Bilkent University

Department of Computer Engineering

Senior Design Project

Etymøn: A Deep-Learning Application for Etymological Clustering of Words

Analysis Report

Nashiha Ahmed, Mert İnan, Cholpon Mambetova, Utku Uçkun

Supervisor: Prof. Mehmet Koyutürk
Jury Members: Prof. Uğur Doğrusöz and Prof. Varol Akman

Analysis Report
Nov 6, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

Analysis Report

Etymøn: A Deep-Learning Application for Etymological Clustering of Words

1. Introduction

Etymøn is an analysis and tracing tool for word origins in all languages. It will be used to review current etymological language families and if possible find new connections that were not already present in current taxonomy. It will accomplish this using a deep learning approach.

In the following sections, a brief description of the system and the system requirements are discussed. In addition, Etymøn's system models are also detailed.

2. Proposed System

2.1. Overview

Etymøn will be a system that contains three main components: deep-learning, augmented-reality with object recognition, and generative dreaming.

Etymøn bases most of its functionalities on the etymological map that will be generated by the deep learning algorithm. The program will collect most of its data about words from online resources and it will cluster the words according to criterias set by the us.

The augmented reality with object recognition component is an auxiliary component. Its purpose is to enable extra functionalities such as using your mobile phone camera with augmented reality or upload a picture to Etymøn webpage and the program will automatically search the name of the object in the scene or in the image.

Generative dreaming/ Hallucination component uses word clouds to generate or imagine new words.

Etymøn system will include a palpable product in the end as an online web-application and mobile application based on this etymological map.

2.2. Definitions

Some definitions of Etymøn jargon are provided.

- The *Language Sea* is the first view that the user is greeted with. It is a zoomed out map of the most abundant words graphed together to make a sea like shape.

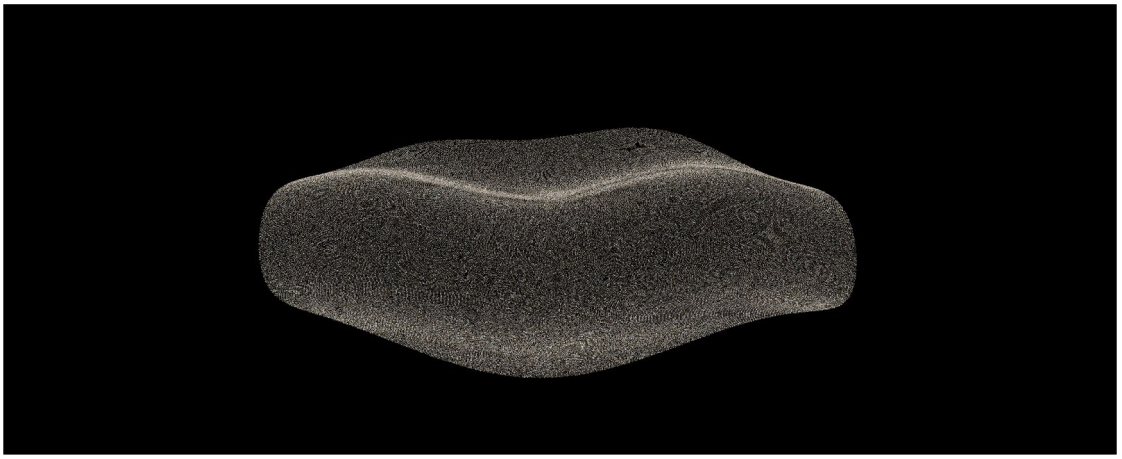

Figure 1 This figure depicts a wave-like pattern that will be like the Language Sea [1]

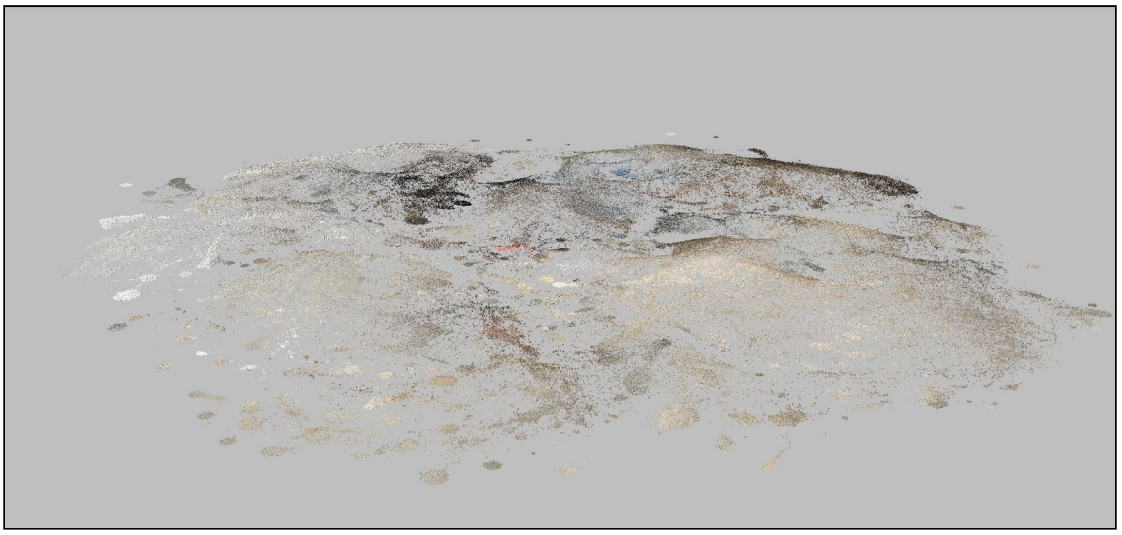

Figure 2 This figure is another clustered space that will be like the Language Sea. [1]

- The *Word Cloud* is a local graph for words clustered close to one each other.

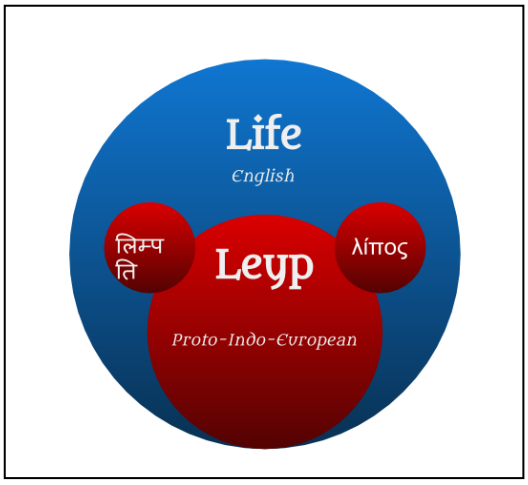

Figure 2: This figure shows a local graph for the English word, "life". Its origin is identified to be "leyp" in the Proto-Indo-European language family, and two descendent words —one in Sanskrit and one in Greek— are given next to the origin word.

2.3. Functional Requirements

- The system should provide etymological information for every word input by the user.
- Etymøn should cluster words according to root features during the deep learning phase.
- The system should show a default language map that contains all the origin languages in the initial screen.
- Etymøn should convert words into vectors in order to be used by the machine learning algorithm (word embedding).
- The system will allow user to search a specific word in the database
- Etymøn will facilitate navigation through the language sea so that the user may examine or zoom in to any word and word cloud.
- The system will allow the user to choose a language area in the language sea for the map to display.
- The system will allow the user to search for random words.

