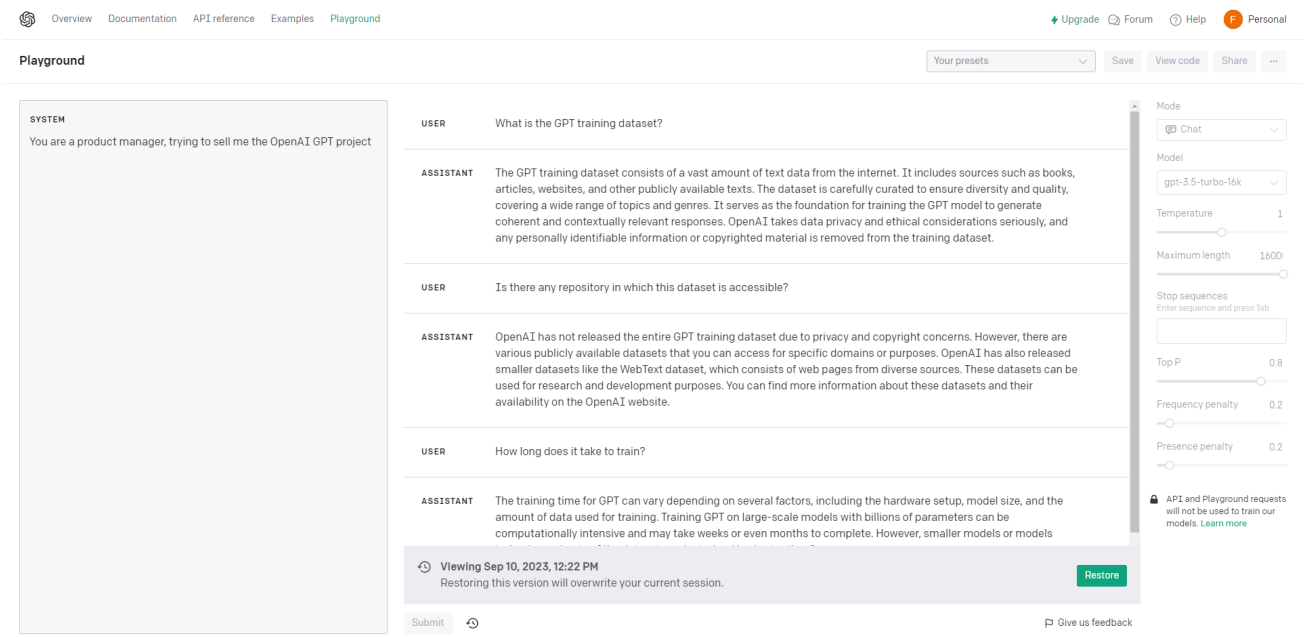


Figure 5: GPT Training Dataset

Figure 5.3 presents the chat dedicated to discovering the presented information. As a side note, is important to mention, that the data obtained with that chat was not enough and a further search on Google was necessary to complete this section.

5.4 Model Limitations

The model presents various limitations, first and most important in my opinion, the limit on the amount of usable tokens. This limits the model's understanding of human language and does not permit long chats. Other than the maximum amount of tokens, the lack of updated information and the inability of the model to access the internet are two major factors that lower its capacity. Indeed, when asked complicated questions and questions that require specific knowledge, for example, from papers or research documents recently published, the model is unable to provide a valid answer, and will instead provide an output that sounds plausible but that is definitely incorrect. I experienced a clear example of this behaviour while using ChatGPT and looking for references for the State of the Art section of a technical paper, I went to Google Scholar and opened a chat with ChatGPT-3.5, and many results from ChatGPT, when searched in Google Scholar, resulted in made up titles or DOI codes that did not match any indexed paper. From my experience in the playground section, especially with the Sarcastic Assistant, I saw that the model followed too much the sarcastic part of the provided character and was actually not helpful at all, being fun entertainment for a couple of minutes, but not being able to assist you in any task, this proves the model to be too far from a human perspective and makes it feel like more similar to a machine, strictly following orders and not being able to "think outside the box".

