DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

WORDFLOW: A MOBILE WORD GAME BASED ON AI

by

Ayberk DİKÇİNAR

Gaye SÜNER

Zafer YALÇIN

Advisor

Asst.Prof.Dr. Feriştah ÖRÜCÜ DALKILIÇ

CHAPTER ONE

INTRODUCTION

Language is present in every aspect of our lives as a tool that mediates us in our interpersonal relationships and regulates our social ties, and it is an integral part of the human self.

Language is made up of an extremely complex combination of a wide variety of sounds. Therefore, language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text or voice data. In order to solve or reproduce this complexity and canonical structure, natural language processing has emerged.

When we analyze the structure of language and start producing, people's expectations for machines increase. The effect of computers to understand, process, interpret and even produce sentences in the language we speak is an indisputable fact. Natural language processing is a discipline in which linguistics and computational sciences such as artificial intelligence and machine learning are used jointly. Technologies such as the chatbot we are talking about on the bank site today, the commands we give to the assistant on our phone, the translations we make in google translate, the prediction of the next word by our phone while writing a message are the result of natural language processing. In addition, thanks to text mining, which has been very popular lately, we can process and make sense of thoughts that accumulate in masses on the Internet. As can be seen from the examples, natural language processing is used in many environments and fields and is one of the promising studies of the future.

1.1 Background Information

Natural language processing technology started to grow its roots in the 16th century. NLP technology was theorized by René Descartes and Gottfried Wilhelm Leibniz, who proposed codes that could relate words between languages. Due to the insufficient development of technology at that time, the theory was not put into practice and examples emerged after about three centuries.

The Georgetown-IBM experiment (1954), which can be considered as one of the first examples of NLP, has been one of the important breakthroughs in this field. As part of this experiment, more than 60 Russian sentences were automatically translated by computers. Nearly 15 years later, artificial intelligence theorist Roger Schank developed the conceptual dependency theory (1969) to understand natural language. (Conceptual dependency theory is a model of natural language understanding used in artificial intelligence systems) The purpose of this study was to enable computers to read meaning independently of the word actually typed. This approach taught machines that no matter how sentences were entered into computer systems, the way they were written didn't matter as long as their meanings were the same. Thanks to Schank's theory, computers are now being trained to understand that the sentences ‘I gave the man a book’, ‘The man got a book from me’ and ‘The book was given to man by me’ mean the same thing. In the 1970s, NLP researcher William A. Woods introduced the Augmented Transition Network (ATN). Representing natural language input, ATN gave computers the ability to analyze sentence structure regardless of complexity.

Today, with the development of technology, more applications have been developed for NLP than ever before. We now see natural language processing technology has usage in many areas of our daily lives. Natural language processing continues to increase its functionality by constantly giving more accurate and consistent results in direct proportion to the development of artificial intelligence technology.

Figure 1.1. shows the most basic steps of NLP.

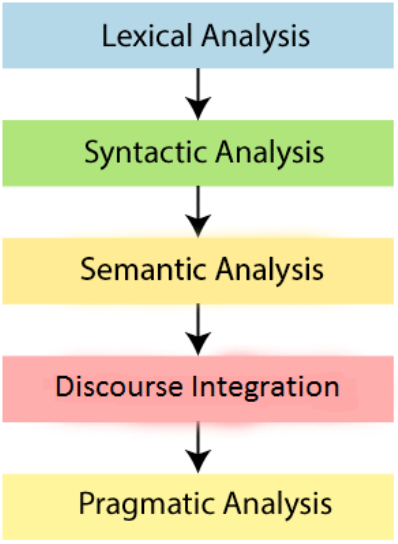

Figure 1.1.

Lexical Analysis: It involves identifying and analyzing the structure of words. It means the collection of words and expressions in a language. Lexical analysis breaks down entire chunks of a text into paragraphs, sentences, and words.

Syntactic Analysis: It includes the analysis of the words in a grammatical sentence and their arrangement to show the relationship between the words.

Semantic Analysis: The full meaning or dictionary meaning of the words collected from the text or dictionary is created on the system. And its text is checked for expressiveness. It is done by matching syntactic structures and objects.

Discourse Integration: It is the stage of integration of meanings between sentences.

Pragmatic Analysis: It involves deriving the linguistic aspects of the generated synthesis that require real-world knowledge.

1.2 Problem Definition

NLP is a very popular and quite compelling software topic nowadays. The most important factor that makes NLP difficult is that people use the language by establishing connections for the context in which the word is used and the purpose of being said, not just the dictionary meanings of the words while conveying a message. Uncertainty, variability, and imprecise semantic differences make NLP difficult for machines to implement. With the increasing number of corpus, research has become more effective and attractive. However, the lack of practice studies and the fact that most of these studies are based on English are also obvious. The inadequacy of Turkish studies and the inability to make significant progress are also a problem today. It is obvious that Turkish is more regular and complex than most languages due to its structure which means that we have a long way to go.

Wordnet is a dictionary database with which words are conceptually associated. A Turkish word has the feature of forming a new word with its suffixes. Natural language processing studies on Word vector space creation for Turkish are insufficient for automating computer terms.

1.3 Motivation / Related Works

We can broadly divide text classification approaches into two groups:

- rule based approaches,
- statistics based approaches (Mihalcea, and Radev, 2011).

We can see the comparison of different approaches on the same issues in Table 1.1.

	Lexical Analysis	Syntactic Analysis	Semantic Analysis	Pragmatic Analysis
Rule-based systems	Use linguistic rules and patterns.	Use thesaurus and lists. Such as WordNet	look for emotional words like "love" or "hate"	Use a very long list of conditions
Staticial NLP	Use lots of labeling of the text to predict new data	Use lots of labeling of the text to predict new data	Use Machine Learning and provide weights to linguistic terms	Use Machine Learning and provide weights to linguistic terms

Table 1.1.

Industrial applications have been created for NLP to find solutions to some mentioned problems. However, some of these studies are not sufficient for agglutinative languages such as Turkish. We hope that our work will be useful for such languages.

1.3.1 Sentiment Analysis

These applications are often used by companies to measure user satisfaction.

1.3.2 Topic Modeling, Text Classification, Document Categorization

It is a method used to find the topic that best describes the texts in a document stack. In order to be successful in this matter, supervised or unsupervised methods are used. They are indispensable tools of quantitative statistical studies. Email classifications, e-commerce product classification examples of these can be given.

1.3.3 Text Matching

Resume shortlisting and job description matching are the studies about this subject.

1.3.4 Word2Vec, GloVe

They are algorithms for generating word vectors. They are the basis of the latest models in natural language processing, including language models such as BERT (Devlin, 2018). There are many officially reported direct applications of the word vector methods. These two tools are used in search engines and recommendation mechanisms.

1.3.5 Text Generation, Question Answering, Chat bot

Text generation is the task of creating text with the aim of appearing incomprehensible to human-written text. In 2020 a language model GPT-3 (Open AI, 2020), which demonstrated good performance on language generation tasks.

