**Analyzing My Conversations With ChatGPT**

In recent years, millions of people, and especially students, started using ChatGPT to get help for their learning programs, courses, assignments, projects etc. I am a part of this group in question, as I need some help from ChatGPT for my school work. Because of that, I wanted to use my ChatGPT conversations data.

Starting point of this project was to analyze my ChatGPT usage to uncover patterns, common topics, and trends, and to visualize these insights. Possible research questions were:

1. In which topics, I need help from ChatGPT?/What are the most common topics I discuss with ChatGPT, and how do they change over time?
2. How does my usage intensity vary across different time periods (daily, weekly, or during key life events)?
   * For example, how my usage increase when I have a midterm soon?
3. How is my attitude towards ChatGPT?
4. Does ChatGPT increase my productivity?

**Research Question**

During the exploratory data analysis, I decided to lean towards one specific hypothesis and work on that. The final research question is actually a mix of questions given above:

“For which courses, I needed help from ChatGPT the most?”

**Hypothesis**

For this research question, naturally, I have put forward one null hypothesis (H0) and an alternative hypothesis (HA). These hypotheses are given below as:

H0: “The amount I needed help from ChatGPT does not change from course to course.”

HA: “I needed more help for some courses compared to others.”

After hypothesis testing, I was going to either fail to reject the null hypothesis or reject it and continue with the alternative hypothesis. Then continue with this to find the course I needed help the most.

**Data Extraction and Format**

ChatGPT has a built-in feature to export your data. I got datasets for:

* Chat logs where each different conversation is separated with proper titles. It was in HTML format.
* Conversations with more detailed information, such as the used model of GPT, timestamps of messages, sender of each message (user or GPT) etc. It was in JSON format
* Feedback I gave to ChatGPT’s answers as negative or positive. It was in JSON format.
* The conversations I shared outside of the ChatGPT. It was in JSON format.
* Model comparisons of GPT. It was in JSON format.
* Saved users data, which is my account. It was in JSON format.

I decided to use the “conversations.json” file, since it included the most detailed data, and it was easy to use in the project code. However, the topic of each message wasn’t labeled, so it was my responsibility to add them.

**Preprocessing and Exploratory Data Analysis**

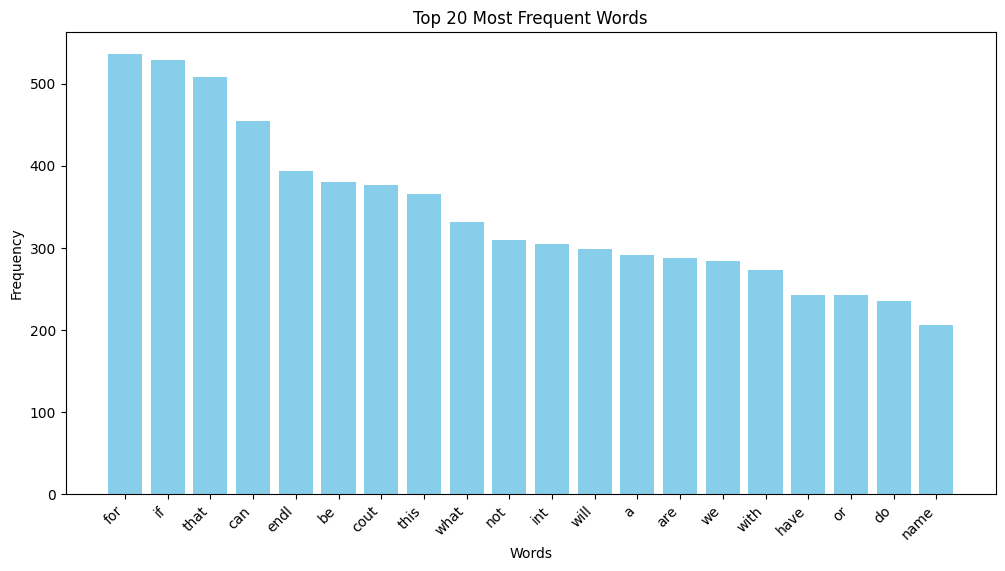
Since I am working with textual data, I needed to operate some preprocessing on the data. Firstly, I iterated over all the conversations and put all my own questions into myMessage class object. These objects have content, timestamp, tokens, topic properties. At first, tokens and topics are empty, but they will be added later through the process.

