Bilkent University

Department of Computer Engineering

Senior Design Project

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

Project Specifications

Nashiha Ahmed, Mert İnan, Cholpon Mambetova, Utku Uçkun

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

Project Specifications
Oct 9, 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.

Contents

Project Specifications

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

1. Introduction

In this report, a general introduction to Etymon is presented.

Etymon 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. Current etymological analyses rely on pattern matching or tracings between different languages by experts in linguistics [1], yet it may be cumbersome or even improbable to detect word origins in situations where direct links cannot be observed between two different words. In this case, Etymon will pose an advantage as it will be using a large corpus of data in order to match words in any given language.

Various studies carried out by linguistic experts and historians improved the understanding of language and its origins [2]. However, there is still "room for improvement" in the field. Currently, most of the studies target the Proto-Indo-European language family [2]. There is sparse research done for other languages, and there is not a single, unified resource for this information. Most of the information is scattered online or among other forms of literature.

Since there is no similar project in the market yet, our software will be designed from scratch, which would make it a greenfield project. However, we will use other existing algorithms to build our software, such as deep learning algorithms among others that will be specified further in the report.

In the following sections, detailed description of the system, its constraints and requirements are discussed. In addition, issues of professional and ethical importance are also evaluated in order to consider the effect of the system on the society and the scientific community.

1.1. Description

Etymon will be a system that contains three main components: deep-learning, augmented-reality with object recognition, and generative dreaming. Etymon bases most of its functionalities on the etymological map that will be generated by the deep learning algorithm. Etymon system will include a palpable product in the end as an online application based on this etymological map. The system and its implementation will be broken down into three stages in respect to its components.

In the initial stage of the project, a deep-learning algorithm will be used to cluster words from different languages to derive a general map of language families and their relation to each other. This will be done based on word and pronunciation data acquired from Wikipedia. Furthermore, for English, pronunciation data from online sound libraries will also be used, and for other languages, International Phonetic Alphabet (IPA) readings of the words will replace sound files for clustering. At this stage, a crucial consideration will be finding a way to convert words into vectors —this is called "word embedding" in

literature [3]— in order to classify them according to their features using machine learning algorithms. At the end of this stage, output from the deep learning algorithm will be a network of words. As the nodes in this graph will be immense in number, instead of showing each word, large clusters of similar words will be sufficient to show language families (Figure 1). This graph will be the etymological map to be used in the consecutive stages of the project and it will be the default graph to be shown when the user wants to search a word.

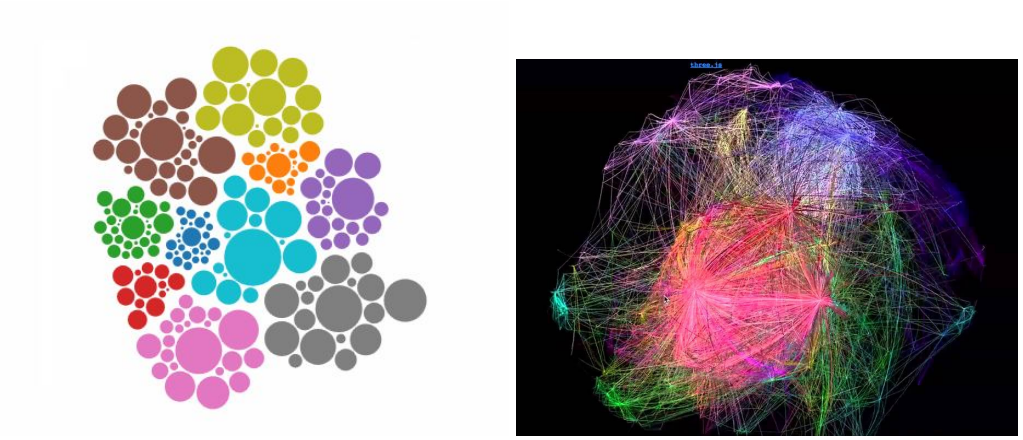

Figure 1: This figure shows two graph visualizations by different softwares that may guide the appearance of the etymological map [4][5].

In the next stage of the project, the etymological map will be used to display specific graphs for words that the user would search in the system. For each word entered by the user, the system will create a smaller and localized graph (Figure 2). This map will show connections between the root of that word and words in different languages. Augmented-reality will be used in this stage as an additional feature. Users will be able to point their cameras to objects and trace their etymological origins. After object recognition, the text will be searched in the etymological map and that map will be overlaid on the object itself using augmented reality.

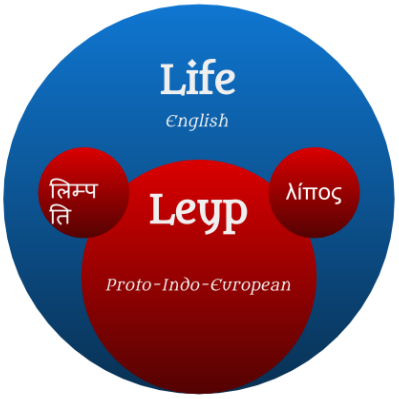

Figure 2: This figure shows a local graph for the English word, "life". Its origin is identified to be "leyp" in the Proto-Indo-European language family, and two descendent words —one in Sanskrit and one in Greek— are given next to the origin word.

As the last stage of the project, the etymological map will be fed into a deep dreaming algorithm in order to generate new words from roots that are present in the language families. This stage will present a creative perspective on the evolution of languages and show alternative formations that languages could have devised. This stage will combine data visualization and deep learning in a visually-appealing manner. When the system is in idle state, it will "hallucinate" these new words, much like that done by a deep learning archiving installation called "Archive Dreaming"[6].

Similar mobile applications such as, "Etymology Explorer" [7], and "Oxford English Etymology" [8] or web services such as "Online Etymology Dictionary" [9] exist in the market yet are largely focused on etymological roots of English words. They also do not analyse words that share the same etymological origin or display languages that exist in the same language families.

The preeminent potential users of Etymon are simply those who are interested in tracing the roots of words. This may range from linguists to anthropologists and historians to the average language learner. Our aim is to provide a tool for the interested to quickly and conveniently visualize word roots and language families. The user can use this program to look up meanings of the words through their etymological origins. For example, one can easily look up the etymological root of his or her name. Also, it can be used as a translator in the sense that one can find words in the other language sharing a common root and similar meanings.

We will use personal testing as well as expert knowledge to validate the accuracy of our program. Different languages will be tested separately to get more representative results. We are planning to use an open-source, pre-trained object recognition algorithm which will require less testing for the augmented reality stage of the project.

All in all, the general description of the Etymon system can be categorized into its three main stages and in the following sections, constraints and requirements of these stages will be supplemented.

1.2. Constraints

In this section, crucial considerations of the Etymon system are listed in detail. Discussing these constraints are valuable to define the details of software implementation.

1.2.1. Implementation Constraints

- The main bottleneck of the system will be the number of words available online in each language.
- The program will require continuous internet connection for word search functionality.
- The search functionality may be slowed down by the increasing size of the etymological map.
- Machine learning algorithms will not run on user devices but on servers to manipulate the large dataset.
- In the augmented reality component, object classification will be restricted by the capabilities of the pre-trained model.
- The amount of words present in each language creates hardware limitations. The system will require additional storage and processing resources.
- Graph visualizations will be done using WebGL and Three.js.

- darkNet [10] and YOLO [11] will be used for object recognition in the augmented reality stage.
- 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 algorithm cannot be used. A new algorithm must be created to cluster words after turning them into vectors [12].

1.2.2. Economic Constraints

- Computing power and database storage may be rented from services like Amazon AWS and Google Cloud Service to run the deep learning algorithm on the dataset.
- The web application will be free to use by all users.
- Open source and free frameworks or machine learning algorithms, such as YOLO, will be used instead of paid products.
- Sound files may need to be bought to include pronunciations in the algorithm in order to avoid copyright infringement.

1.2.3. Ethical Constraints

- The software may cause conflict of interests with linguists and may pose a threat to their expertise.
- Search history of the user will not be saved in order to protect privacy.
- Data entered by the user to train the machine learning model further will be used anonymously and for a short amount of time.

1.2.4. Social Constraints

- Profanity and other inappropriate words may need to be filtered or removed from the dataset.
- Some words that are falsely attributed to different origins as a result of folklore will be shown with their original roots causing inaccuracies.
- Our capacity to evaluate the program’s accuracy in some languages will be limited by the knowledge of the experts in those languages.