1.4 Goal / Contribution

1.4.1 Goal

It is aimed to develop a ***Turkish Word Vector*** developed by an AI trained using *DP, ML* etc. techniques. It is aimed to develop an interactive word game by using this developed *Turkish Word Vector* and *English Word Vectors* that are currently available for use. Out of the N number of words that appear on the screen during the game, n of them are closer to each other. It is aimed to find these n words with the least possible hint. For this, using Word Vector, the semantic closeness of the words to each other will be checked and a hint(word) will be given accordingly. Hints will be provided by our *pre-trained* AI.

In conclusion, the main goals are:

- A Turkish Word Vector Space will be created
- An AI will be trained with the Turkish Word Vector Space and English Word Vector Space (Gensim) (Řehůřek, 2011).
- Game interface and mechanics should be developed for mobile platforms.
- AI will contribute to the game without any delay.

1.4.2 Contribution

When the project is completed, a comprehensive Word Vector created for Turkish will be added to the NLP literature. Moreover, if Word Vector is desired to be created later, especially for *agglutinative languages* such as Turkish, this document will serve as a source through both the studies done and the sources brought together.

1.5 Project Scope

This study revealed an artificial intelligence supported word game to be played on mobile platforms. The word game was supported by an AI that gives hints to find the related words of a certain number of words that display. User will try to find all the words as soon as possible with the hints given by the AI. In order for the AI to give a hint to the user during the game, it must be trained with a Word Vector in English and Turkish languages. Existing English Word Vectors provide adequate support for this study WordFlow. Unfortunately, there is no such support for Turkish, so a wide comprehensive Word Vector space should be revealed through this study.

1.6 Methodology / Tools / Libraries

Text representation is the process of converting texts into machine language, for instance numbers, that computers can understand. In the study we aim to do, Since it is necessary to determine the distances of words in terms of meaning, we should use text representation methods. Word representations attempt to map a word to a vector via the vocabulary.

Vocabulary : non-repeated list of all words in corpus.

We can examine text representation methods in general under two main headings:

- Frequency-based Representation
- Prediction-based Representation

Frequency-based Representation

Frequency-based Representation is a method of representing texts with sparse vectors as they are frequency-based (Kulkarni and Shivananda, 2019).

We can examine frequency-based representation methods under three subheadings:

- Bag-of-words model and count vectors
- TF*IDF vectorization
- Co-occurrence matrix (Manning, 2000)

Prediction-based Representation

Frequency-based Representation is a method of representing texts with sparse vectors as they are frequency-based.

We can examine frequency-based representation methods in terms of already existing products under the sub-title.

- Word2Vec (Continuous Bag of Words and skip-grams)
- GloVe (Global Vectors for word representation)

Since frequency-based representations are deterministic methods, they may be insufficient in word representations. That's why prediction-based models are in demand these days. Basically, the purpose of these models is to group words with similar meanings or similar analogies in a word space.

During this Project, the programs will be developed on the Colab platform with Python Programming Language.

Turkish language, unlike English, is an agglutinative language. Therefore, to find the body or body of the words in the sentence to be analyzed, different rules of thumb are valid. We can write rule-based Turkish language algorithms to perform these operations, but writing this will take more time than the finishing project.

The Java-based Zemberek library, which has been used for the Turkish language for years, continues to be developed to perform many NLP operations, especially the basic body finding algorithms of the Turkish language.

- The processes required in our project are as follows;
- Cleaning of raw data,
- Tokenization and Lemmatization transactions,
- Finding N-grams.

Python is the recommended platform to use to process the data that is then cleaned and extracted. Although Zemberek is written based on Java, it has an interface to be used with Python.

The reason we chose the Python language is that many statistical tools and functions to be used for machine learning are provided from certain Python libraries.

The most commonly used libraries in natural language processing are:

NLTK: Natural Language toolkit and commonly called the mother of all NLP libraries.

SpaCy: Spacy is recently a trending library, as it comes with added flavors of a deep learning framework.

TextBlob: It's a library that combines both NLTK and Pattern (Mikolov, 2013).

Word2vec: It is a word embedding algorithm developed by Google. It will use all the words of the whole corpus and predict the nearby words.

Gensim: Gensim is designed to process raw data using Word2Vec.

The goal of this project is to produce a tool like Gensim by using the Word2Vec algorithm for Turkish.

CHAPTER TWO

LITERATURE REVIEW

2.1 Efficient Estimation of Word Representations in Vector Space

Many current NLP systems and techniques treat words as atomic units. It has been observed that the complexity and performance of the system increase in parallel as the amount of data we have grows. The most popular model used for statistical language modeling is the n-gram model.

With the advancement of machine learning techniques in recent years, it has become possible to train more complex models on much larger datasets, and they typically outperform simple models. Probably the most successful concept is to use distributed representations of words. For example, neural network-based language models significantly outperformed N-gram models (Mikolov, Chen, Corrado & Dean, 2013).

While it is easy to show that the word Turkey resembles the United States and perhaps some other countries, it is much more difficult to subject these vectors to a more complex similarity task. We follow the previous observation that there can be many different kinds of similarity between words, for example, the 'biggest' word has the same meaning as 'big'. Another example of a relationship would be the 'biggest' and 'largest' (Mikolov, Yih, Zweig, 2013).

"What is the word that is similar to small in the same sense as biggest is similar to big?". These questions can be answered by performing simple algebraic operations with the vector representation of words (Mikolov, Chen, Corrado & Dean, 2013).. Mikolov et al. developed an algorithm for those simple algebraic operations. For the following algorithm 2.1.

$$vectorX = vector("fastest") - vector("fast) + vector(slow)$$

Formula 2.1

When word vectors are well trained, it is possible to find the correct answer using this method. When we train high-dimensional word vectors on a large amount of data, the resulting vectors can be used to understand semantic relationships more effectively between words, such as a player and the team he/she belongs to or a movie and its actors. Word vectors with such semantic relationships can be used to enhance many existing NLP applications, such as machine translation, information retrieval and question answering systems.

2.2 Some Methods in Related Works

2.2.1 Traditional Models

Although n-gram, TF-IDF and bag of words methods, which are traditional vectorization approaches used to classify texts, are powerful methods, they are weak in finding the semantic distances of texts (Ciya, Shamim and Paul, 2001).

2.2.2 Statical Methods

If we want to examine more modern vectorization approaches, unsupervised artificial neural network-based statistical approaches have given very successful results in semantic language modeling (Bengio and Ducharme, 2003).

2.2.3 Word Embedding Models

Mikolov et al. (2013) proposed two simple and powerful neural network architectures, called the Skip-Gram and CBOW models, to train word representations. These methods, using a neural network with a hidden layer, proposed to keep words with their context. These two methods are the most powerful methods that are used for word representations these days.