- Etymøn will hallucinate/create new words when prompted by the user.
- Etymøn must detect objects scanned by the user and find the etymologies of the words corresponding to those objects. This will be done through object recognition. The language sea will be displayed through augmented reality on the screen on which the object that is scanned is displayed.
- Etymøn should provide etymology of the word in the desired language in the augmented reality feature.
- The system should provide definitions of the words and pronunciation information in addition to their origin information.

2.4. Non-functional Requirements

Performance

Since our word database is going to be massive in size we need to implement our program in such a way that it will feel responsive.

- Etymøn should spend less than 5ms when prompted to search for words.
- Etymøn should not lag when the user navigates through the language sea. The transitions from word cloud to word cloud must be smooth.
- If a word does not exist in the Etymøn database, re-training of the algorithm should be done under 5ms. The system should inform the user that the algorithm is being re-trained.
- Response time is crucial for the software, especially when database enlarges, it should be able to give a user a desired map without taking an extensive time.

Correctness

- Etymøn should give reliable output after proper analysis. A user should not receive wrong matches.

Usability

- Etymøn should have a simple and straightforward user interface.
- Etymøn should be easily navigated by the user.

2.5. Pseudo Requirements

- Graph visualizations will be done using WebGL and Three.js.
- darkNet and YOLO will be used for object recognition in the augmented reality stage [2].
- Python, Java, HTML, Swift will be used for implementation.
- ARKit for iPhones, and ARCore for Android will be used for augmented reality components and the online component will use WebAR.
- Word2vec [3] algorithm cannot be used. A new algorithm must be created to cluster words after turning them into vectors.
- Machine learning algorithms will not run on user devices but on servers to manipulate the large dataset.

3. System Models

In this section of the report, models of the system are described in detail, these models include use-case models, dynamic models and activity models. Furthermore, class and object models are also described in this section. As the classes observed in the problem domain are trivial, they do not involve various states, hence state diagrams are not presented.

3.1. Use Case Model

In this section, use cases of Etymøn are presented in detail. Firstly, a UML use case diagram is shown to demonstrate an overview of the relations of use cases with the actors of the system. There are four different actors involved in the system. Main actor is the user, which is followed by the administrator, another human actor. Two additional actors exist which are word database and object recognition module. Etymøn will be outsourcing the object recognition module as previously described, using the YOLO library [2]. Additionally, it will be relying on the Word Database as an actor to provide the required word definitions and various stored data of the word. The use case diagram outlining all of these aspects can be found below.

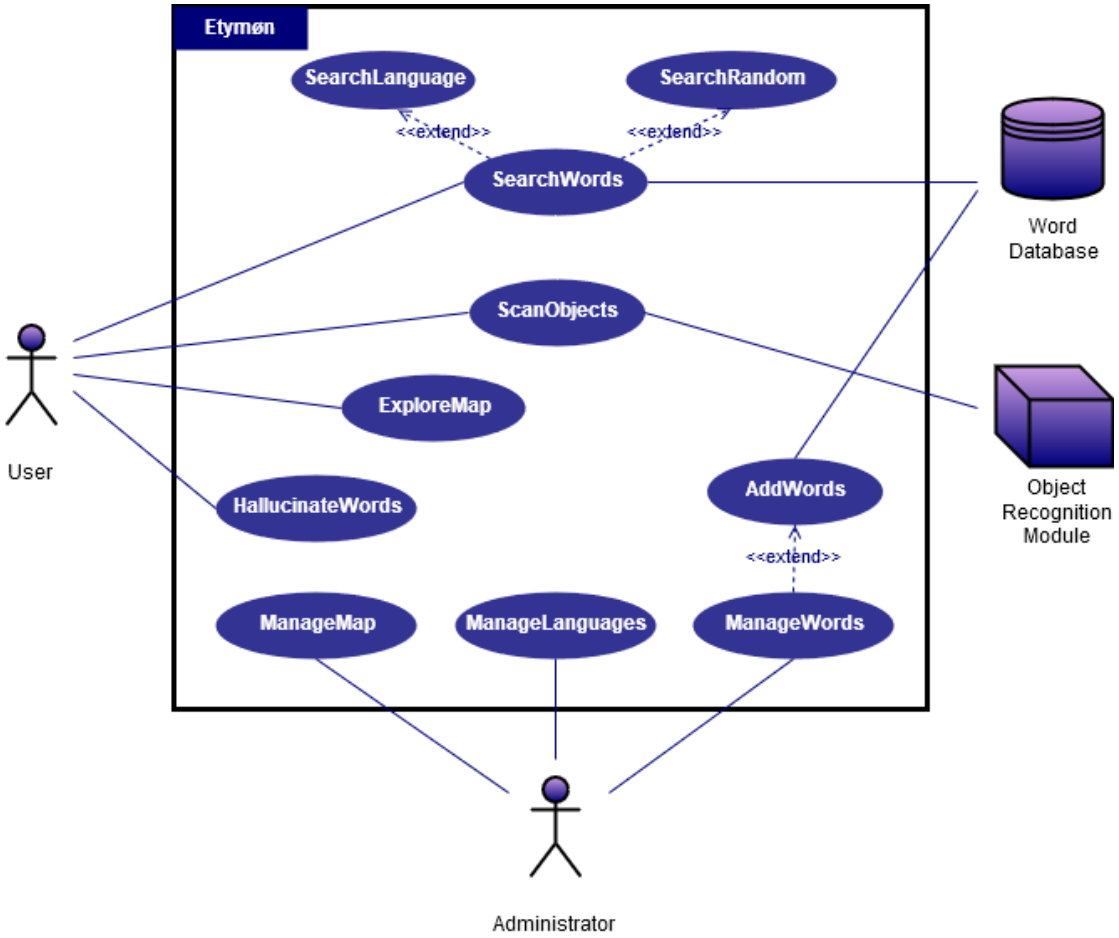

Figure 3 This figure is the UML Use-Case diagram that shows the general relationships between the actors and the use cases of the system.

3.1.1. Visionary Real-Life Scenarios

These scenarios represent the functionalities of the program in a more casual manner. They help ordinary people understand the use cases of the program as well as help facilitate the design of technical use cases by us.

Scenario name	WordCluster
Participating actor instances	Prof. Varol Akman: User
Flow of events	1. Professor Akman is keen on learning new languages but has a full plate with research and teaching. He wants to learn a new language that would take as little effort and time as possible. He discerns that learning a language with many similar words as Turkish may accomplish that goal. 2. He decides to use the website called Etymøn to search for languages that are mapped closely to Turkish. 3. He goes to the Etymøn website and is greeted with a cluster map of words mapped closely together that have the same origin. 4. He chooses to view the map where Turkish is located. 5. The map rotates to the Turkish words and searches around to see words of languages that share the same etymologies and thus have similar words to Turkish. 6. He finds that Persian is such a language and decides to learn Persian. 7. He checks the website daily to find words that are clustered between Turkish and Persian and learns those words.