I kept track of usages of all words and created a bar chart for most used words. Naturally, the first common words were the cornerstones of English language like “the, a, of, and, to, i, is, in, it, you, for, if, that, can, be, this, what, not”. I realized some words specifically coming from certain contexts such as “endl, cout”.



**Fig.1:Top 20 Most Frequently Used Words Raw Data**

Next, I defined the top 10 most frequently used words as **stopwords**, and at least eliminated the huge difference between “the” and others.

**Fig.2:Top 20 Most Frequently Used Words Without the Stopwords**

Then continued preprocessing with lowercasing, tokenization by word, removing non-alphabetic tokens and stopwords, and finally lemmatization of tokens. Finally, I assigned these tokens to each message object’s “tokens” attribute.

lemmatizer = WordNetLemmatizer()

stop\_words = ["the", "a", "of", "and", "to", "i", "is", "in", "it", "you"]

def preprocess\_content(messageObj):

tokens = word\_tokenize(messageObj.content.lower())

# Lowercase and tokenize

tokens = [word for word in tokens if word.isalnum()]

# Remove non-alphanumeric tokens

tokens = [word for word in tokens if word not in stop\_words]

# Remove stop words

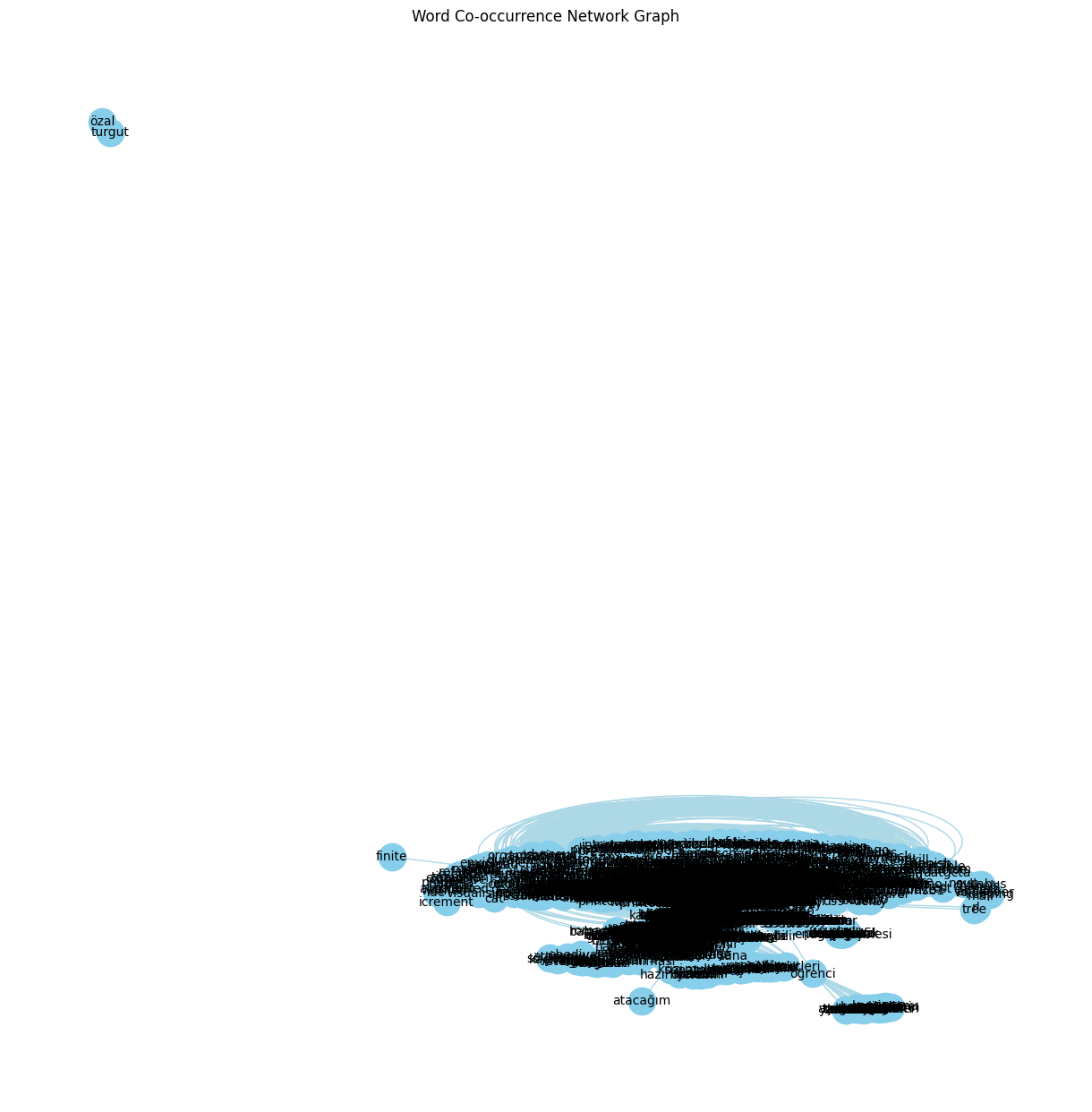
tokens = [lemmatizer.lemmatize(word) for word in tokens]