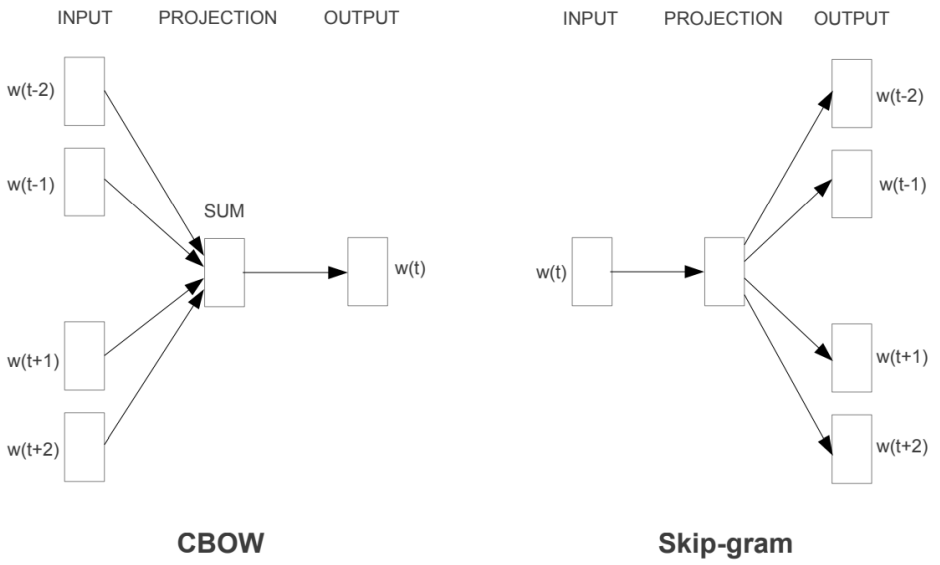

Figure 2.1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word (Mikolov, T., Chen, K., Corrado, G., & Dean, J. 2013).

2.2.4 Skip-Gram Method

The first method, skip-gram, works with an algorithm of trying to guess its neighbors in the corpus by looking at the word, while CBOW works with an algorithm that predicts the target word by looking at the neighboring words in the corpus (Mikalov, 2013).

The skip-gram algorithm tries to predict the words it is related to by looking at the equidistant before and after of the given word. Common or stop words such as the are ignored when estimating.

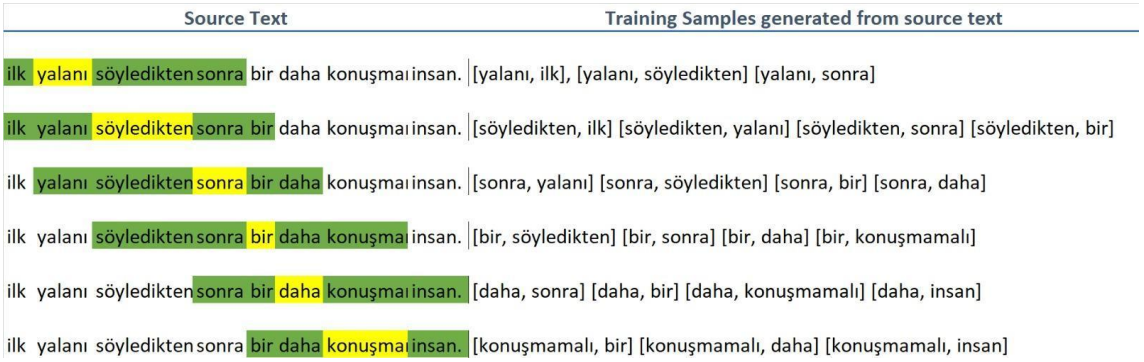

Figure 2.2: A simple Skip-gram example based in 2 width for Turkish

Skip-gram's progression stages can be reduced to 7 basic steps.

1. Words are converted to vectors with one hot encoding. The size of the resulting vector is [1, |v|].

2. The word w(t) is passed to the hidden layer from |v| neurons.

3. In this stage hidden layer vector W[|v|, N] and our input vector w(t) performs a dot product. With this operation, the W[|v|, N] output of the (t)th line results in output(H[1, N]).

4. Since no activision functions are used in the hidden layer, H[1, k] is sent directly to the output layer.

5. In the output layer, dot multiplication is applied to H[1, N] and W'[N, |v|]. This way vector U is returned.

6. At this stage, the probability of each vector to be used in the softmax function is calculated. It will output vector U in hot encoding type from each iteration.

7. If the prediction of the word with the highest probability outcome is incorrect, backpropagation is performed again and the weights of the W and W' vectors are modified.

Probability function for Skip-gram

For the following Formula 2.2;
w(c,j0) is the jth word predicted on the cth context position;
w(O,c) is the actual word present on the cth context position;
w(I) is the only input word;
and u(c,j) is the jth value in the U vector when predicting the word for cth context position.

$$p(\omega_{c,j} = \omega_{O,c}|\omega_I) = \frac{expu_{c,j}}{\Sigma_{j'=1}^{V} expu_j}$$

Formula 2.2: Softmax Probability

Loss function for Skip-gram

For the following Formula 2.3
Maximize the probability of predicting w(c,j) on the cth context position can represent the loss function L.

$$L = log\mathbb{p}(\omega_{c,1}, \omega_{c,2},..., \omega_{c,C}|\omega_o) = -log\prod_{c=1}^{C}\mathbb{p}(\omega_{c,i}|\omega_o)$$

$$= -log\prod_{c=1}^{C}\frac{exp(u_{c,j^*})}{\Sigma_{j=1}^{V}exp(u_{c,j})} = -\sum_{c=1}^{C}u_{c,j} + \sum_{c=1}^{C}log\sum_{j=1}^{V}exp(u_{c,j})$$

Formula 2.3: Loss Function

Pennington et al. (2014) used global co-occurrences to train word vectors in GloVe models.

2.3 Word Embedding Works for Turkish

The methods suggested above for word representations have also been applied to the Turkish language. Şen and Erdogan (2014) tried to successfully use the skip-gram algorithm for Turkish that was already successful in English and published a paper to encourage the application of deep learning studies on Turkish language.

The artificial neural network-based skip-gram algorithm proposed by Mikalov, which is a text representation method prepared for the English language, has achieved greater success compared to other format-independent algorithms, although it is not specialized for Turkish (Yildiz et al, 2016).

2.4 Required Technologies for Transmissions

2.4.1 Web Service

Web service is a standard in the distribution part of data and processes among multiple applications and services that are available on the Internet. Applications on the internet are generally made using different programming languages and should be able to run well on operating systems, even on different devices (Misbah, 2016). Web service utilization is to solve problems in integration between processes and data on current exchange of separate information, as well as data security level can be handled properly. The architecture that is generally used in building a web service is REST and SOAP. REST API is defined as an architectural tool designed on web services that works over the HTTP protocol that provides communication between client and server. It enables the application to communicate by carrying data between the client and server (Ma, 2015). REST web service works by identifying the URI and then modifying it with GET, POST, PUT or DELETE commands which can be represented on some language programming such as XML or JSON. Services using REST architecture are called RESTful services.