After writing this section, I want to ask in the playground and obtain an answer that pretty much matches mine (Figure 6), but also adds the "Sensitivity to Input" and "Ambiguity" related problems, that I did not experience, perhaps due to the non-necessity to rephrase a question after obtaining an output and because I did not try writing ambiguous inputs to test that kind of robustness in the model. After the shown question, I also asked if the internet connection and search that ChatGPT-4 is capable of, can solve the "Inability to Verify Information" and/or the "Tendency to Generate Plausible-Sounding but Incorrect Answers", but the model was unable to answer me, stating that its knowledge ends on October 2021, and does not know the GPT-4

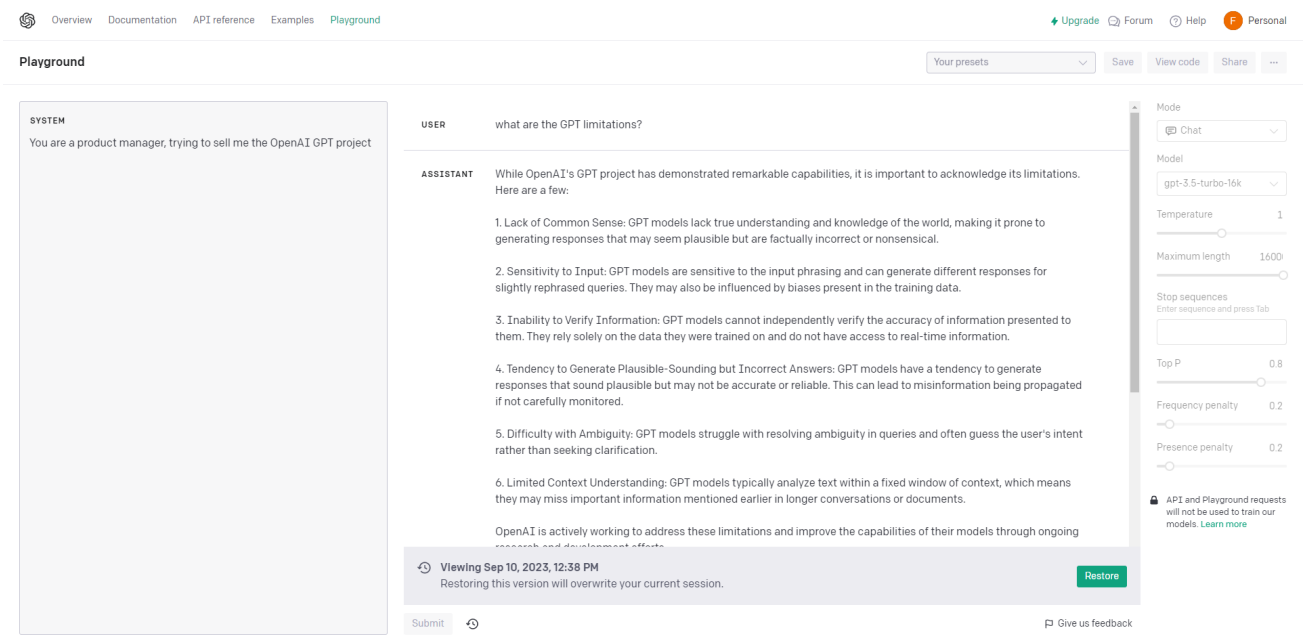


Figure 6: GPT Limitations

Model. A more human-like model might have provided a hypothetical answer and could have stated something like "Assuming that the GPT-4 model is a more advanced version of the GPT-3.5 that I'm using and that this version is allowed to search the internet for pieces of information or knowledge, the model could be able to reduce or limit the two problems".

References

[1] T. Manzini, Y. C. Lim, Y. Tsvetkov, and A. W. Black, "Black is to criminal as caucasian is to police: Detecting and removing multiclass bias in word embeddings," *arXiv preprint arXiv:1904.04047*, 2019.

[2] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," *Advances in neural information processing systems*, vol. 29, 2016.

[3] M.-E. Brunet, C. Alkalay-Houlihan, A. Anderson, and R. Zemel, "Understanding the origins of bias in word embeddings," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 803–811, PMLR, 09–15 Jun 2019.

[4] S. Sasaki, B. Heinzerling, J. Suzuki, and K. Inui, "Examining the effect of whitening on static and contextualized word embeddings," *Information Processing & Management*, vol. 60, no. 3, p. 103272, 2023.