Scenario name	objectNameTracing
Participating actor instances	Prof. Uğur Doğrusöz: User
Flow of events	1. While eating a pineapple slice, Prof. Uğur immediately realizes that the word “pineapple” is different than its equivalents in most other languages. 2. Intrigued by this thought, Prof. Uğur wants to explore more about the origin of pineapple. 3. Using Etymøn, Prof. Uğur shows the pineapple to the camera of his smartphone. 4. Prof. Uğur gets the origin information as an Augmented Reality word cloud for the recognized pineapple.

Scenario name	WordTracing
Participating actor instances	Prof. Mehmet Koyutürk: User
Flow of events	1. Prof. Mehmet Koyutürk is bored and is looking for a fun way to kill some time on the internet. However, he does not want to waste time. 2. He decides to use the new website he heard called Etymøn. He searches the word 'kalem.' 3. He enjoys as she watches fancy animations turning a sea of languages into a cloud-like figure zooming into the word 'kalem' appears on screen as the website does its job. 4. Then a map that looks like a cloud of words appears on the screen. He sees that word 'Kalem' is connected to the words 'ḳalam' and 'calamus'. 5. He learns that origin of word kalem is coming from word 'kamis' which means 'made from wood'. Also he learns that kalam and calamus are from same origin as kalem.

3.1.2. Use Case Descriptions

Different than visionary scenarios use case descriptions involve system responses to actor's actions.

Use case name	SearchWords*
Participating actors	User, WordDatabase
Entry condition	User wants to search when Etymøn is open
Exit condition	User gets the word cloud and decides to go back to the language sea.
Main flow of events	1. User opens Etymøn. 2. Etymøn generates the language sea. 3. User enters a word to search. 4. Etymøn queries the word from WordDatabase. 5. Then zooms into the word in the language sea and creates a word cloud for the searched word. 6. User looks at the word cloud and after getting the necessary etymological information, decides to go back to the language sea.
Alternative flow of events	● User enters a word unknown to the system. ● Etymøn gets the definition and pronunciation information for the word from a dictionary and trains the machine learning algorithm with it and returns the newly-generated word cloud. ● After getting the word cloud, user jumps to a related word near in the cloud. ● Etymøn transitions into the word cloud that related word.
Quality requirements	● While rendering the word cloud, Etymøn uses the nearest word relations in order to present the adequate number of relevant words for the searched word. ● When a word is not found, a message should be displayed as the training of the machine learning algorithm may take time.

* The use case, SearchRandom, has a similar scenario as SearchWords. Instead of searching for a specific word, the user prompts the system to search for any random word. The system responds just as it does for SearchWords, except that it only searches words that already exist in the database, so there will not be an alternative flow of events.

Use case name	`manageMap`
Participating actors	`Administrator, WordDatabase`
Entry condition	Administrator wants to modify the arrangement of two word nodes in the WordDatabase
Exit condition	Administrator successfully modifies the relation of two words.
Main flow of events	1. Administrator opens `Etymøn`. 2. `Etymøn` opens administration panel. 3. Administrator navigates through WordDatabase and finds the two words she wants to alter. 4. Administrator can choose to delete a word, delete an existing relation or create a new relation between words. She can also strengthen or weaken the particular relation between them. 5. `Etymøn` applies the queried operation to WordDatabase 6. After making the wanted changes administrator saves the changes and exits the administration panel.
Quality requirements	● While navigating through the WordDatabase user interface should help the navigator find his/her target with useful filters and search tools. ● Altering relation between words should be simple and easy.

Use case name	ExploreMap
Participating actors	User, WordDatabase
Entry condition	User wants to explore the words available in the Language Sea and see their relation to other words.
Exit condition	User decides to stop exploring and closes the application
Main flow of events	1. User opens Etymøn. 2. Etymøn generates the language sea. 3. User navigates through the Language Sea. 4. User selects a word in a word cloud. 5. Etymøn displays the information about the word that it gets from the Word Database such as its origin, meaning pronounciation and context. 6. After exploring enough words, word clouds, and languages user closes the application..
Quality requirements	● While navigating through the WordDatabase user interface should help the navigator find his/her target with useful filters and search tools. ● Information about the selected words should be presented to the user in a nice and understandable manner.

Use case name	ScanObject
Participating actors	User, AR Module, Object Recognition Module
Entry condition	User scans an object with their camera and wants to search the word of the object on Etymøn
Exit condition	User sees the word cloud on the scanned object and stops the scanning.
Main flow of events	1. User opens Etymøn. 2. User points at object with camera and scans it. 3. Etymøn queries the object image from the Object Recognition Module to search the word. 4. Etymøn queries the word from WordDatabase to get its features to be trained by the machine learning module. 5. Then zooms into the word in the language sea and creates a word cloud for the searched word. It uses the AR module to display this on the screen where the object is being scanned. 6. User looks at the word cloud and after getting the necessary etymological information and closes Etymøn.
Alternative flow of events	● User scans an object not recognizable to the system. ● Etymøn displays an error message and prompts user to properly focus camera more on the image.
Quality requirements	● Objects scanned are limited to the WordDatabase. If the word database does not have recognized object word, an error message will be displayed.

Use case name	ManageWord
Participating actors	Admin, WordDatabase
Entry condition	Admin wants to add a definition to the word
Exit condition	Admin sees the notification informing whether the word definition was successfully changed.
Main flow of events	1. Admin enters Etymøn system as an admin. 2. Etymøn shows the page with menu for admins. 3. Admin clicks on magnifier which indicates a word search. 4. Meanwhile it is opening the WordDatabase module for interactive search of words. 5. Etymøn returns extended search bar indicating it is ready to search a word. 6. Admin enters a word in search bar. 7. Etymøn shows the list of words starting with the letters entered (for example, if user enters “break”, the system shows “breakage”, “breakdown”, “breakfast”, “breakthrough”, etc) in alphabetical order under the search bar. 8. Admin chooses the desired word. 9. The system shows the page containing the data on the word from WordDatabase. It includes the definition section with multiple definitions, pronunciation, related languages, etc. 10. Admin choose “edit definition” indicated as a pencil near Definitions section. 11. The system returns the Edit Definition page where the data on definition is shown and have +, - and pencil symbols near related sections. 12. Admin chooses a + sign meaning “add definition”. 13. Etymøn shows an empty bar under all existing definitions. 14. Admin enters the new definition and clicks Save. 15. The system updates the definition by adding one more definition to the word. 16. Etymøn shows a notification saying that the update was successful, if it was successful. If it wasn’t for some reason, it shows a notification stating that the word definition wasn’t updated.

Use case name	ManageLanguage
Participating actors	Admin, LanguageSea, LanguageFamily, Language
Entry condition	Admin wants to add a language to language sea
Exit condition	Admin sees the notification on whether the language was successfully added.
Main flow of events	1. Admin enters Etymøn system as an admin. 2. Etymøn shows the page with menu for admins. 3. Admin enters the Language Sea. 4. Etymøn shows the page with language sea options. 5. It shows the list of language families and the list of languages within the families. 6. Admin chooses to add a new language. 7. The system shows the page containing the blank sections needed to be filled about the language, like the name of language, and potions to choose, like language family it belongs. 8. Admin fills the blank sections and clicks Save. 9. The system updates the language sea by adding the new language. 10. Etymøn shows a notification saying that the update was successful, if it was successful. If it wasn't for some reason, it shows a notification stating that the language was not added.
Alternative flow of events	● Admin does not find a language family to which the language should be added. 1. He/she chooses to add new language family in the add new language page by clicking "add new language family" at the end of language families list. 2. Etymøn shows the empty text bar under the "add new language family" asking the name of new language family. 3. Admin types the name and presses Enter. 4. The system creates new language family and shows it in the add new language page under the the language families list.