2.4.2 Rest Api In Python

We should design a rest api for communication between our backend and frontend. When it comes to developing web applications with Python, we will have to choose between two popular frameworks: Django or Flask. Django is more mature and a little bit more popular than Flask. However, Flask has its strengths too. From the beginning, Flask was built to be scalable and simple to get started with. Applications built with Flask are clearly lighter when compared to Django counterparts. As such, Python developers usually refer to Flask as a microframework.

CHAPTER THREE

REQUIREMENTS/REQUIREMENT ENGINEERING

A skip-gram algorithm will be used to create Word Vector Space, a Python environment is required to run this algorithm. Environments such as Colab, Spyder etc. will be used for execution.

A user interface will be designed using Flutter. From this interface, a request will be sent to the server, a response will be returned to the received request in a specific file format using Word Vector Space.

3.1 Functional Requirements

- Creating Turkish Word Vector Space with skip-gram algorithm.
- New game request handling.
- Preparing word tables on the server side.
- Sending the word table back to the client.
- Display user interface.
- Sending hints and a number of related words to the client.
- Clients should choose the game mode (Multiplayer or single player).
- If clients choose the multiplayer mode, they should share the game link to other desired clients.
- Clients should specify his/her name before starting a new game or creating a room.
- Clients should find the related words with the given word in a certain time.
- Clients should arrange the timer settings.
- Clients should be able to select the language they want before the game starts.

3.2 Non-Functional Requirements

- The game will be accessible for all user segments. The game server will provide 7/24 accessibility by keeping the server on a cloud provider.
- In order to establish a fast, reliable and healthy communication between the front end and back end server, Web service will be used.
- The architecture of the game will be clearly explained. User instruction will be well designed.
- For the purpose of the elimination of some delay problems, the scaling of the necessary service will be provided.
- Clients must have an internet connection with a minimum 2 Mbps.
- The average response time must be less than 1.5 seconds. If the response time takes longer than expected, it is evaluated as a timeout and sends a request again.
- To reduce the response time some data will be cached or loaded in the background as soon as the process starts or page loads.

CHAPTER FOUR

DESIGN

4.1. Architectural View

An interface is presented to the client with the help of the front end server and the client starts a new game from that interface. The new game request is forwarded to the backend server with the help of the web service, and a word vector space model is created with help of the skip gram on the back end and sent to the front end in json format for the purpose of presenting to the client. Shown in Image 4.1.1.

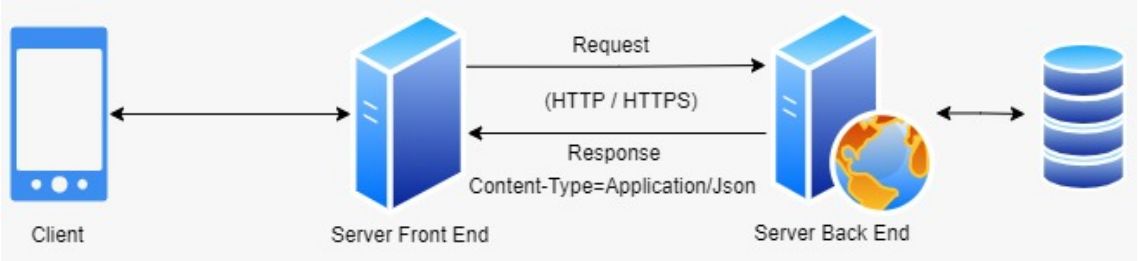

Image 4.1.1

4.2. UML Class Diagram

The relationship of the classes in the study is as shown in Figure 4.2.1. The system consists of 7 different classes that perform various operations. Communication is provided by using the necessary classes and functions from the Client and Server interfaces. Classes also assign the necessary features and functions for the system to play single or multiplayer games to the user. Classes shown and prepared may vary due to minor changes.

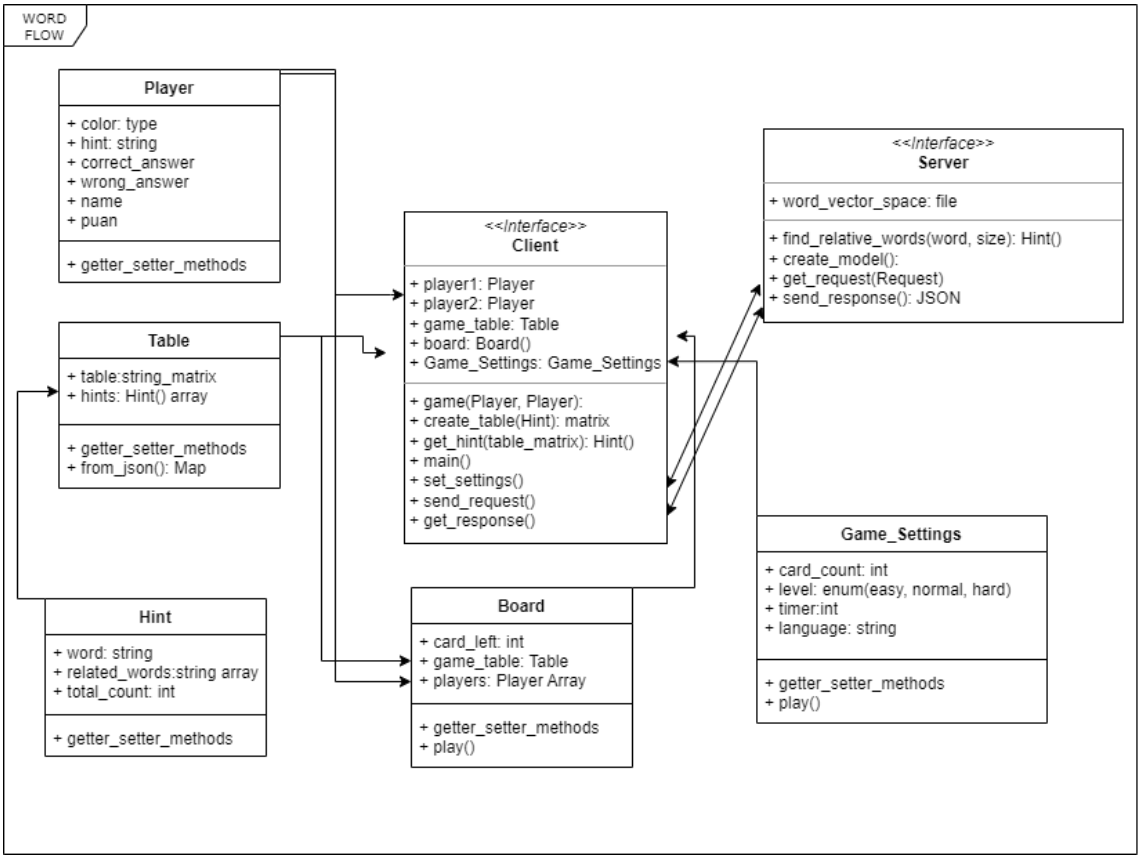

Figure 4.2.1

4.3. UI Design

When the user starts the new game, the words and clue will be displayed as shown in Figure 4.3.1. Then, depending on the hint, the right and wrong choices made by the user will be marked. As shown in Figure 4.3.3 and 4.3.4.