1.2.5. Political Constraints

- Countries united within similar language families that are currently under political conflict may give rise to political issues.
- Languages that do not have a highly developed dataset or a dataset that is difficult to find may not be sufficiently represented by the software, which may be viewed as a bias.
- Some words in languages having different dialects might be shown under the same language; therefore, creating political conflicts.

1.3. Professional & Ethical Considerations

Ethical and professional responsibilities of all team members during the onset of this project will include:

1. Giving credit to intellectual property: Each team member and the system itself will be responsible with the data or code that they will use in the project and will give credit in order to protect intellectual property rights of the users or experts in linguistics.
2. Striving to achieve high quality work in both the processes and products of development: All team members are responsible to paying attention to product development. During the implementation phase, commenting, testing and readable code-writing are essential aspects in addition to concise and effective report writing.
3. Evaluating critically the outcomes of machine learning algorithms: The outcomes of the machine learning algorithms will be evaluated according to quantitative error measuring strategies in the literature and the efficiency and correctness of the algorithms will be evaluated critically based on these measures.
4. Understanding and reminding the uncertain foundations of certain machine learning approaches: As machine learning algorithms that this project will include are black-box models, their internal working mechanism is not understood-well. Hence while using these algorithms, precautions on their output should be taken.
5. Evaluating quantitatively the effectiveness of products for human consumption: As the end products of the system include AR components and visualizations, their effectiveness will be evaluated by surveys or tests that will be done in cooperation with the users.

In addition to the code of conduct that every group member affiliated with the project would abide by, we would like to further discuss the ethical implications of the system.

Ethical considerations can be categorized into two main sections: user privacy and anonymity, and use of intellectual property by the machine learning algorithm. As the user data will be used extensively, it will be a crucial consideration to take care of the anonymity of the data input by the user. In addition, as most of the data used by the machine learning algorithm will contain work from experts in linguistics and the users that input the algorithm, it is vital to discuss the credibility of the output.

As aforementioned user privacy will be regarded. Therefore, when a user searches a word or takes or uploads a picture for the augmented reality feature, user information, for instance IP addresses, will not be stored. In addition to this, encryption on user material can also be implemented. This also implies that no attribution will be given to the user for the input data that improves the "learning" algorithm, which may give rise to intellectual property disputes.

Moreover, resources that have already been classified by linguists will be used, which may be considered as a threat to human intellect. This may also give rise to intellectual property disputes, as we will be using material without any direct attribution to the experts in the field, even as we take careful consideration copyright laws when using datasets. Although the purpose of Etymon is to aid users such as linguists, the advancement of the program could lead to a decrease in the need for the expertise of linguists and translators in research on etymology. This could give rise to the classic issue in the field of artificial intelligence of artificial intelligence "taking over" jobs.

2. Requirements

- The system should provide etymological information for every word input by the user.
- Etymon 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.
- The system should show word hallucinations while it is idle and no user input are entered.
- The system should "reorder" the map when user searches the etymology of a specific word.
- If a word does not already exist in the map, the system should provide the user with relevant error and help information.
- Etymon should convert words into vectors in order to be used by the machine learning algorithm (word embedding).
- The system should identify closely related languages as language families. It should cluster languages in the same families closer together.
- Etymon should have a simple and straightforward user interface.
- The system should use a deep learning algorithm that is optimal for word clustering.
- Etymon should be easily navigated by the user.
- The system should provide definitions of the words in addition to their origin information.
- Etymon should give reliable output after proper analysis. A user should not get wrong matches.
- The system should provide augmented reality overlays on objects recognized in still pictures and moving camera objects.
- The system should provide fast augmented reality overlays on the objects.
- Etymon should provide etymology of the word in the desired language in the augmented reality feature.
- Etymon should improve etymology maps and clusters when related user and expert data is given.
- 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.

3. References

[1] "Introduction", *Etymonline.com*, 2017. [Online]. Available: http://www.etymonline.com/columns/abbr?utm_source=etymonline_footer&utm_medium=link_exchange. [Accessed: 09- Oct- 2017].

[2] "Etymology", *en.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Etymology. [Accessed: 09- Oct- 2017].

[3] "Word Embeddings", *en.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Word_embedding. [Accessed: 09- Oct- 2017].

[4] Nodes, "D3 force layout - How to achieve 3D look of nodes?", *Stackoverflow.com*, 2017. [Online]. Available: https://stackoverflow.com/questions/24673627/d3-force-layout-how-to-achieve-3d-look-of-nodes. [Accessed: 08- Oct- 2017].

[5] "Graph", *I.ytimg.com*, 2017. [Online]. Available: https://i.ytimg.com/vi/qHkjSxbnzAU/maxresdefault.jpg. [Accessed: 08- Oct- 2017].

[6] F. Visnjic, "Archive Dreaming – Building relations and drawing alt-history with machine learning," CreativeApplications. [Online]. Available: http://www.creativeapplications.net/vvvv/archive-dreaming/. [Accessed: 09-Oct-2017]

[7] "Etymology Explorer -", *etymologyexplorer.com*, 2017. [Online]. Available: http://www.etymologyexplorer.com/ety_explore/. [Accessed: 09- Oct- 2017].

[8] "Oxford English Etymology", *Google Play*, 2017. [Online]. Available: https://play.google.com/store/apps/details?id=com.mobisystems.msdict.embedded.wireless.oxford.oxfordenglishetymology&hl=en. [Accessed: 09- Oct- 2017].

[9] "Online Etymology Dictionary", *etymonline.com*, 2001. [Online]. Available: http://www.etymonline.com/. [Accessed: 09- Oct- 2017].

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

[11] 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.

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

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.

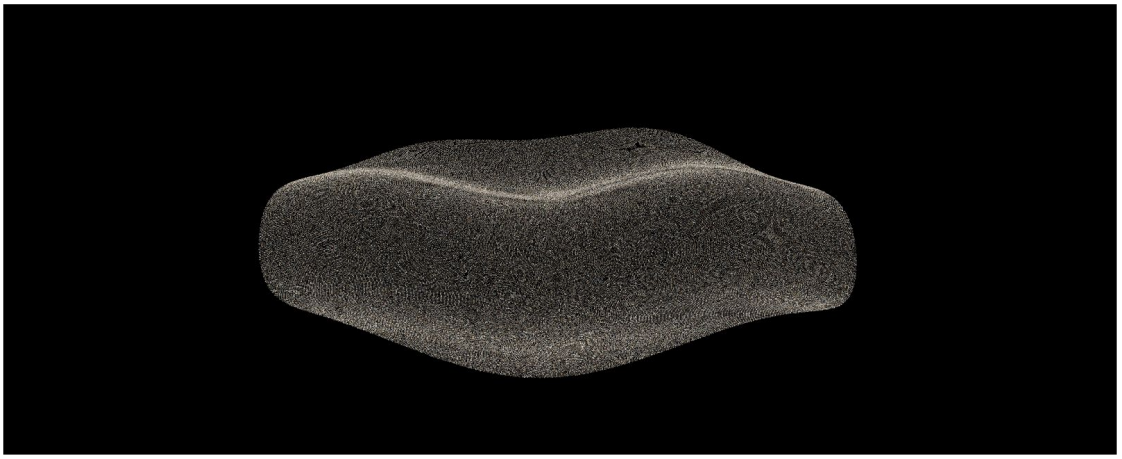

Figure 1 This figure depicts a wave-like pattern that will be like the Language Sea [1]

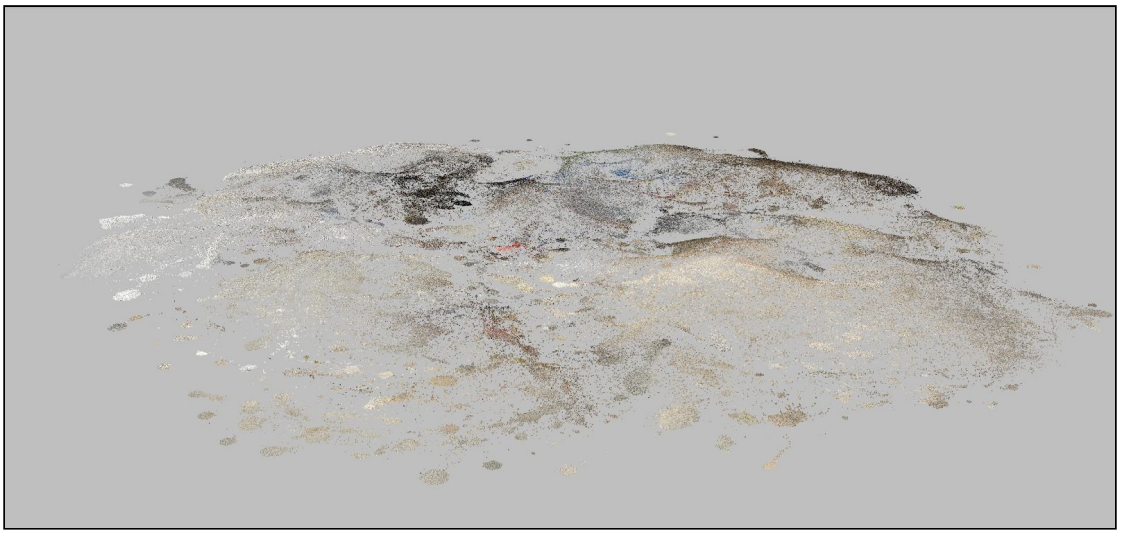

Figure 2 This figure is another clustered space that will be like the Language Sea. [1]