Use case name	`HallucinateWords`
Participating actors	`User, WordDatabase`
Entry condition	User wants to hallucinate from a given word
Exit condition	User decides to return to the language sea.
Main flow of events	1. User decides to hallucinate a new word. 2. `Etymøn` asks whether the user wants to start from a random word or a specific one. 3. User chooses to begin with a random word. 4. `Etymøn` selects a random word 5. Then zooms into the word in the language sea. 6. `Etymøn` retrieves the features of the word from WordDatabase to be used in the machine learning (ML) algorithm. 7. `Etymøn` starts transfiguring the word into different words based on the outputs from the ML algorithm. 8. User looks at the development of new words, then decides to go back to the general view of the language sea.
Alternative flow of events	● User decides to start with a specific word. ● `Etymøn` directly uses the given word in ML algorithm instead of a random word.
Quality requirements	● Machine learning algorithm that will hallucinate based on the input text should use generative network algorithms.

3.2. Dynamic Model

In this section, the problem domain is analyzed in terms of its dynamic nature in order to understand the required component actions. This dynamic behavior is represented by several scenarios and their respective UML sequence diagrams in the following pages. As the last dynamic model, an activity diagram is presented that goes over the actions taken by the system during the onset of system lifecycle. As previously mentioned, state diagrams are not represented for each class here, as most the classes in the object model are simple ones without multiple states.

Scenario name	`Enter Etymøn`
Participating actor instances	`User`
Scenario	User starts the program by opening the window or mobile application. The system displays the menu and generates a language sea and performs some animation.

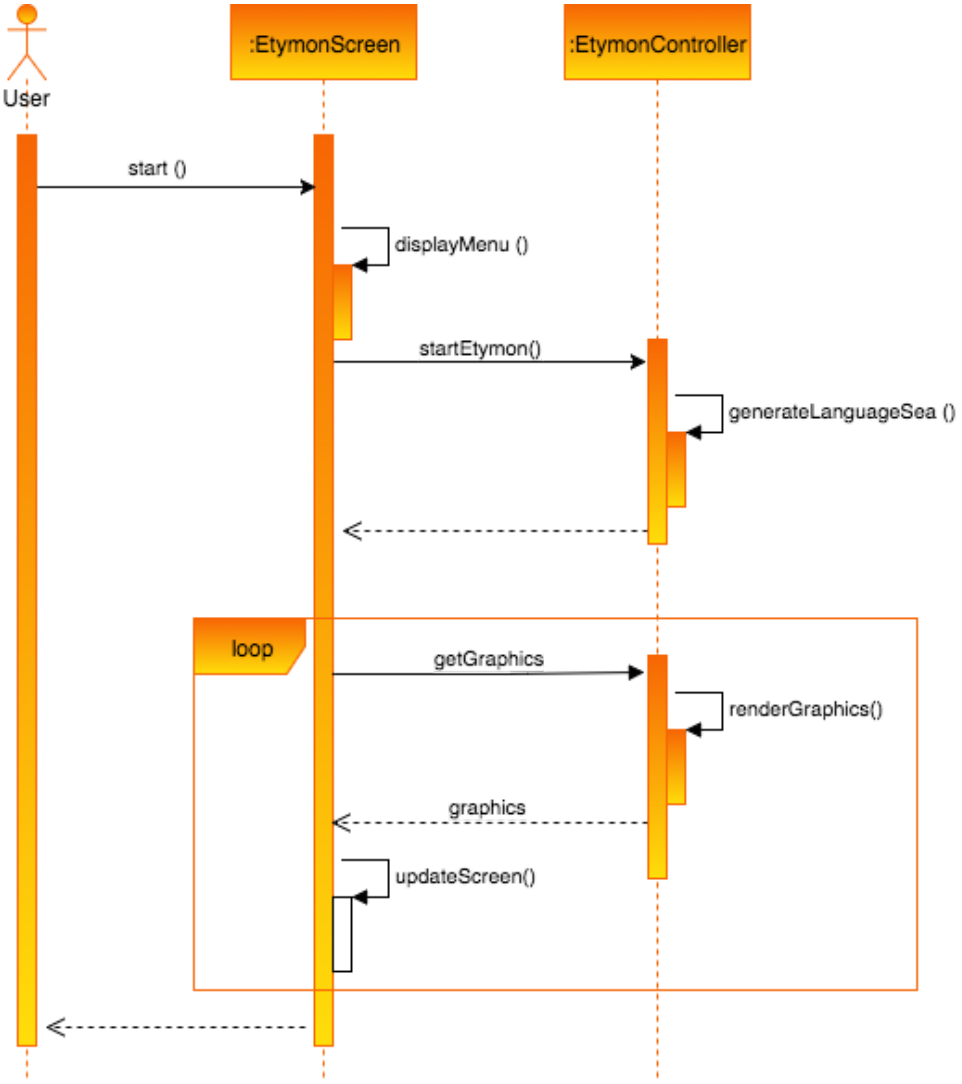

Figure 4 This is the UML sequence diagram for the Enter Etymon scenario.

Scenario name	`Word Search`
Participating actor instances	`User`
Scenario	User chooses to perform a search a certain word. The system looks up for the word in the database. If there is no such word, it generates it (by using definitions and pronunciation data from dictionary). Then it zooms in to the word in the language sea, generates word cloud of that word, and shows it.