Figure 4.3.1 Main

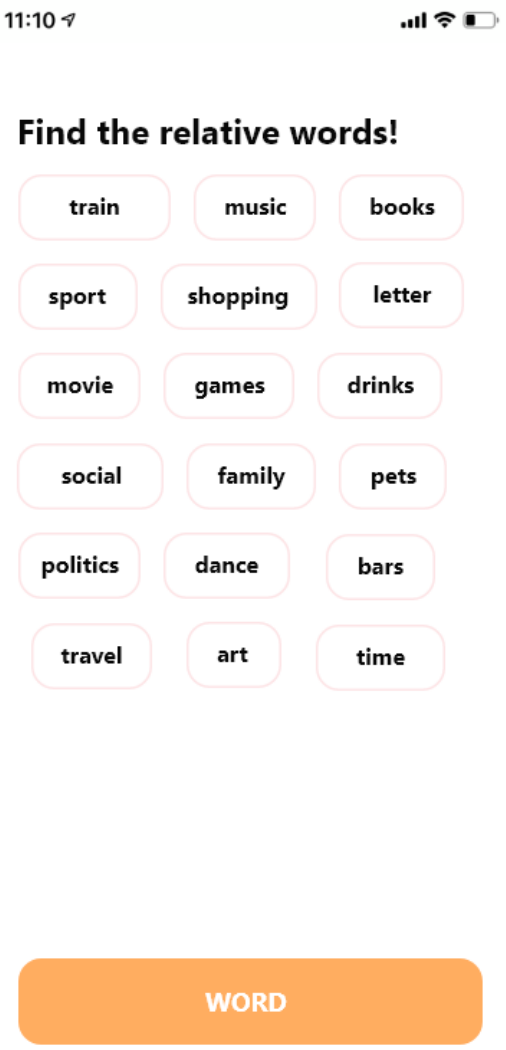

Figure 4.3.2 Game

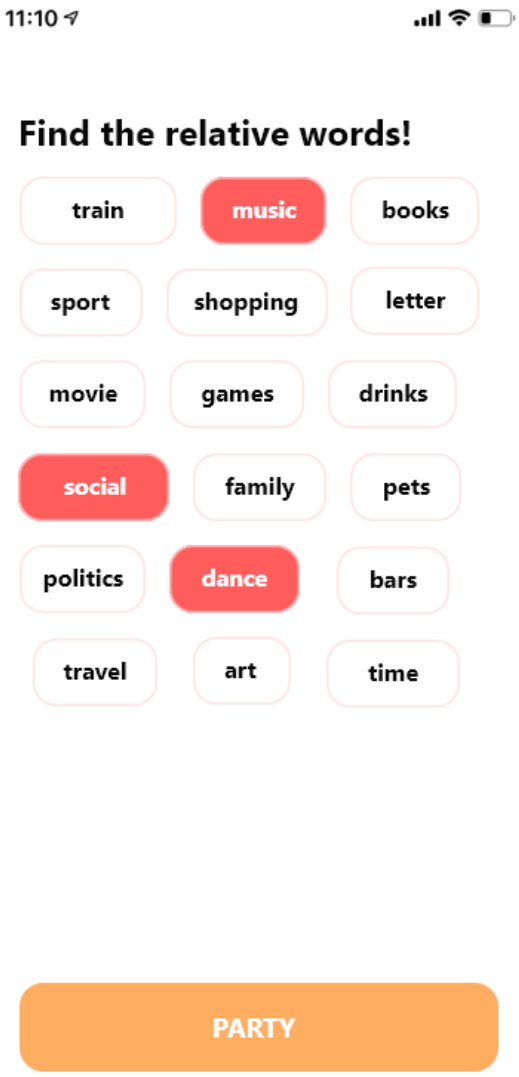

Figure 4.3.3 Game Correct Choice

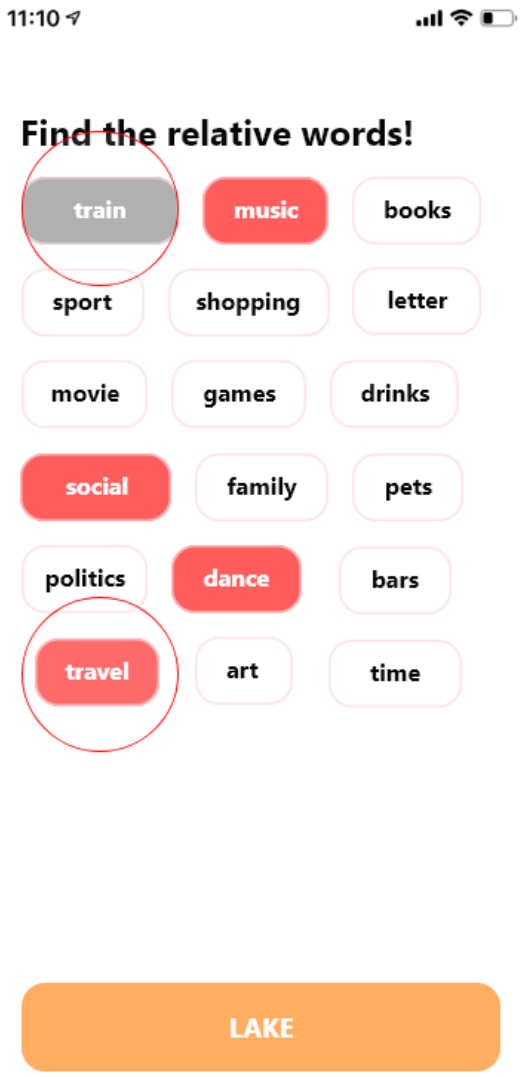

Figure 4.3.4 Game Incorrect Choice

4.4. Use Cases

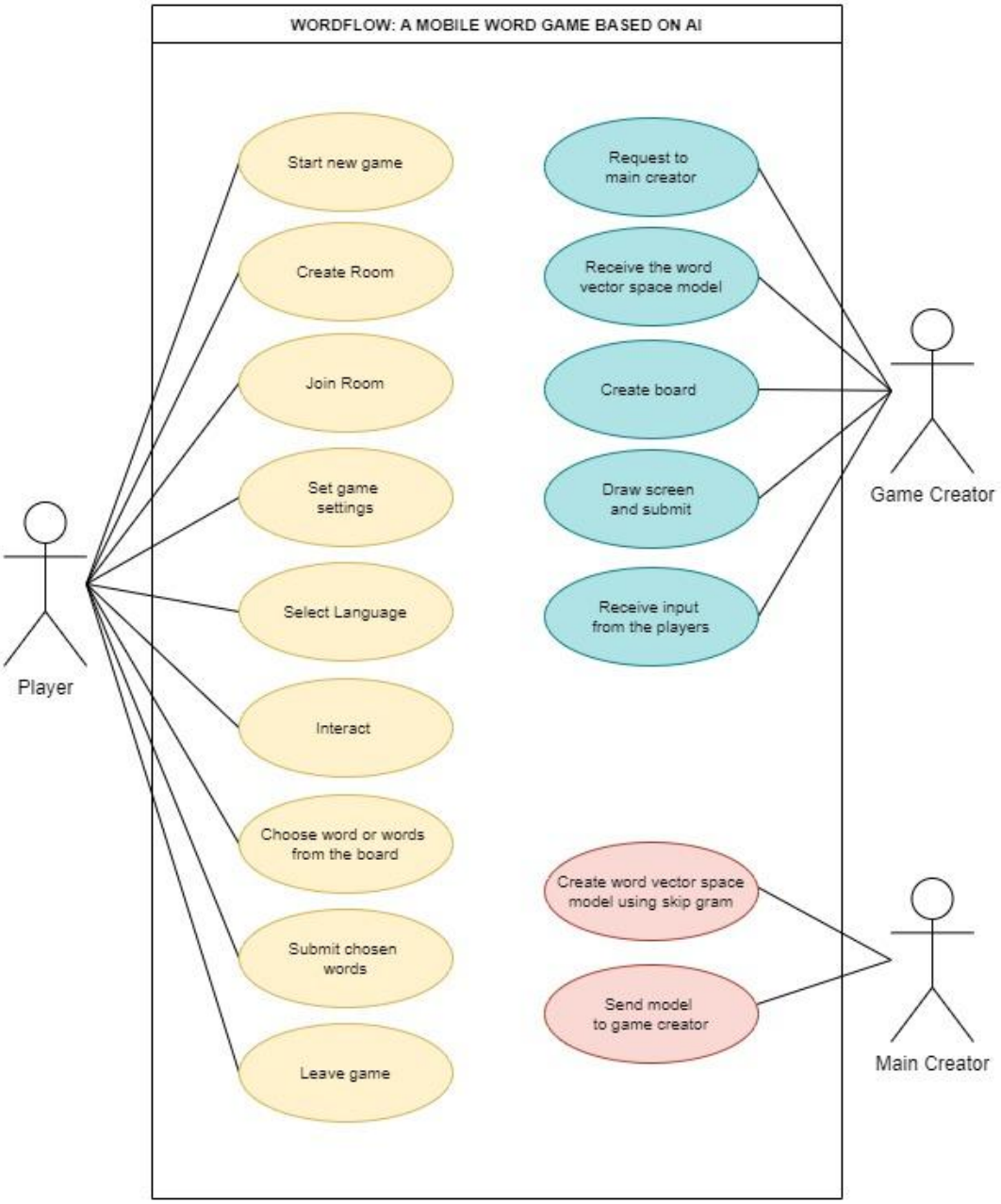

Figure 4.4.1 Use Case Diagram

Player use cases

- **Name:** Start new game.
- **Summary:** The player connects the mobile application and creates a new game.
- **Actors:** The player.
- **Preconditions:** None.

- **Basic sequence:**
    - **Step 1:** The player connects the mobile application.
    - **Step 2:** The player chooses to start a new game option.
    - **Step 3:** The player specifies his/her name.
    - **Step 4:** The player sets the game setting as he/she wishes.
- **Exceptions:**
    - **Step 2:** The game might not be started because of any delay in the system or problems about creators.
- **Post conditions:** The player has now a ready to start game.

- **Name:** Create room.
- **Summary:** The player creates his/her game room and shares the link of the created room with the other players.
- **Actors:** The owner player of the game room.
- **Preconditions:** The player has selected multiplayer mode.
- **Basic sequence:**
    - **Step 1:** The player starts the mobile application.
    - **Step 2:** The player chooses multiplayer mode.
    - **Step 3:** The player sets the name.
    - **Step 4:** The player sets the game settings.
    - **Step 5:** The game creator creates a room link.
    - **Step 6:** The player shares the game link with his/her friends.
- **Exceptions:** None.
- **Post conditions:** The player is now waiting for the other players.

- **Name:** Join room.
- **Summary:** The player starts the game and joins the room.

- **Actors:** The player
- **Preconditions:** The player has started the mobile application.
- **Basic sequence:**
    - **Step 1:** The player starts the game mobile application.
    - **Step 2:** The player chooses multiplayer mode.
    - **Step 3:** The player enters the room link he/she wants to join.
    - **Step 4:** The player sets the name.
    - **Step 5:** The player joins the room.
- **Exceptions:**
    - **Step 3:** The room link might not be true.
- **Post conditions:** The player is now in the room.

- **Name:** Leave game.
- **Summary:** The player stops playing the game and exits the game.
- **Actors:** The player.
- **Preconditions:** The player must be in the game.
- **Basic sequence:**
    - **Step 1:** The player selects Exit Game command from the client.
    - **Step 2:** The player is out of the game.
- **Exceptions:** None.