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

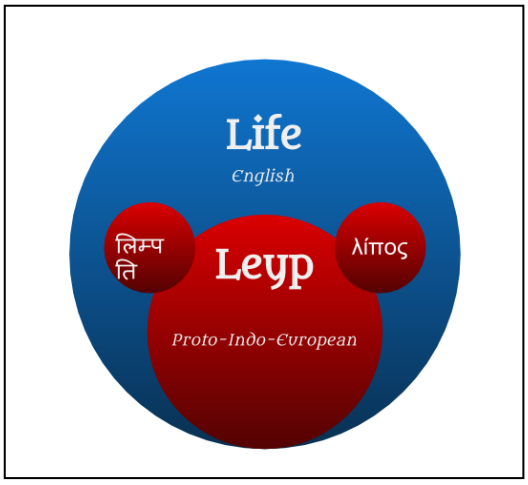

Figure 2: This figure shows a local graph for the English word, "life". Its origin is identified to be "leyp" in the Proto-Indo-European language family, and two descendent words —one in Sanskrit and one in Greek— are given next to the origin word.

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.

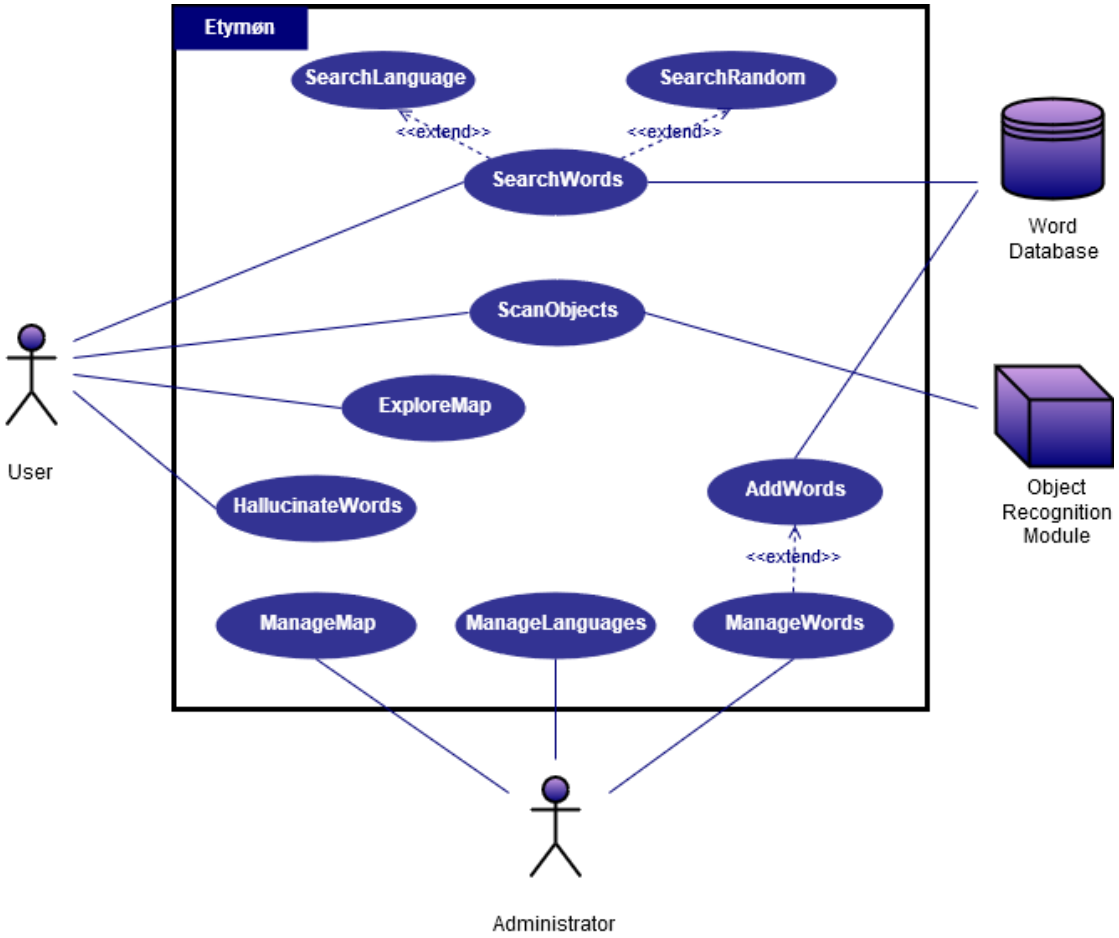

Figure 3 This figure is the UML Use-Case diagram that shows the general relationships between the actors and the use cases of the system.

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.

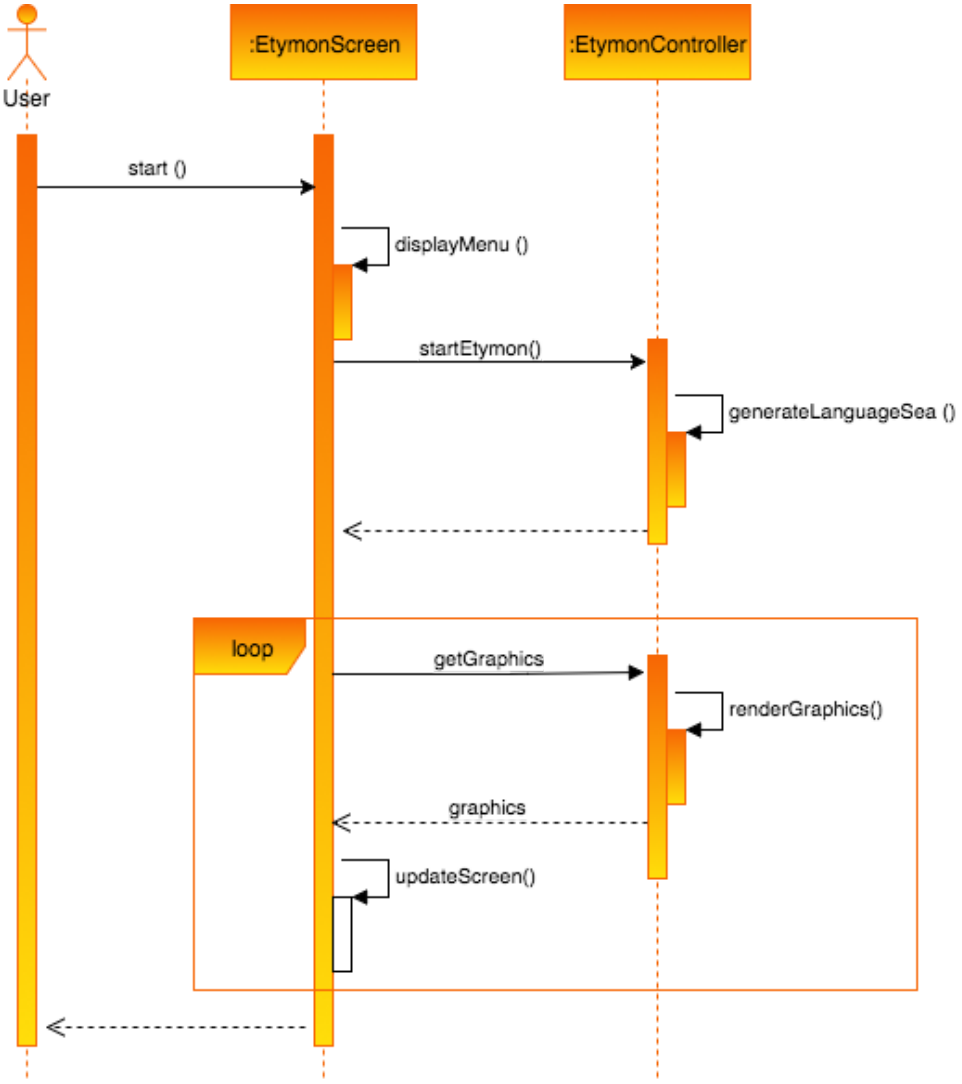

Figure 4 This is the UML sequence diagram for the Enter Etymon scenario.