# Lemmatize words

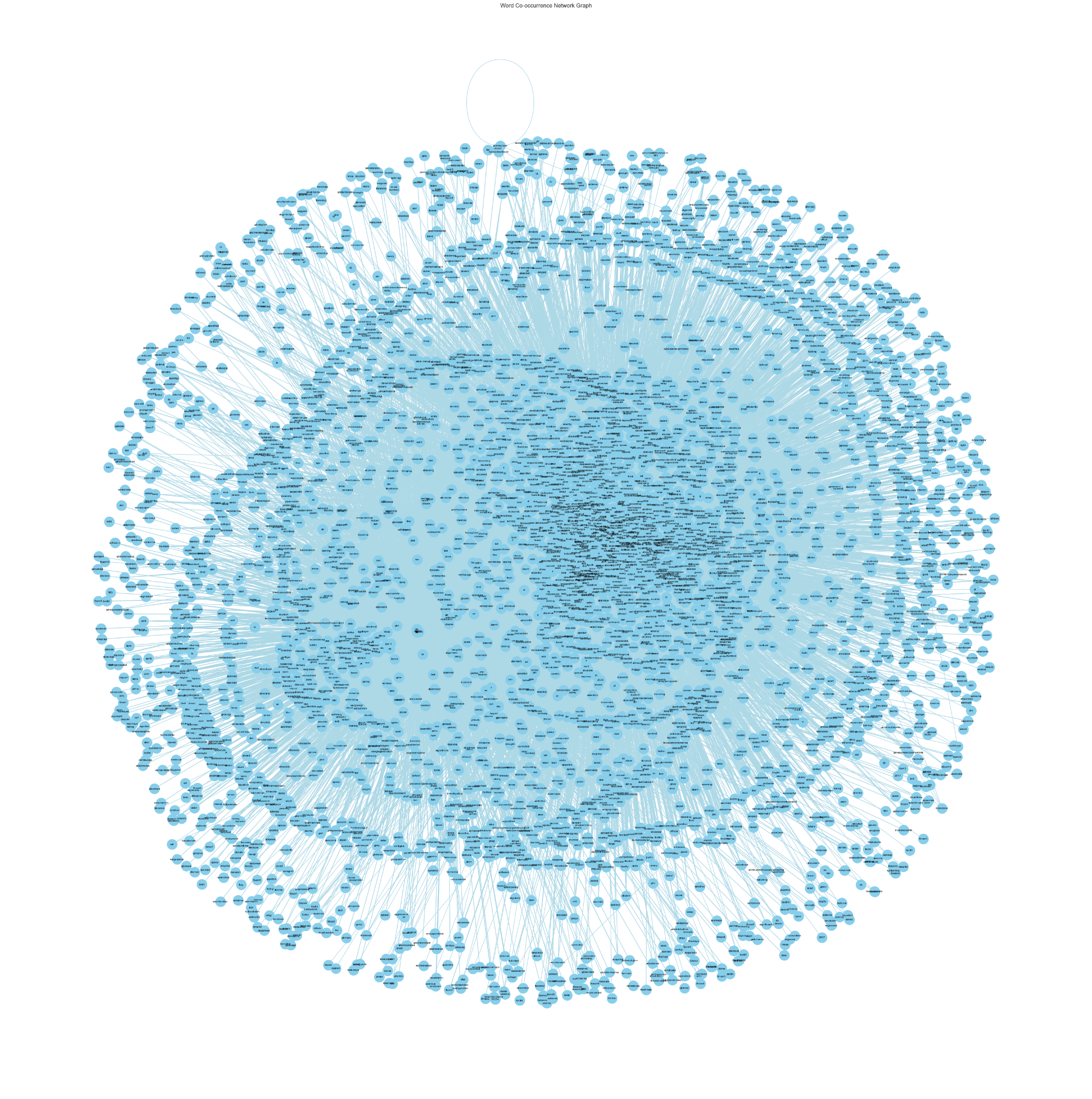
tokens = [word for word in tokens if not word.isdigit()]