- **Post conditions:** The player is off game.

- **Name:** Set game settings.
- **Summary:** The player sets the game settings such as timer and words count.
- **Actors:** The player.
- **Preconditions:** The player must be the owner of the game or room.

- **Basic sequence:**
    - **Step 1:** The player selects game settings from the interface.
    - **Step 2:** The player arranges settings as he/she wishes.
- **Exceptions:** None.
- **Post conditions:** The player has now changed the settings of the game.

- **Name:** Select language.
- **Summary:** The player sets the language settings.
- **Actors:** The player.
- **Preconditions:** The player must start the mobile application.
- **Basic sequence:**
    - **Step 1:** The player selects language settings from the interface.
    - **Step 2:** The player changes language.
- **Exceptions:** None.
- **Post conditions:** The player has now changed the language.

- **Name:** Interact.
- **Summary:** The player interacts with the game board.
- **Actors:** The player.
- **Preconditions:** The player must be in the game.
- **Basic sequence:** None.
- **Exceptions:** None.
- **Post conditions:** None.

- **Name:** Choose words from the board.
- **Summary:** The player chooses words.

- **Actors:** The player.
- **Preconditions:** The player must be in the game and clues must be given by the game creator.
- **Basic sequence:**
    - **Step 1:** The player selects words that are relative with the given clue from the board.
- **Exceptions:** None.
- **Post conditions:** The game creator waits for the player to submit chosen words.

- **Name:** Submit chosen words.
- **Summary:** The player submits the word or words that he/she chose.
- **Actors:** The player.
- **Preconditions:** The player must be in the game and words must be chosen by the player.
- **Basic sequence:**
    - **Step 1:** The player submits words that are chosen.
    - **Step 2:** The game creator checks if the chosen words are true or not.
    - **Step 3:** The player gets a message.
- **Exceptions:** None.
- **Post conditions:** The game state changes.

Game Creator use cases

- **Name:** Request to main creator.
- **Summary:** The game creator requests a new model when it receives a new game request.
- **Actors:** The game creator.

- **Preconditions:** The player must be in the game and words must be chosen by the player.
- **Basic sequence:**
    - **Step 1:** The player wants to start a new game and sends a request.
    - **Step 2:** The game creator gets the request and it requests the main creator to get a model for the game.
- **Exceptions:** None.
- **Post conditions:** The new game request by the player.