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.

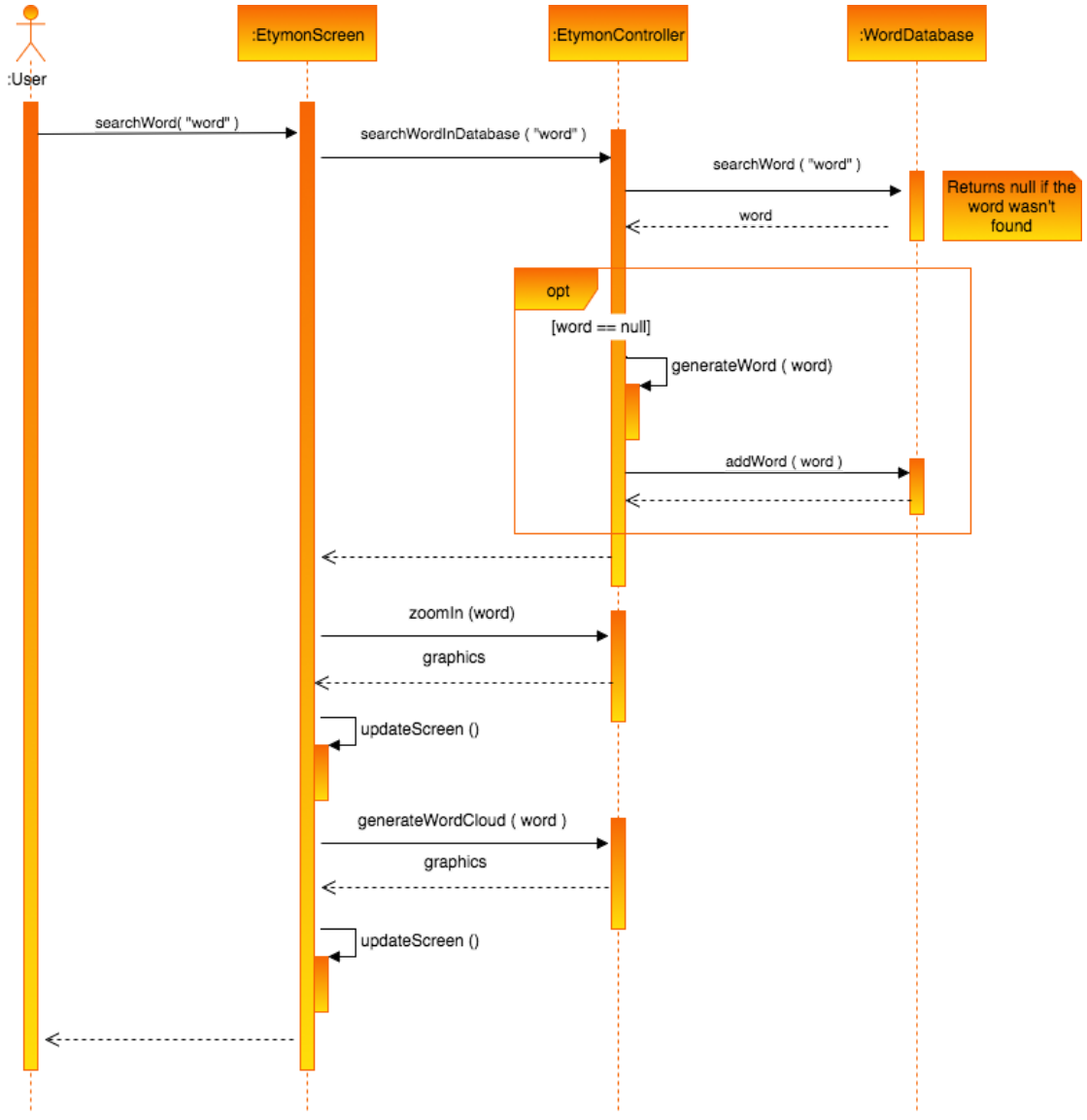

Figure 5 This is the UML sequence diagram for the Word Search scenario.

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.

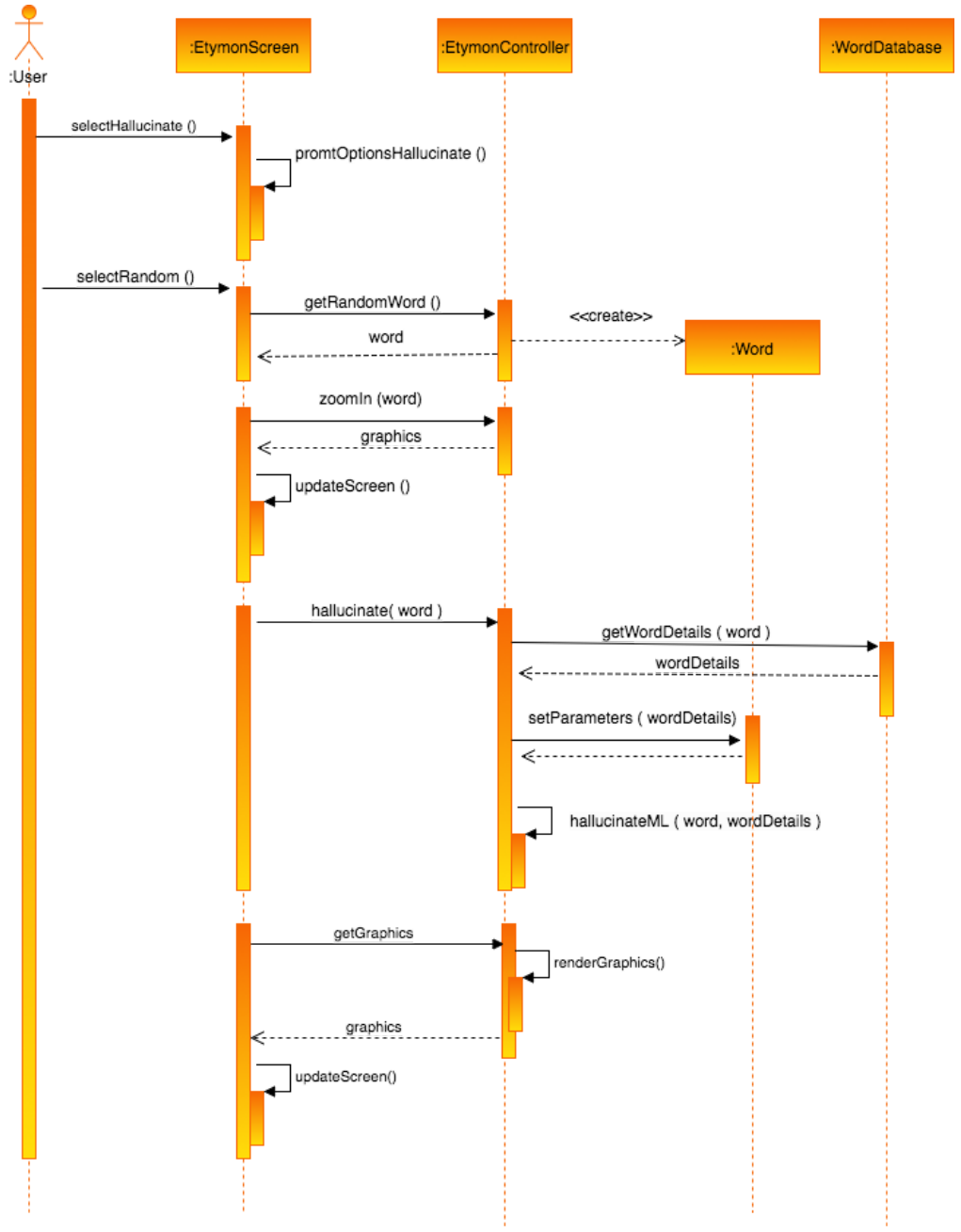

Figure 6 This is the UML sequence diagram for the Hallucinate scenario.

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.