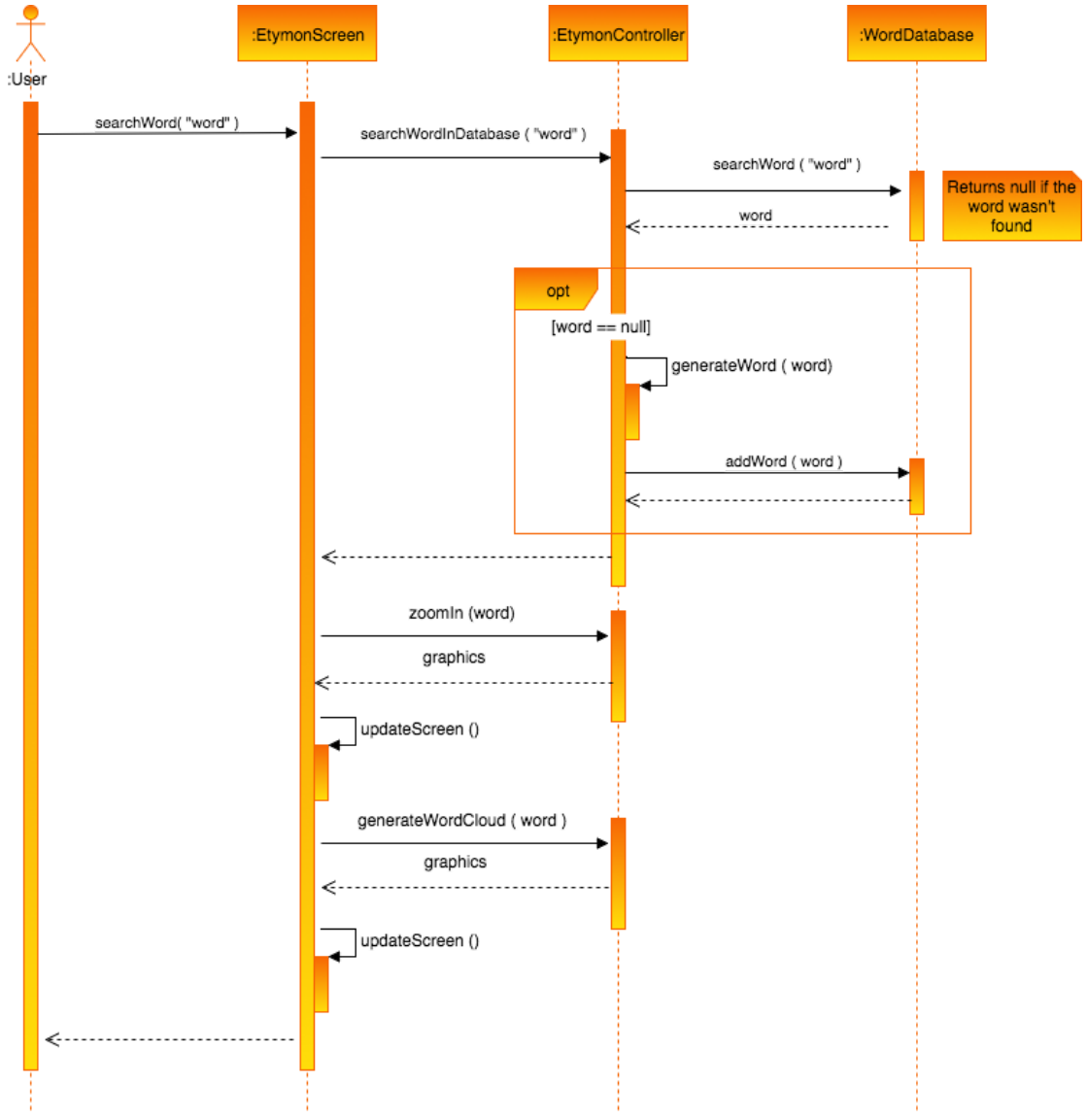

Figure 5 This is the UML sequence diagram for the Word Search scenario.

Scenario name	`Hallucinate`
Participating actor instances	`User`
Scenario	User chooses to perform a search a certain word. The system looks up for the word in the database. If there is no such word, it generates it (by using definitions and pronunciation data from dictionary). Then it zooms in to the word in the language sea, generates word cloud of that word, and shows it.

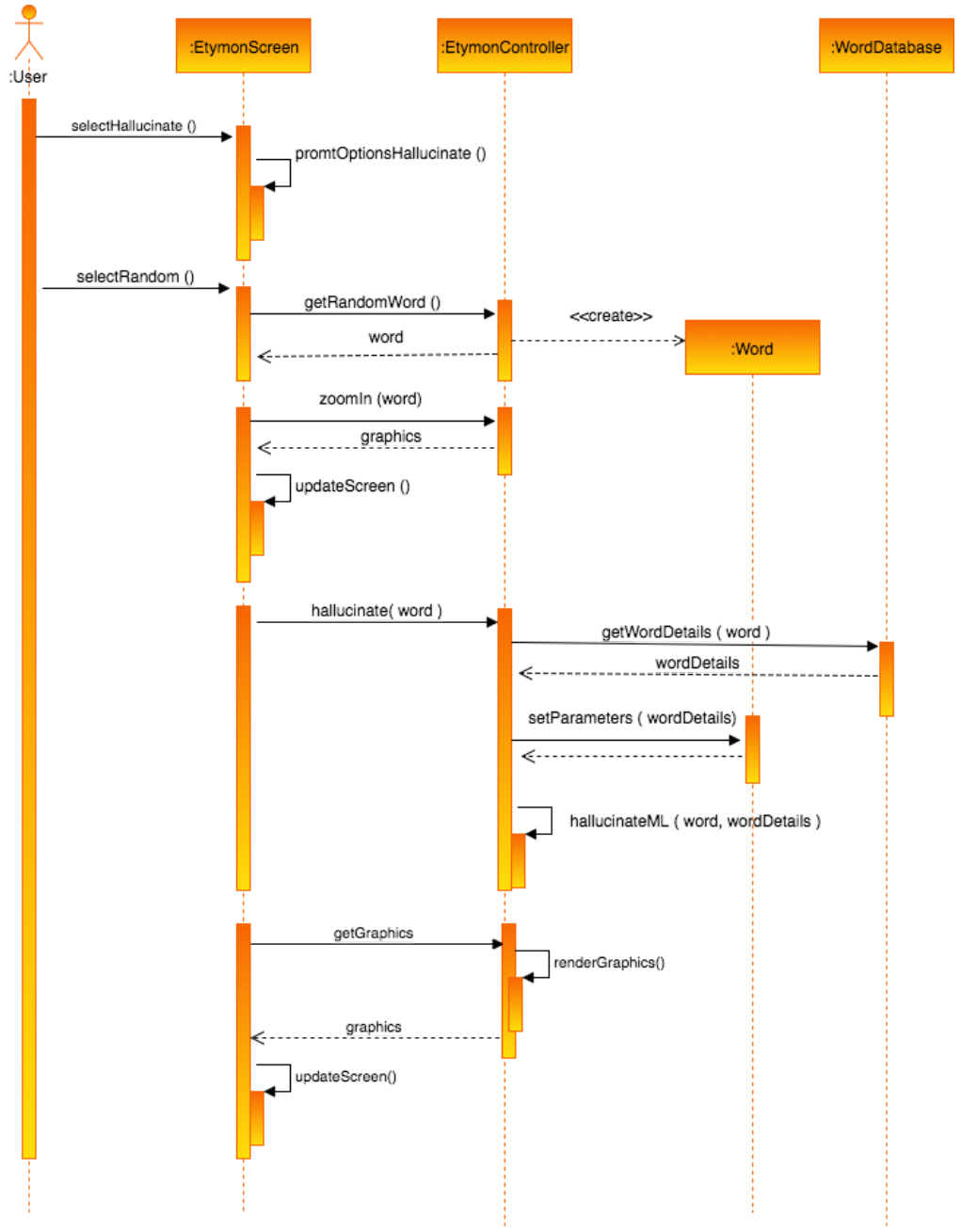

Figure 6 This is the UML sequence diagram for the Hallucinate scenario.

Scenario name	`Scan Object`
Participating actor instances	`User`
Scenario	User uses a phone application version and chooses scan object option. The system first opens the camera and then using the object recognition module scans and recognizes the object that is pointed by user with camera. Object recognition module returns the name of the object. The system then searches the word in database and shows it on the object using augmented reality module. Then it zooms in to that word on language sea, generates a word cloud, and displays it still using AR module.