- **Name:** Receive the word vector space model.
- **Summary:** The game creator receives the model from the main creator.
- **Actors:** The game creator.
- **Preconditions:** The game creator must send a request to the main creator.
- **Basic sequence:**
    - **Step 1:** The game creator receives the model.
- **Exceptions:** None.
- **Post conditions:** The game creator can now create the board.

- **Name:** Create board.
- **Summary:** The game creator fills the game board in the background.
- **Actors:** The game creator.
- **Preconditions:** The main creator must send the model to the game creator.
- **Basic sequence:** None.
- **Exceptions:** None.
- **Post conditions:** None.

- **Name:** Draw screen and submit.

- **Summary:** The game creator draws the screen that is seen by the players.
- **Actors:** The game creator.
- **Preconditions:** The creation of the board must be completed.
- **Basic sequence:**
    - **Step 1:** The game creator draws the screen.
    - **Step 2:** When step 1 is finished. It submits for the player.
- **Exceptions:** None.
- **Post conditions:** Player is now able to play the game.

- **Name:** Receive input from the players.
- **Summary:** The game creator receives the player interactions.
- **Preconditions:** The player should interact with the screen.
- **Basic sequence:**
    - **Step 1:** The game creator gets input from the player.
    - **Step 2:** The game creator evaluates the input.
    - **Step 3:** The player gets a message.
- **Exceptions:** None.
- **Post conditions:** The game status is changed.

Main Creator use cases

- **Name:** Create word vector space model using skip gram.
- **Summary:** The main creator creates the model.
- **Preconditions:** None.
- **Basic sequence:** None.
- **Exceptions:** None.

- **Post conditions:** The model has been created.

- **Name:** Send model to game creator.
- **Summary:** The main creator sends the model to the game creator.
- **Preconditions:** The main creator must create the model.
- **Basic sequence:** None.
- **Exceptions:** None.
- **Post conditions:** The model has been sent to the game creator. The game creator now is able to use the model.

4.5. Sequence Diagram

The sequence diagram of the operations that took place between the start of the game and the creation of the word table is given in figure 4.5.1. After the player presses the game start button, the create_deck request is sent to the server. Then the json generation function on the server is triggered. The word table and their clues that will be seen on the screen from now on are created in json format.

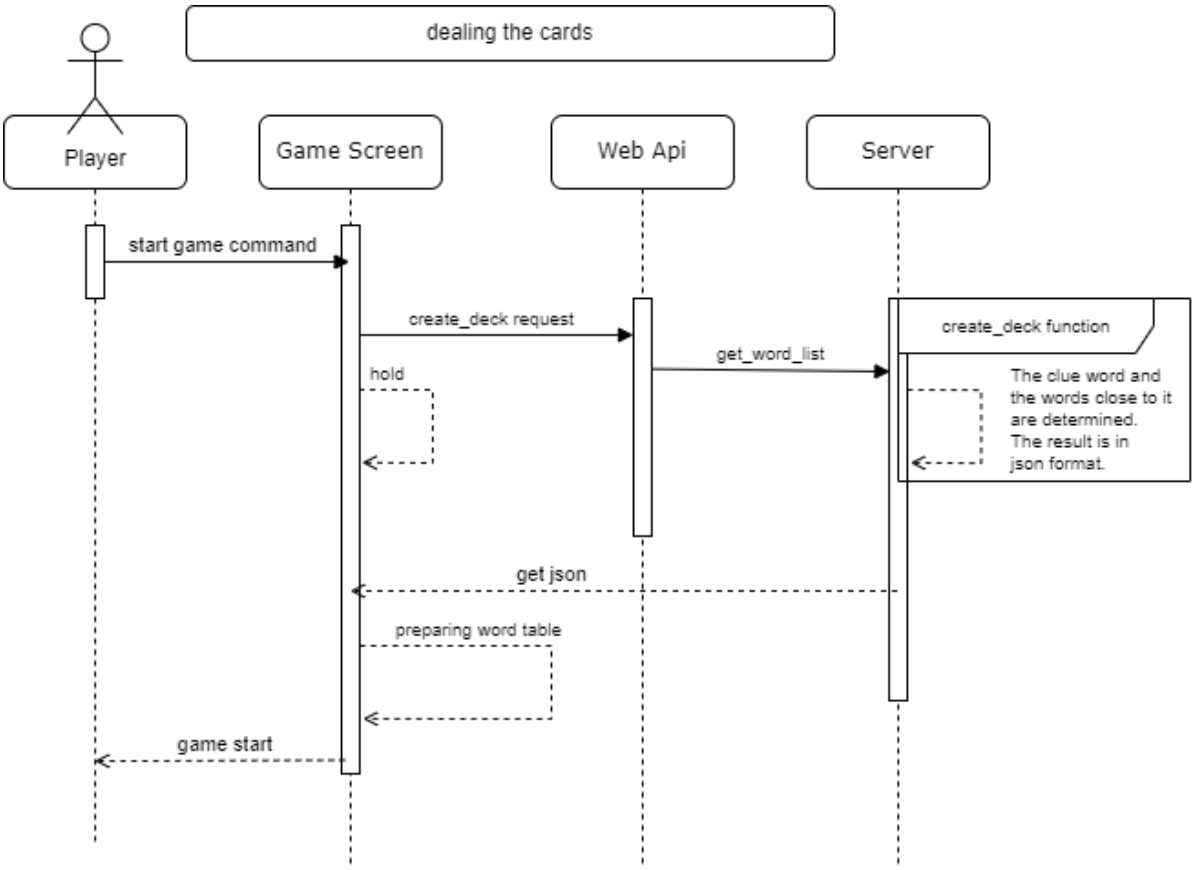

Figure 4.5.1 Dealing Card Sequence Diagram

4.6. Activity Diagram

An Activity Diagram was drawn in figure 4.6.1 in order to explain the json creation function triggered on the server in more detail. An empty json file is created when the function starts. A limited number of lists are opened for clues in the json file. Words close to the clue are also added to a list within the clue. When the words that will appear on the screen fill the number limit, the resulting json file is sent to the game screen.