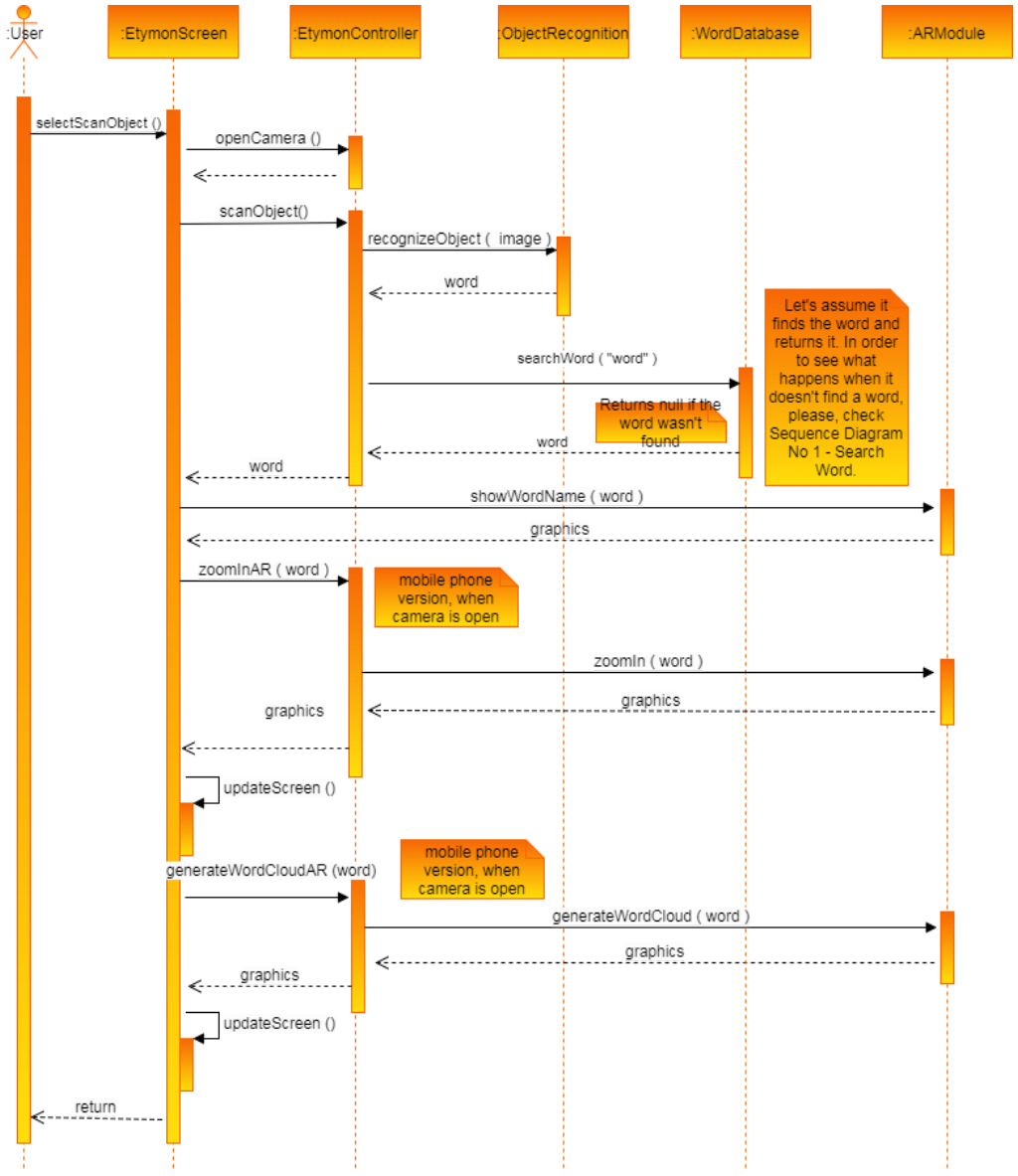

Figure 7 This is the UML sequence diagram for the Scan Object scenario.

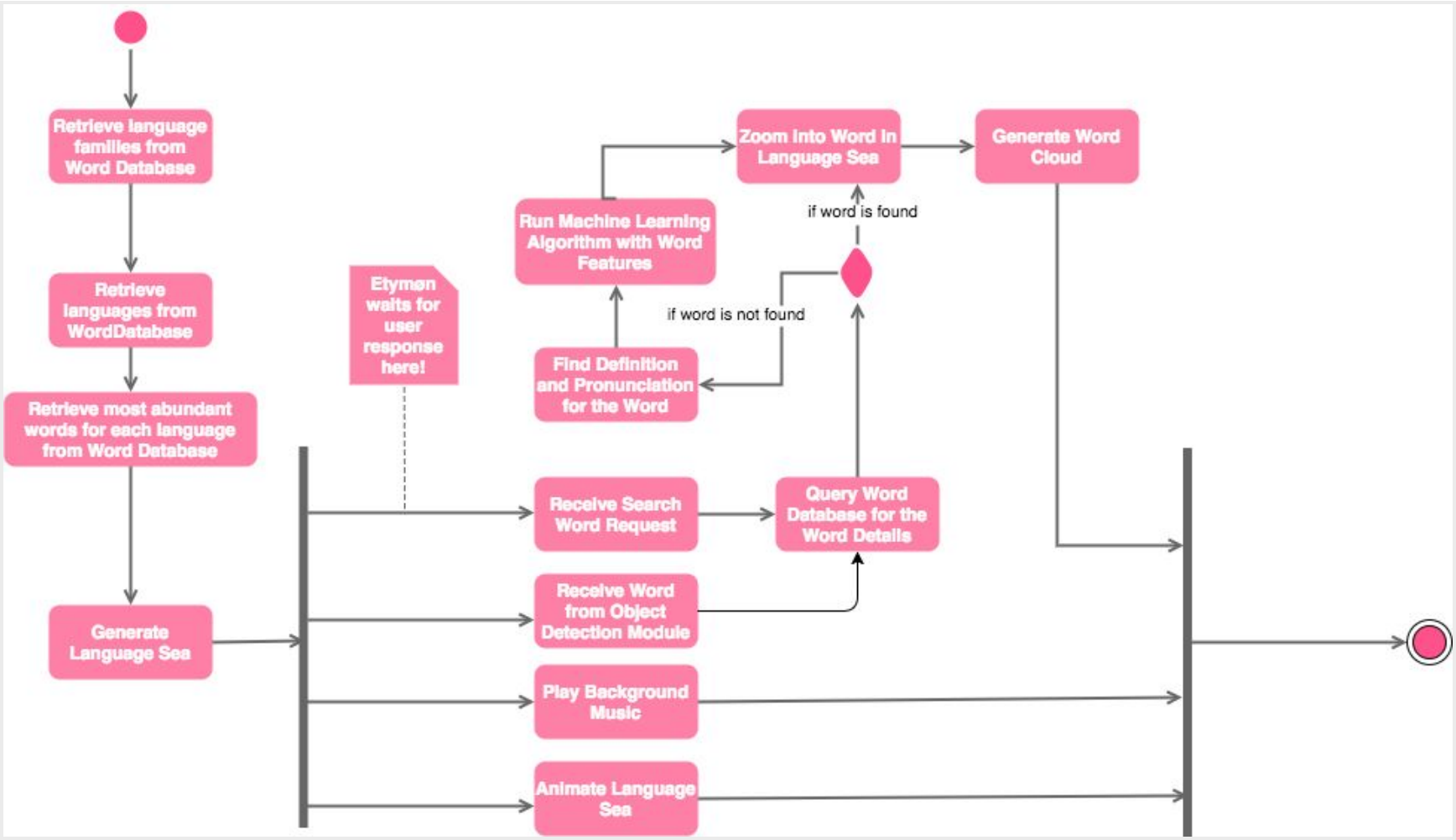

Figure 8 This figure shows the event flow of Etymøn, it is the UML activity diagram of the system.

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.