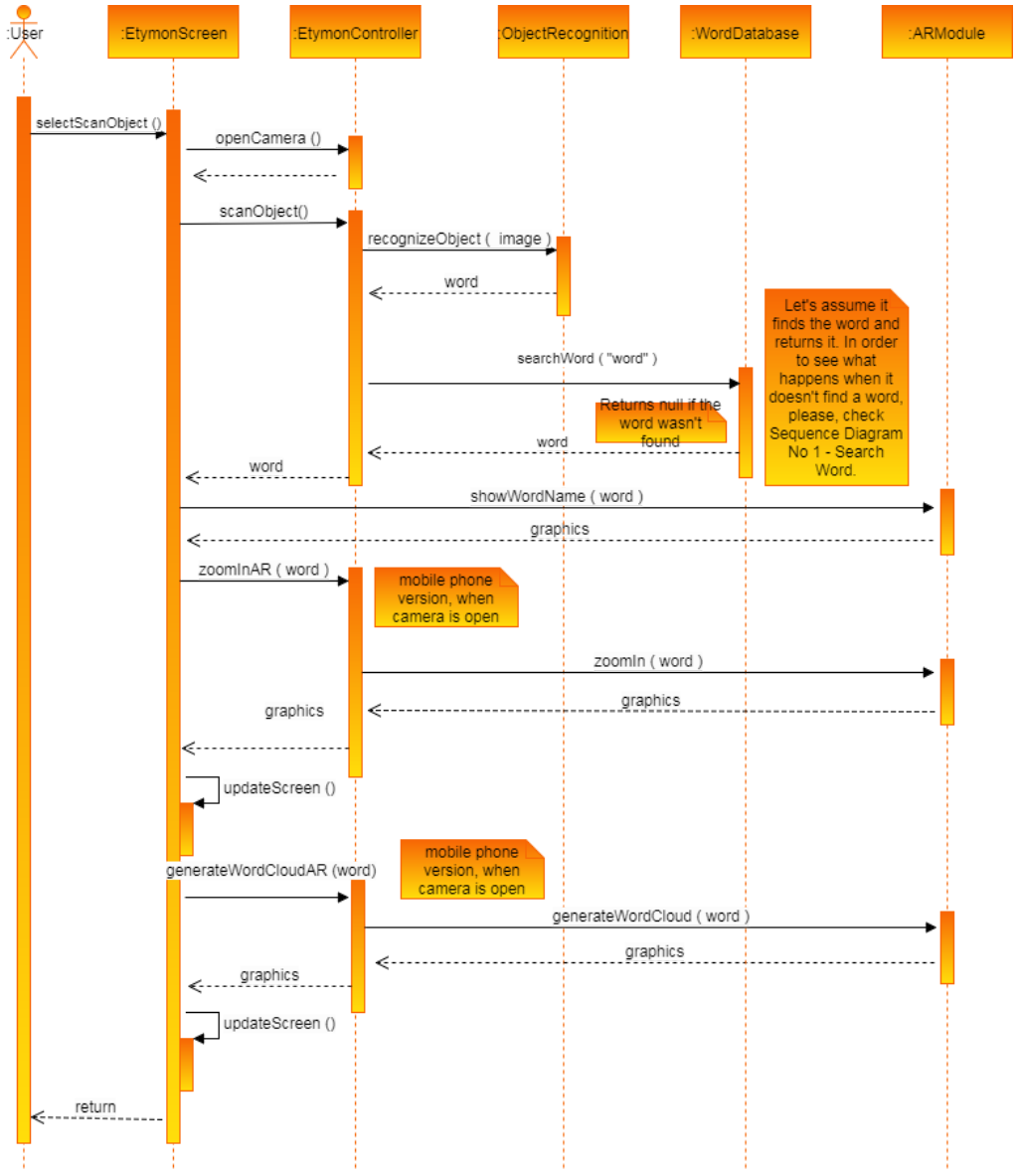

Figure 7 This is the UML sequence diagram for the Scan Object scenario.

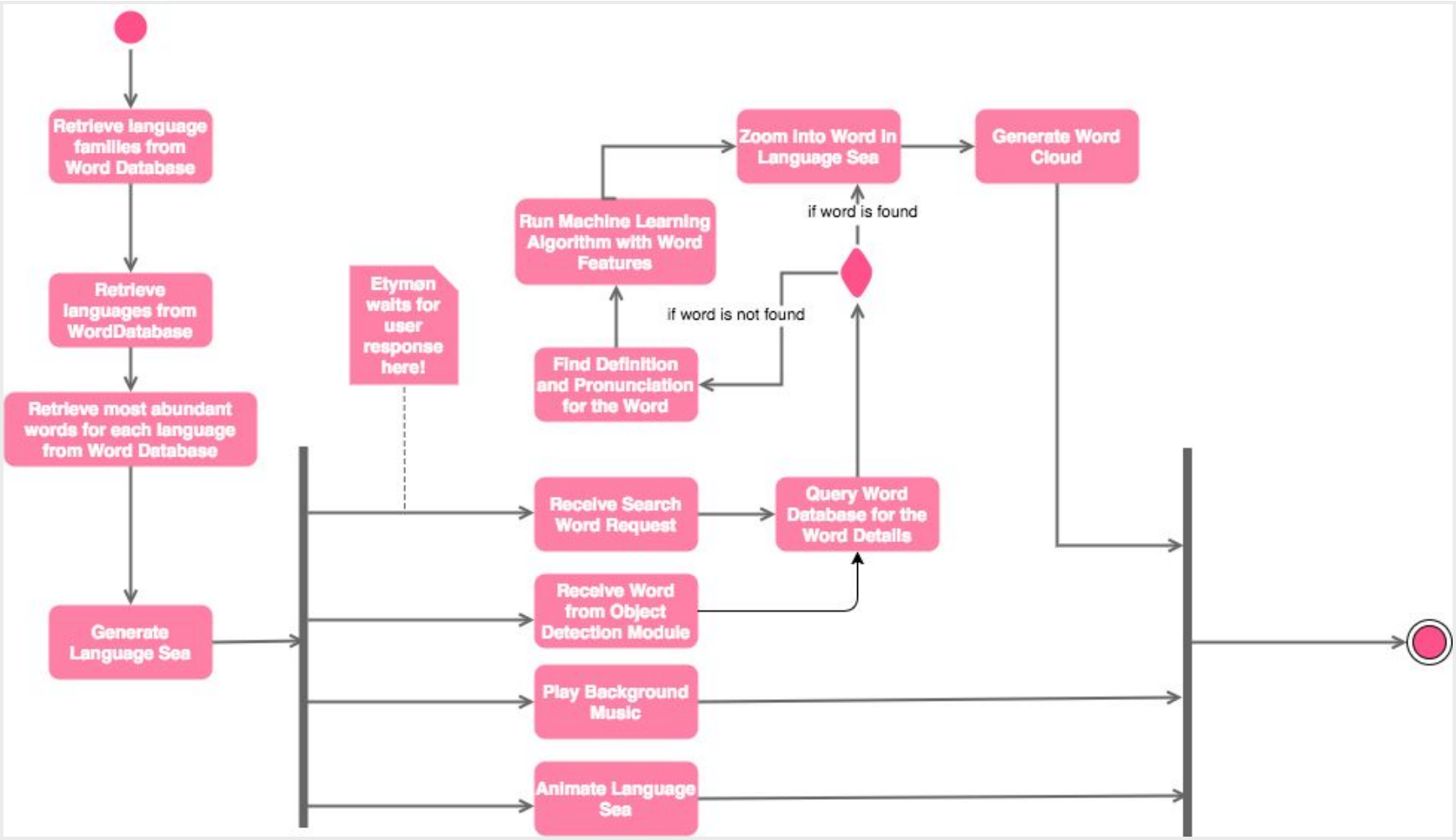

Figure 8 This figure shows the event flow of Etymøn, it is the UML activity diagram of the system.