return tokens

For the exploratory data analysis, I thought that if I created a network map of words in which each node is connected by their co-occurrence by message, with other words, then I would have an idea about the topics from cumulations. Basically, I was expecting some words that are used in the same context to be closer together in the map. I had some tries to find better representations; however, even after the improvements on the graph, there were no distinct groups of words. Except, Turkish words and English words were kind of separated. In addition, I wanted to add the first ever graph I got even though it was not understandable at all, because nodes “turgut” and “özal” being separate from all the other nodes which indicated that I basically asked about Turgut Özal and never again in another message. That was unexpected and seemed interesting to me.



**Fig.3 Failed First Try of Network Representation**



**Fig.4 Final Network Representation**

**(For better quality:** [**WordCooccurenceNetworkMap\_v6.png**](https://drive.google.com/file/d/1rMPHXgU2I8e5dbEsnxvha3q1xQKIoaEv/view?usp=sharing)**)**

Still couldn’t get the expected distinct groups of words in the final version, so I decided to try different ways to label each node and represent them later.

**Classification: Using Machine Learning Models and Large Language Models**

1. **Large Language Model BERT**

BERT is a [deep learning](https://www.coursera.org/articles/what-is-deep-learning) language model designed to improve the efficiency of [natural language processing (NLP)](https://www.coursera.org/articles/natural-language-processing) tasks. It is famous for its ability to consider context by analyzing the relationships between words in a sentence bidirectionally. It was introduced in October 2018 by researchers at [Google](https://en.wikipedia.org/wiki/Google).

( <https://www.coursera.org/articles/bert-model> )

I created a list of topics consisting of my university courses and gave BERT my messages and this topic list so that it could classify each message into categories.

["History",

"Computer Science and Programming",

"Mathematics",

"Natural Sciences",

"Literature",

"Electronic Circuits",

"Language",

"Health",

"Personal Life",

"Other",

"Digital System Design",

"Data Science"

]

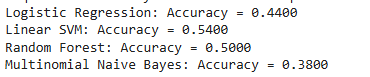
However, BERT had two big problems. Firstly, it took so much time to process each message, since there were too many messages. It took BERT to process some of the messages for more than 5 hours until my code gave me a runtime error. I was going to fix the error and try again, but when I checked BERT’s classifications, there were too many messages labeled as “Other”. As a result, I didn’t find it very accurate for the classification and decided to continue with another way.

1. **Supervised Learning Models**

I manually labeled 250 messages into the same categories mentioned above. Then used 4 different supervised ML models to test:

* Logistic regression
* Linear SVM
* Random Forest
* Multinomial Naive Bayes

Considering the test results and the accuracies, I decided to increase the labeled sample data size, because the results weren’t satisfactory.



**(Model Accuracy Comparison With 250 Samples)**

After the first test, I increased the sample size and again manually labeled 500 messages, saved them, then used the labeled messages to test these machine learning models again.



**(Model Accuracy Comparison With 500 Samples)**

Contrary to my expectations, accuracy rates did not increase, some of them even decreased. At this point I decided to use Linear SVM, as it had the most accurate results. I gave the labeled samples to the model, it then labeled all the other messages. Finally all the messages were labeled into topics/my university courses.

**Hypothesis Testing**

To test my hypothesis, I had to look at the message count in each category and display some tests on the data. Even though there was a gap between “Computer Science and Programming”, I decided to do the “Chi-Square Test” on the data to mathematically test and lean towards one of the hypotheses.