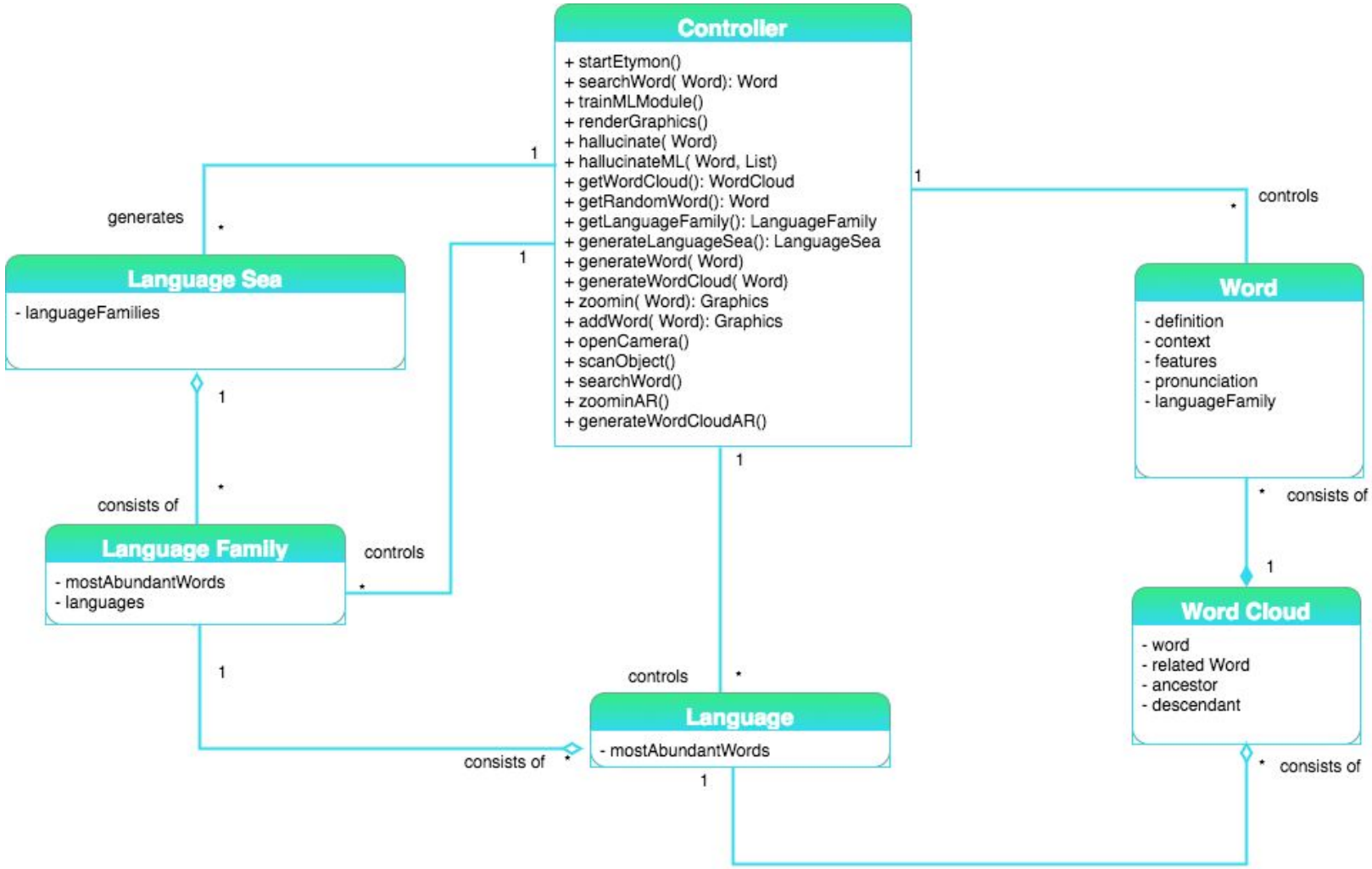

Figure 9 This figure shows the UML Class diagram of the system. The main component is the controller and the other classes are trivial representations of actual objects in problem space.

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.

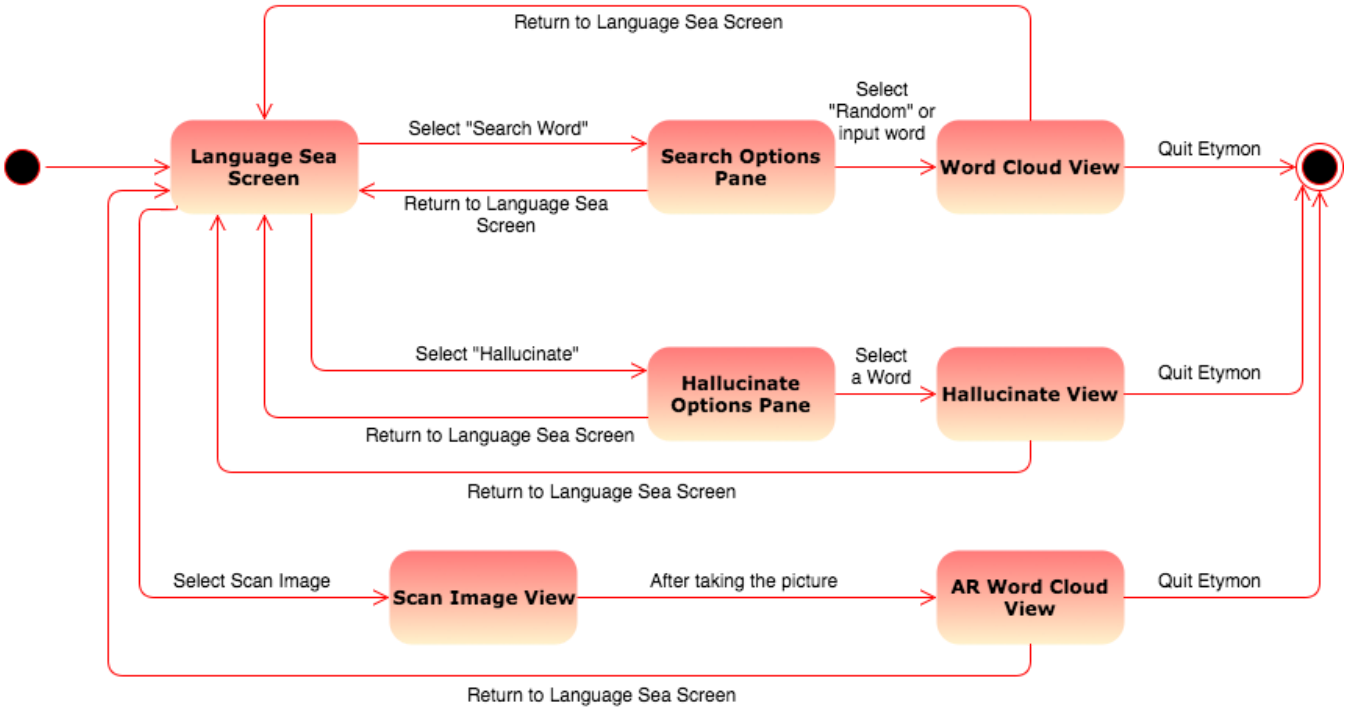

Figure 10 This is the diagram for the UI transitions based on input from user.

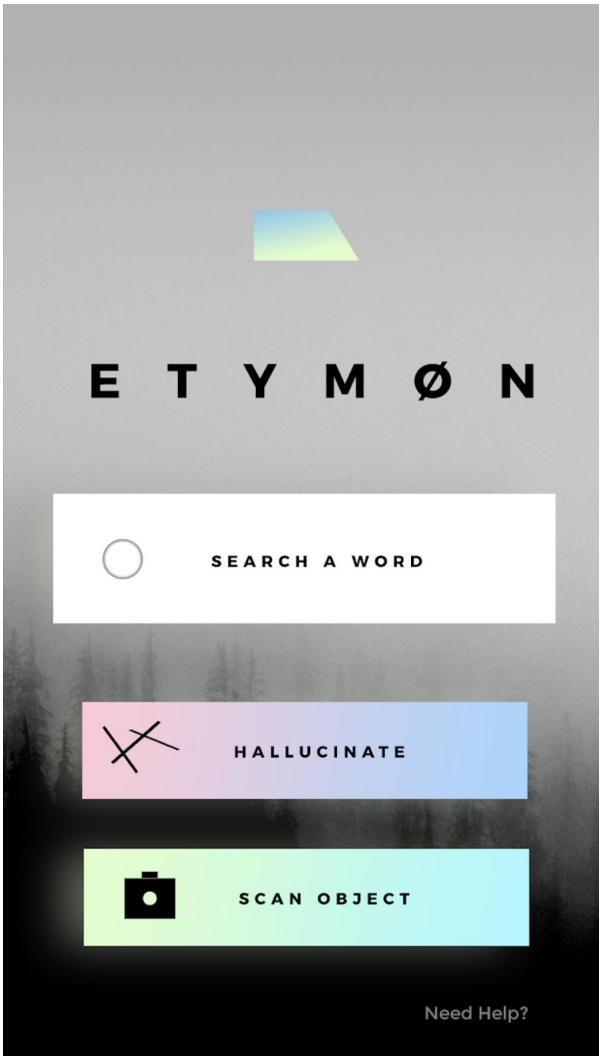

Figure 11 Etymøn's welcome page. From this panel you can go to search word, hallucinate and help panels. You can also "Scan Object" option to open your camera and use augmented reality functionality.