3.3 Object & Class Model

In this section, relations between objects of the Etymøn System are shown using an UML class diagrams. Attributes and methods of the various classes are represented.

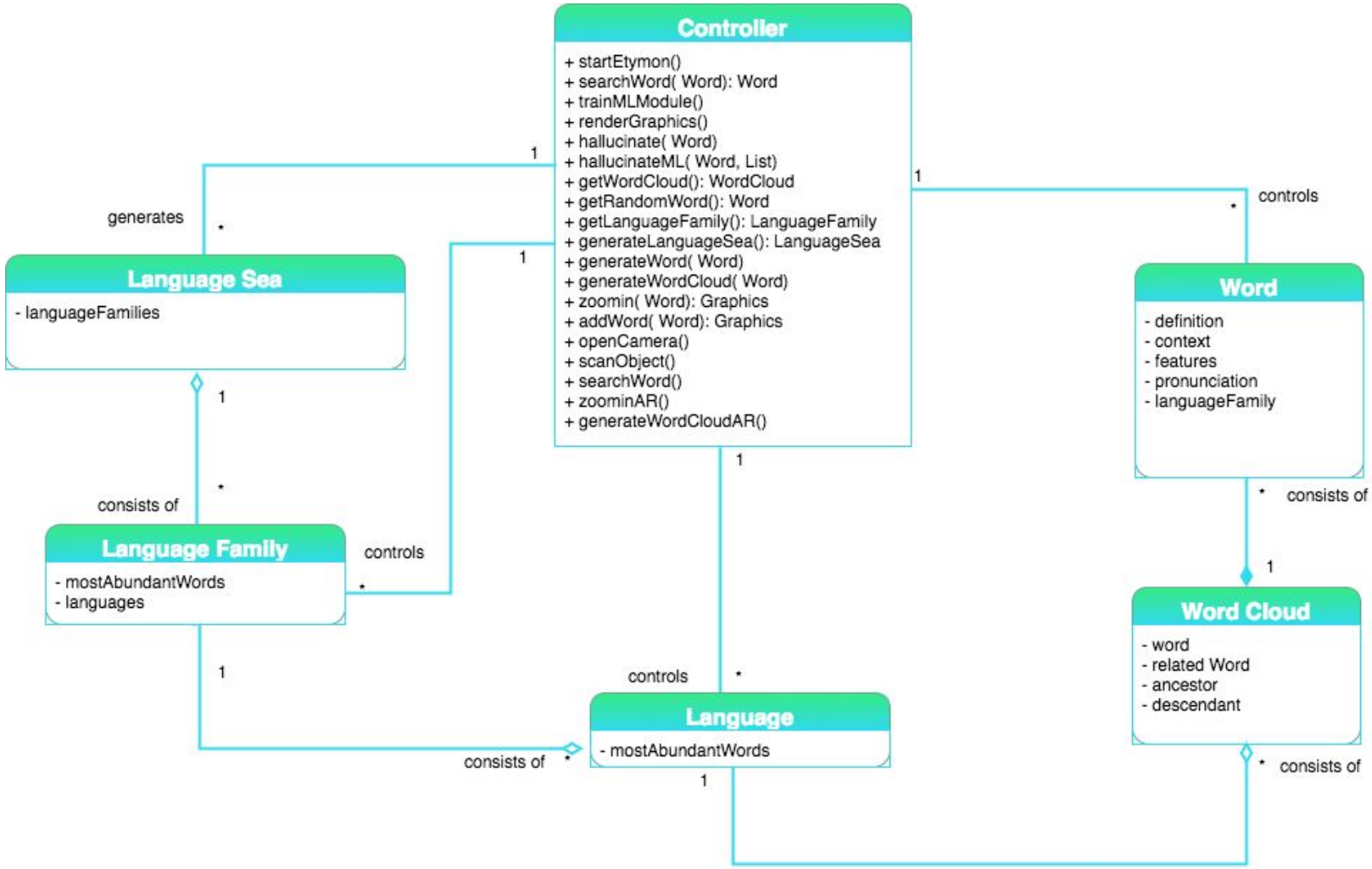

Figure 9 This figure shows the UML Class diagram of the system. The main component is the controller and the other classes are trivial representations of actual objects in problem space.

3.4. User Interface

This section gives the details of the user interface of the Etymon system. The overall walkthrough of the user interface is given in Figure 10. Based on these, screenshots of several screens are presented in the following figures.

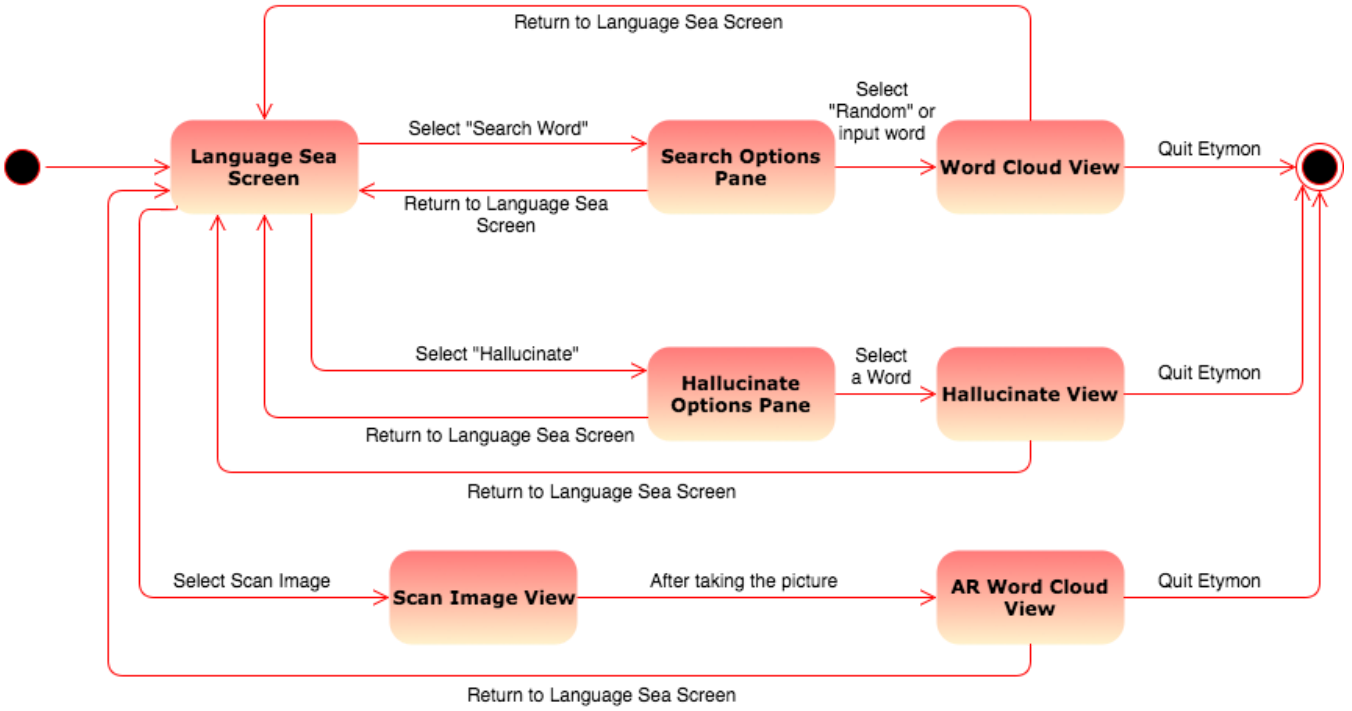

Figure 10 This is the diagram for the UI transitions based on input from user.