**Observed counts:** Counter(

{'Computer Science and Programming': 368,

'Personal Life': 127,

'Other': 112,

'Mathematics': 96,

'Literature': 78,

'Natural Sciences': 73,

'History': 38,

'Data Science': 32,

'Health': 31,

'Digital System Design': 30,

'Language': 27,

'Electronic Circuits': 26})

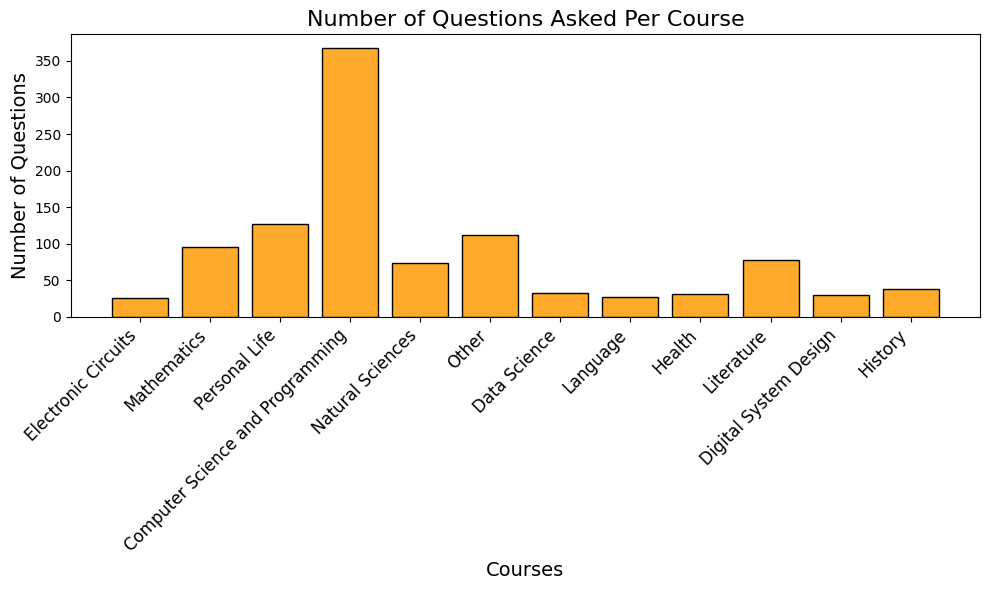
**Chi-Square Test Results:**

**Chi-Square Statistic:** 1163.849710982659

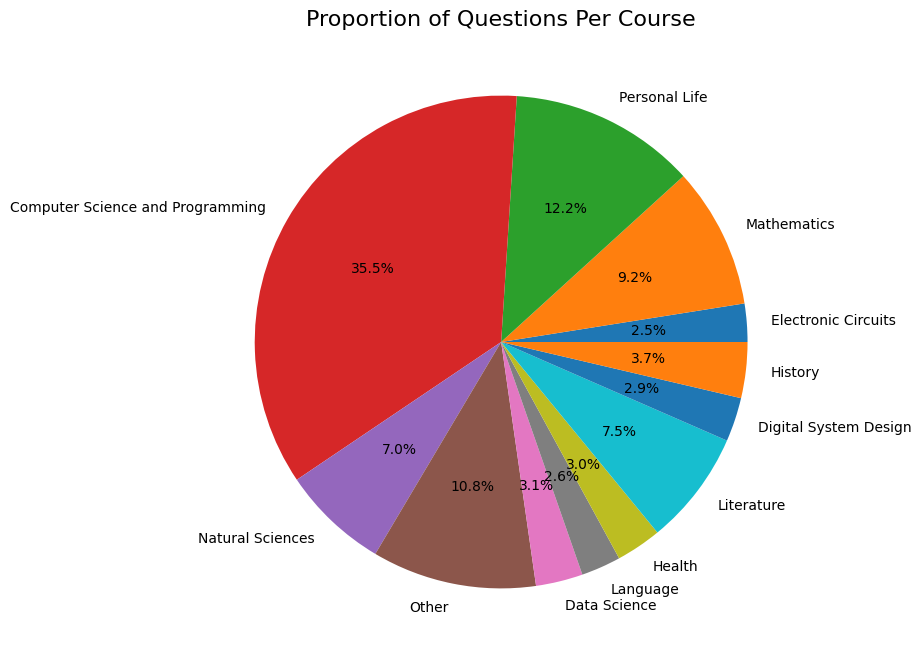
**P-Value:** 9.992088837402096e-243

**Reject the null hypothesis:** There is a significant difference in the number of questions asked about courses.

**Final Visualizations Of Message Count Per Category**

****

**Fig.5**

****

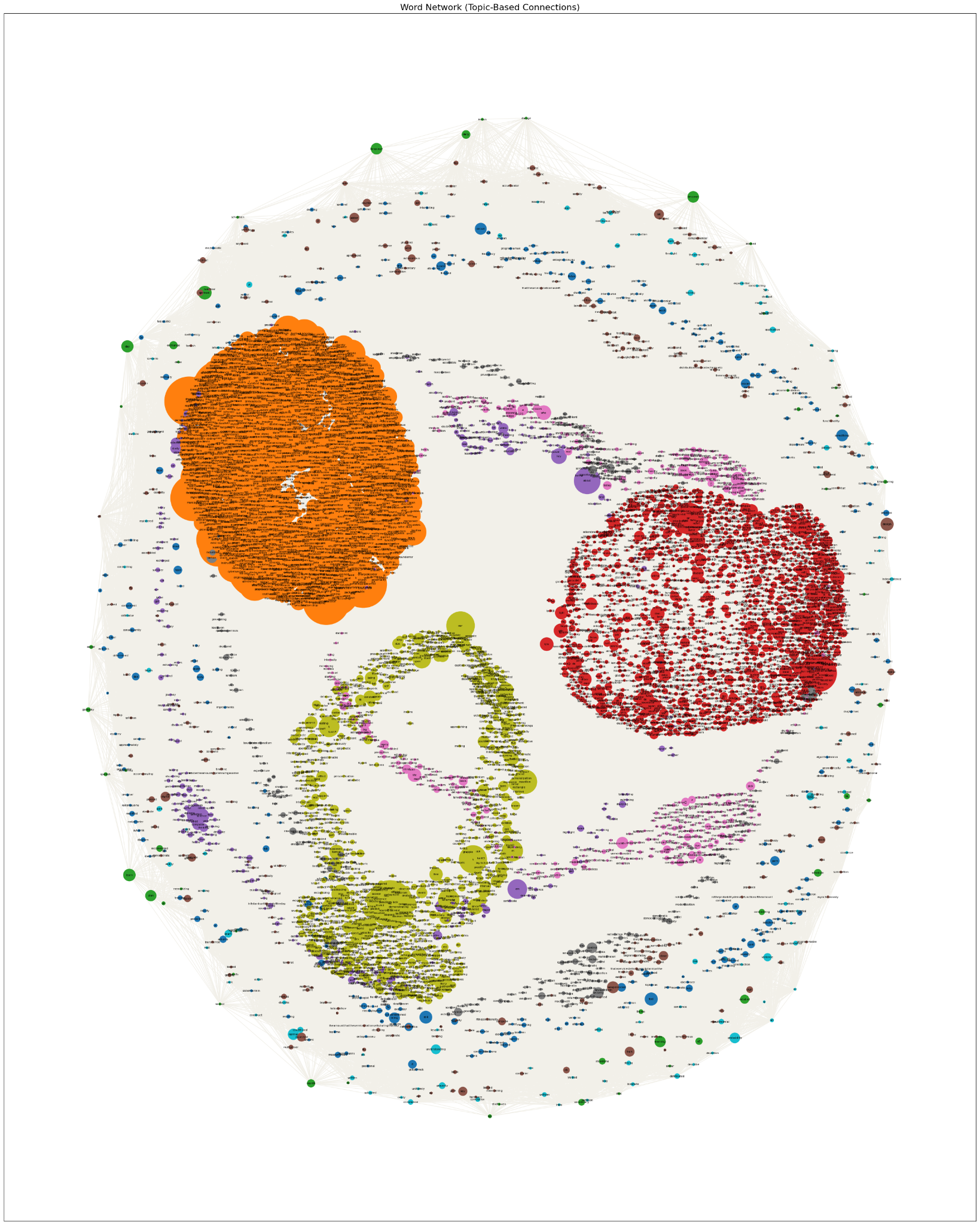
**Fig.6**

Finally, I labeled each word depending on their frequency of occurring in a category. For example,

“cout” is mostly used in the messages that were labeled as “Computer Science and Programming”. After labeling each word with the categories, I created a graph which is a mixture of network connection and bubble graph representation. In the given map, each node represents a word, they are colored depending on their categories, and they are sized according to their frequency.

It displays the distinction and correlations/intertwinement between the topics, as you can see the nodes with orange color, Computer Science and Programming, cover a bigger area. “Electronic Circuits” and “Digital System Design” create an outer ring together. Turkish words are cumulated together with red color under the “Literature” category because of my “Turkish Language and Literature” course.

You can find the final product below:

****

**Fig.7**

**( For better quality:** [**Network with topics.png**](https://drive.google.com/file/d/1fgMhMt2-UKghP34Ypmk6Yjh8NsBYbABb/view?usp=sharing) **)**