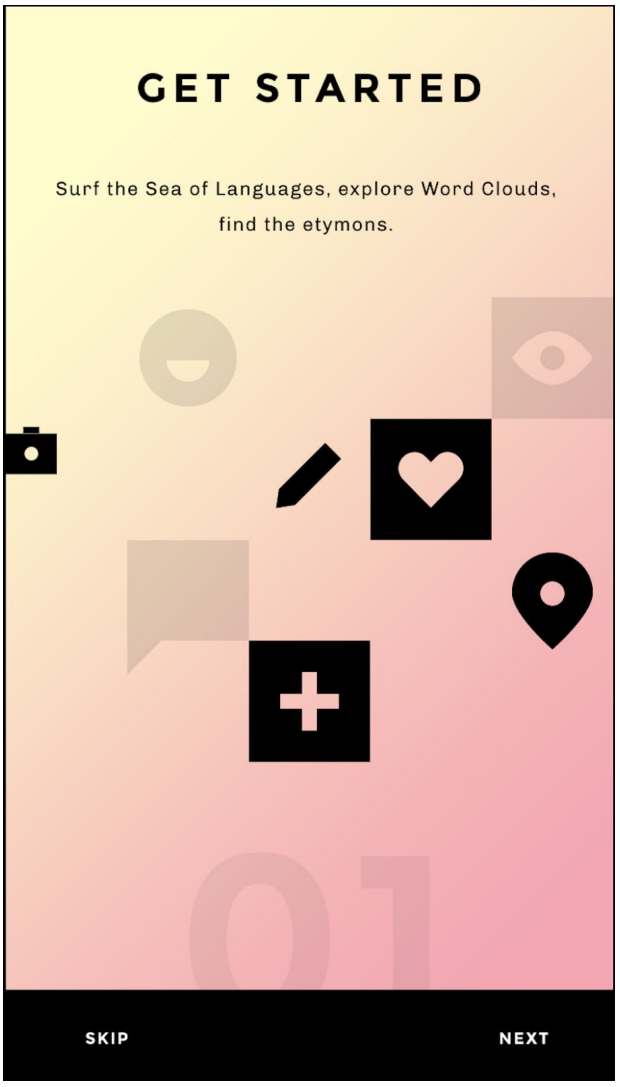

Figure 12 This figure represents the help panel Etymøn provides to the user. Slides of panels will help user to learn how to use the application efficiently.

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.

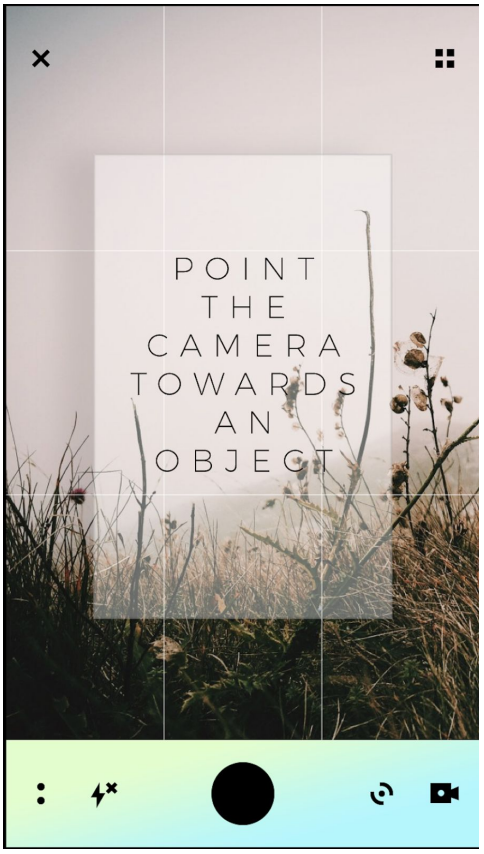

Figure *14* This figure represents the augmented reality functionality of Etymøn.

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.

Bilkent University

Department of Computer Engineering

Senior Design Project

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

High-Level Design Report

Nashiha Ahmed, Mert İnan, Cholpon Mambetova, Utku Uçkun

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

High-Level Design Report
Dec 24, 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

High Level Design 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 high-level design is also detailed.

1.1. Purpose of the System

Current etymological analyses rely on pattern matching or tracings between different languages by experts in linguistics [1], yet it may be cumbersome or even improbable to detect word origins in situations where direct links cannot be observed between two different words. In this case, Etymøn will pose an advantage as it will be using a large corpus of data in order to match words in any given language.

Various studies carried out by linguistic experts and historians improved the understanding of language and its origins [2]. However, there is still "room for improvement" in the field. Currently, most of the studies target the Proto-Indo-European language family [2]. There is sparse research done for other languages, and there is not a single, unified resource for this information. Most of the information is scattered online or among other forms of literature.

Since there is no similar project in the market yet, our software will be designed from scratch, which would make it a greenfield project. However, we will use other existing algorithms to build our software, such as deep learning algorithms among others that will be specified further in the report.

1.2. Design Goals

Design goals of the system in the areas of performance, dependability, maintenance and end-user criteria are described below/

Performance:

- Response Time: Etymon should be able to respond to user's queries within an unnoticeable delay. After finding the result of the user's query, it should be able to zoom into to the map smoothly and within unnoticeable delay.
- Throughput: As there can be multiple users of the system at the same time, Etymon should be able to respond to all of the search queries of all the users. The system should perform search and map traversal functions for each user at one time.
- Memory: Space required for Etymon will be entirely dependent on the size of the data. Machine learning model and codes of other modules will not occupy space as the data. Data to train should be compressible and it should not occupy space on the devices of the users, hence an online server is required to store all the training and testing data and the generated language sea.

Dependability:

- Robustness: Users may input invalid search words. Etymon should be able to survive invalid user inputs.
- Reliability: Etymon's machine learning components should behave as described. Differences between observed and specified behaviors of the system will be tolerated minimally.
- Availability: The system should be available all the time as the queries can be received at all times. Down times of servers should be minimal.

Maintenance:

- Extensibility: Etymon should be extensible in its machine learning, augmented reality and object recognition modules. As these modules can be updated in the community quite rapidly, the system should be able to adapt to changes in those modules and allow interfacing of new classes and models.
- Modifiability: Especially in terms of its machine learning algorithm, Etymon should be able to allow change. As better machine learning models can be developed, Etymon should be able to accommodate these changes and train the model again on the data.
- Readability: As the code of the system will be minimal yet math-intensive, the code should be readable.
- Traceability of requirements: It should be easy to map the code of specific subsystems to specific requirements.

End-user:

- Utility: Etymon should help the user to understand the connections between different languages and help the user's work in comprehending the connections between different words and their origins.
- Usability: It should be very simple for the user to traverse the system. Using Etymon should not require any tutorials or training.
- Understandability: Understanding the working mechanism of Etymon should be easy. Tracing the correctness of the results by the users should be possible.

1.3. Trade-Offs

Overfitting vs. Underfitting: During the training of the machine learning algorithm, overfitting will be favored against underfitting. As the training data is the only aspect that is important in this report, underfitting cannot be tolerated.

Precision vs. Accuracy: Even though both precision and accuracy of the results are favored, if a choice will be needed, then precision will be favored. As the accuracy of the mapped words cannot be checked for their correctness, precision among different language families is more important than the accuracy of classification of that word.

Delivery Time vs. Functionality: If testing and training runs behind the schedule, leaving out features such as augmented reality or hallucination may be possible, even though undesirable.

Model Complexity vs. Machine Type: As there are multiple machines with different efficiencies that can run machine learning algorithms, they will be put to use. Instead of increasing model complexity, in order to gain faster results, machine type will be changed to a faster one.

2. Proposed Software Architecture

2.1. Overview

In this section of the report, we will be analyzing some components of the high level architecture of Etymøn. Subsystem decomposition will be presented with its UML diagram in order to simply show the modules in the application domain to the problem.

2.2. Subsystem Decomposition

An ideal candidate architecture for our purposes and that fits the design of Etymøn is the client-server architecture. The user's machine will serve as client and will request and receive queries from a machine learning server. The server is our host computer where permanent data will be stored and processed, such as the etymological map, connections among words and languages; whereas the client side may perform some tasks such as animation, AR and word hallucination.