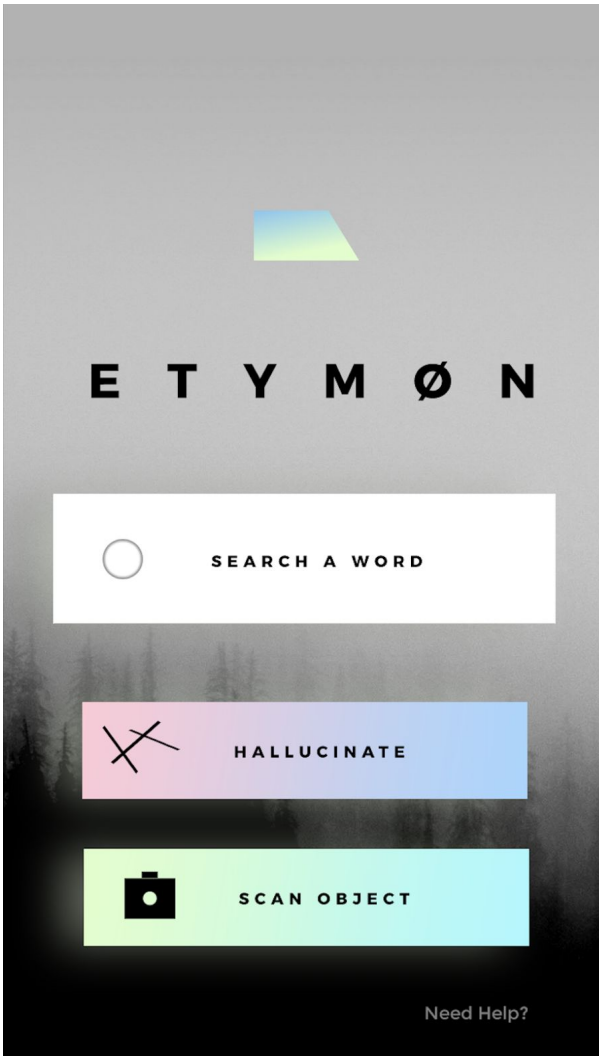

Figure 11 Etymøn's welcome page. From this panel you can go to search word, hallucinate and help panels. You can also "Scan Object" option to open your camera and use augmented reality functionality.

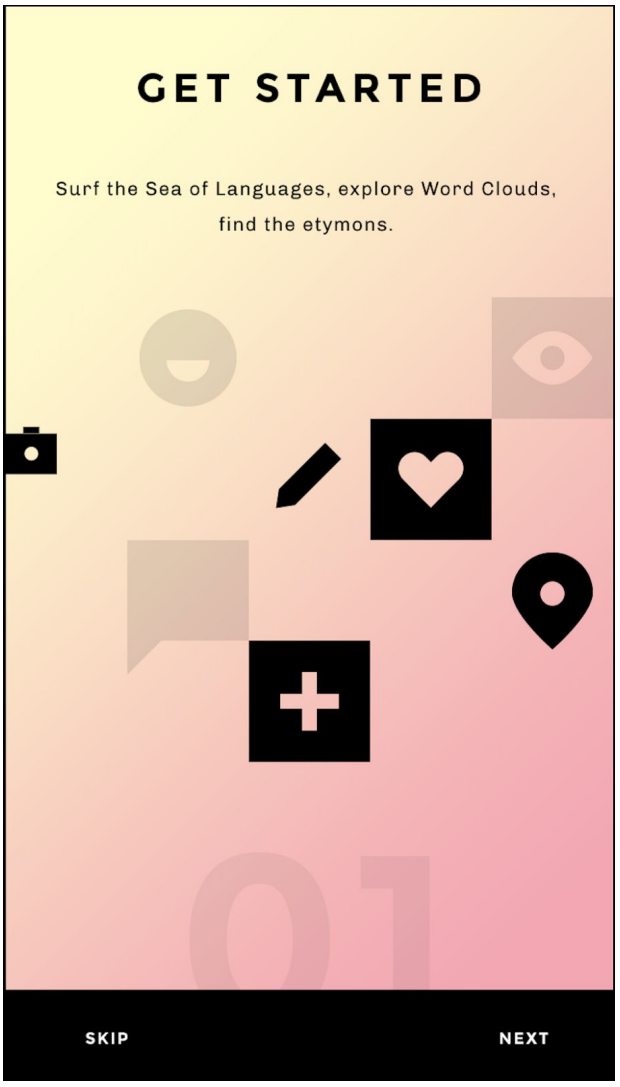

Figure 12 This figure represents the help panel Etymøn provides to the user. Slides of panels will help user to learn how to use the application efficiently.

Figure 13 In this panel Etymøn will show random words from same word cloud —etymologically similar words— and will hallucinate, imagine new words from the words it selected and present it to the user.

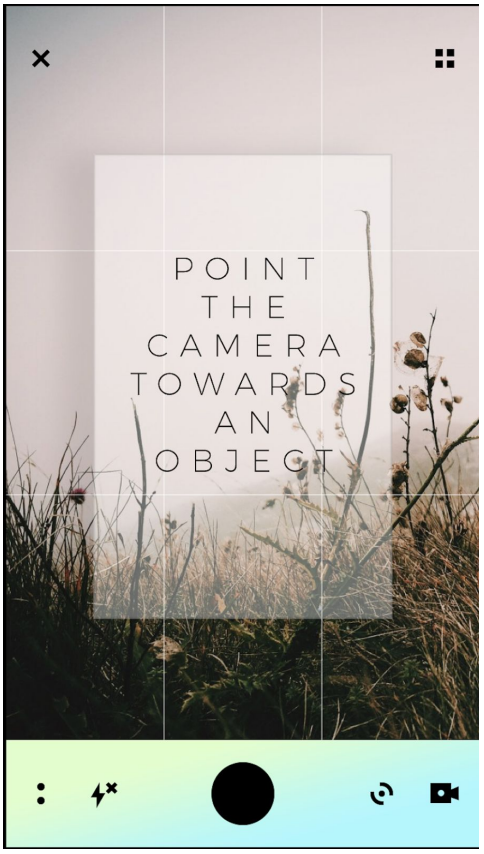

Figure *14* This figure represents the augmented reality functionality of Etymøn.

4. References

[1] C. Diagne and N. Barradeau, *Free Fall,* https://artsexperiments.withgoogle.com/freefall/wave. [Accessed: 09-Oct-2017].

[2] J. Redmon, *YOLO: Real-Time Object Detection*. [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed: 09-Oct-2017].

[3] "Word2vec," *Wikipedia*, 26-Sep-2017. [Online]. Available: https://en.wikipedia.org/wiki/Word2vec. [Accessed: 09-Oct-2017].

[4] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.