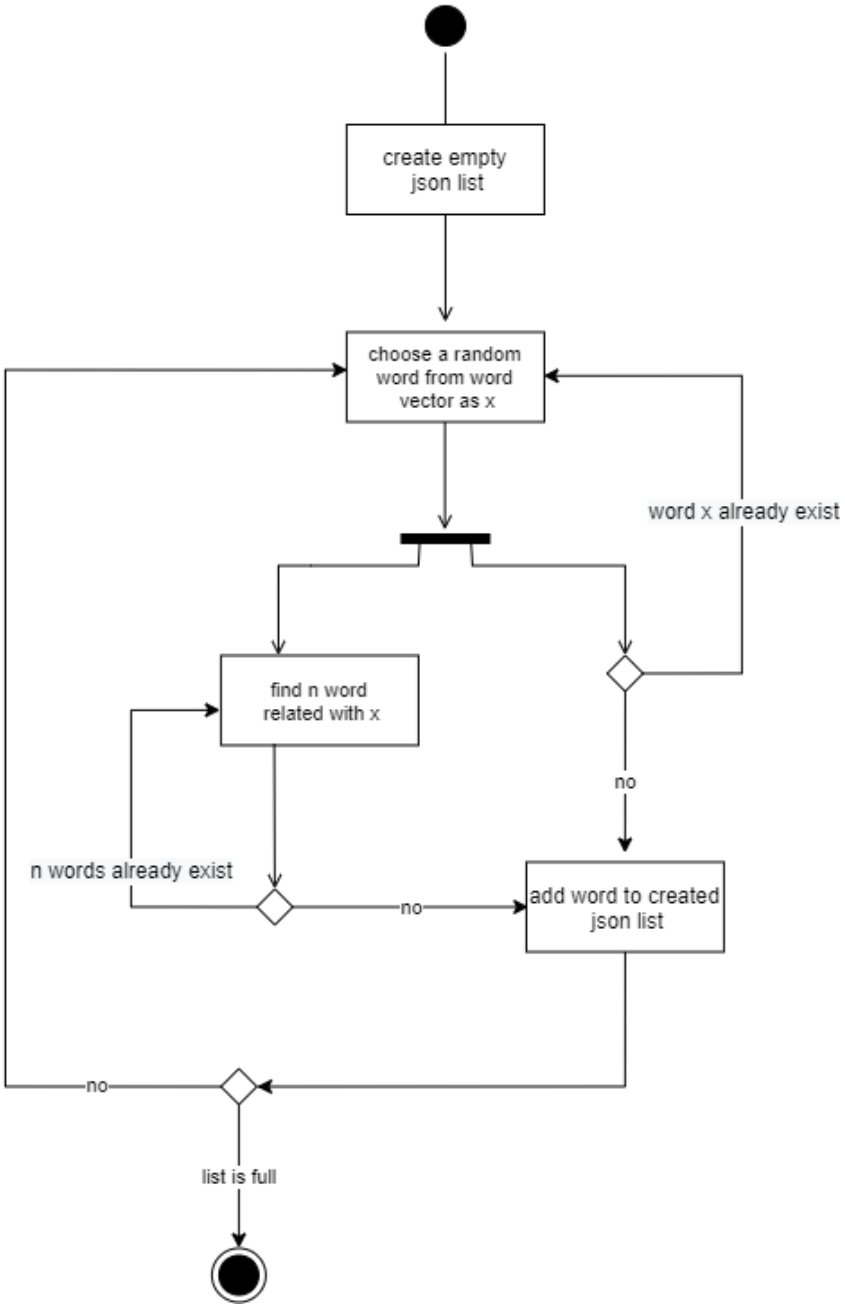

Figure 4.6.1 Creating Card Table

4.7. Deployment Diagram

The player uses his/her smartphone to open the application. After that, the application uses an api service for communication with the server. When the server receives the request from the application, the producer service requests a model for the game from the composer. The communication between the application and the server has a type of json format. Main format of the json is given below:

[{"hint":"screen","related_words":["A", "B", "C"], "total_count":"3", }, { "hint":"flower", "related_words":["D", "E"], "total_count":"2", }, { "hint":"teacher", "related_words":["F", "G", "H","I"], "total_count":"4", },]

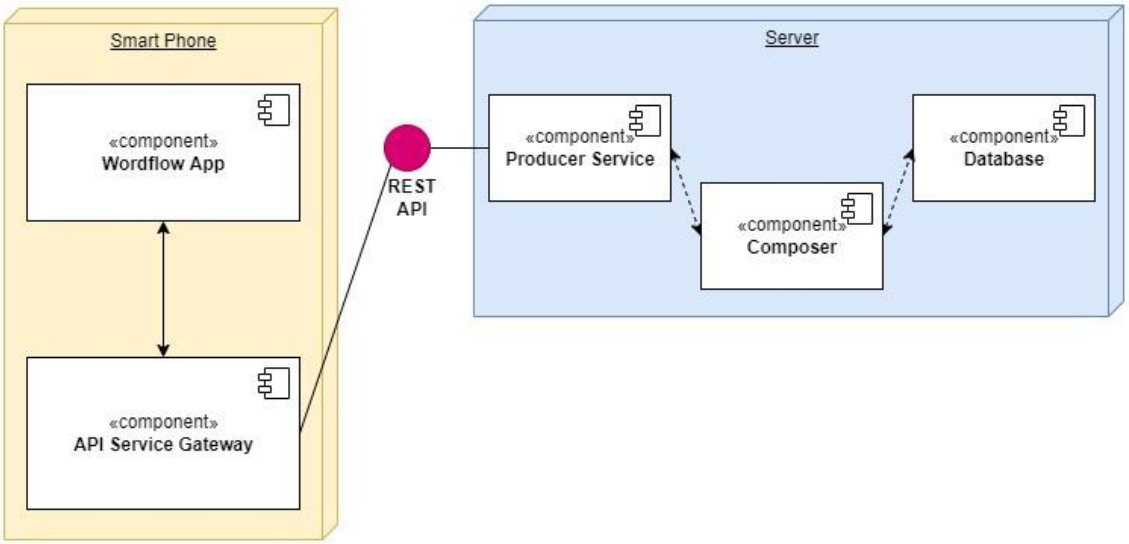

Figure 4.6.2 Deployment Diagram

REFERENCES

A. E. Anass MISBAH (2016), “Towards a Standard WSDL Implementation of Multiview Web Services,” International Conference on Multimedia Computing and Systems (ICMCS), pp. 1 - 5

C.-Y. H. Y.-Y. F. J.-Y. K. Shang-Pin Ma (2015), “Configurable RESTful Service Mashup: A Process-DataWidget Approach,” Applied Mathematics & Information Sciences An International Journal, vol. 9, no. 2L, pp. 637-644, -

Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding".

E. Yildiz, C. Tirkaz, B. Sahin, M. T. Eren, and O. Sonmez (2016), “A morphology-aware network for morphological disambiguation,”

Kulkarni, A., Shivananda, A. (2019). Natural Language Processing Recipes (xxiv. page.); India: Apress.

L. Ciya, A. Shamim, and D. Paul (2001), “Feature preparation in text categorization,” Oracle Text Selected Papers and Presentations.

Manning, C.D., Schütze, H. (2000). Foundations of Statistical Natural Language Processing (539-549 . pages.); Cambridge, Massachusetts, London, England: The MIT Press.

Mihalcea, R., Radev, D. (2011). graph-based natural language processing and information retrieval . Cambridge: Cambridge.

Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)

Řehůřek, Radim (2011). Scalability of Semantic Analysis in Natural Language Processing. my open-source *gensim* software package that accompanies this thesis

Sen, M. U. and Erdogan, H. (2014). Learning word representations for Turkish. In 2014 22nd Signal Processing and Communications Applications Conference

T. Mikolov, W.T. Yih, G. Zweig (2013). Linguistic Regularities in Continuous Space Word Representations. NAACL HLT.