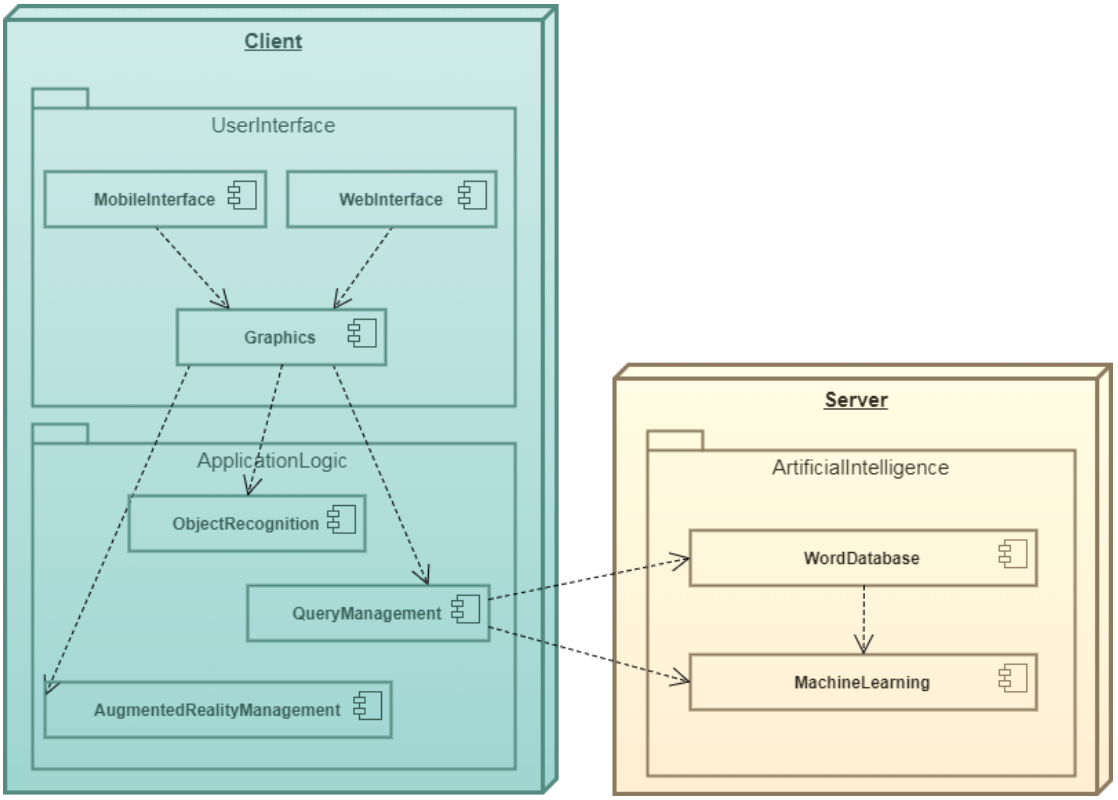

Figure 1: This figure shows the subsystem decomposition of Etymon and the architectural design.

2.3. Hardware/Software Mapping

Since we want Etymøn to work for all languages and deliver a fast word look-up service, it will require good hardware to run on. Storing words, their pronunciations and meanings for multiple languages will require great amount of memory space. Also, running deep learning algorithms on this graph and traversing it in a reasonable time will require computational power. Therefore, we must use a supercomputer to run our program in addition to our simple desktop computers. For this, we will use Google's Cloud Machine Learning Engine [3], TensorFlow [4]. Smartphones will also be required for the augmented reality and object recognition components of the project.

2.4. Persistent Data Management

Etymøn's effectiveness highly depends on its data management. It will find, store, manage and present information. For this reason we decided to use a robust database system which allow us to store excessive amount of data as quickly as possible. Etymøn will store many words from many different languages. For each word it will also store its meaning, pronunciation but also the data which will be used in deep learning process.

The word and pronunciation data will be acquired from Wikipedia. Furthermore, for English, pronunciation data from online sound libraries will also be used, and for other languages, International Phonetic Alphabet (IPA) readings of the words will replace sound files for clustering. To store meanings of the words, online dictionaries can be used. Using this data, we will create a Word Database.

2.5. Access Control and Security

Our software has no authentication method and can be accessed by all users. All users have the same privileges except for the admins. The end-users of Etymøn will not have direct access to word database of the program. They will be authorized to traverse the word information freely but will not be able to alter it. Administrative users will have access to graph modification tools. They will be able to alter the data of words, alter the relations formed by deep learning algorithm, add or remove words etc. The program will run on procedure-driven centralized control for the control of the software itself with triggers from the user, and the control will reside in controller objects in the program code.

2.6. Boundary Conditions

In the boundary conditions, the system will act according to the following.

- Initialization: When Etymøn is brought from a non-initialized state—such as the initial executable of the program—to the steady-state, the main processes will be done by the user interface subsystem. The user interface will display the splash screen during startup and then show the Language Sea.
- Termination: All cleanup procedures will be started by the program controller. No single subsystem is allowed to terminate on its own. Every subsystem is notified if a single subsystem is terminated. Read and write to the filesystem will be completed before termination.
- Failure: If a failure occurs during read and write to the Word Database or search, warning messages will be show. If failure occurs during getting the background images or component images, a restart of the software will be requested.

3. Subsystem Services

The services that the subsystems will provide are on the interfaces of client and server. As communication is necessary between them, information providers between modules such as etymological data will be sent through these services of the subsystems. These services are given in Figure 2.

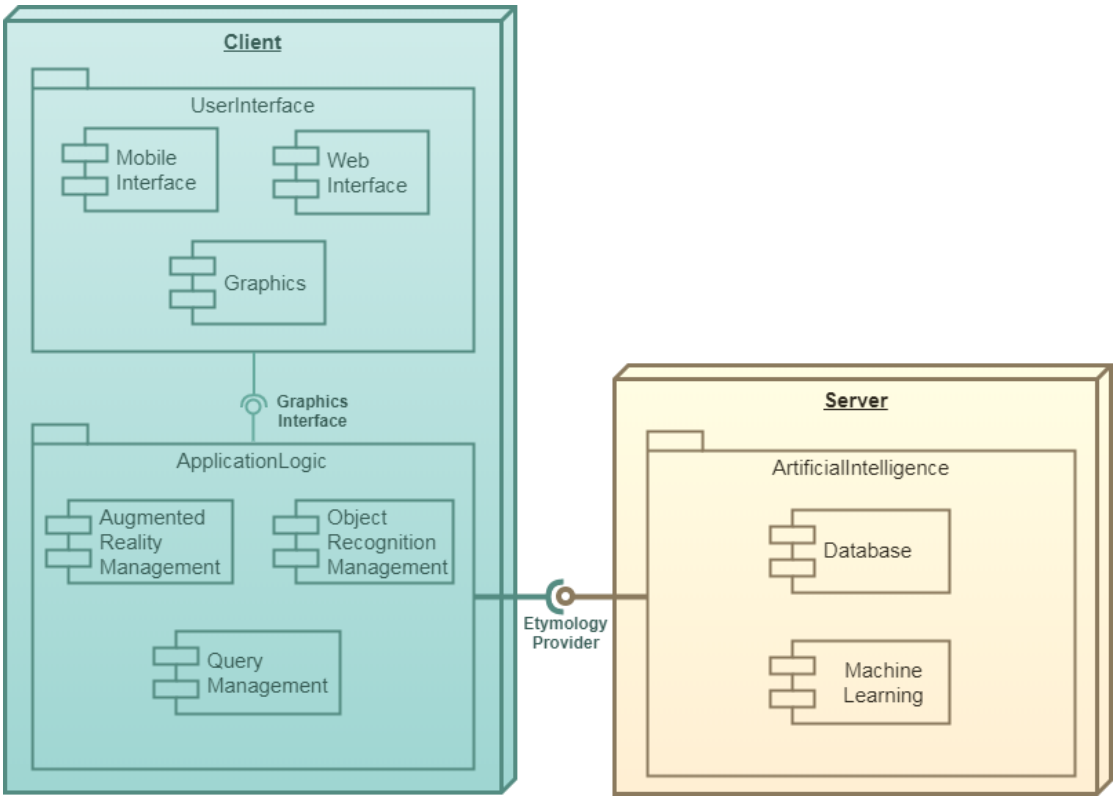

Figure 2: This figure shows the subsystem services of Etymon.

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] "Cloud ML Engine Overview | Cloud Machine Learning Engine (Cloud ML Engine) | Google Cloud Platform," Google. [Online]. Available: https://cloud.google.com/ml-engine/docs/technical-overview. [Accessed: 25-Dec-2017].

[4] "TensorFlow," TensorFlow. [Online]. Available: https://www.tensorflow.org/. [Accessed: 25-Dec-2017].

Bilkent University

Department of Computer Engineering

Senior Design Project

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

Low-Level Design Report

Nashiha Ahmed, Mert İnan, Cholpon Mambetova, Utku Uçkun

Supervisor: Prof. Mehmet Koyutürk
Jury Members: Prof. Uğur Doğrusöz and Prof. Çiğdem Gündüz Demir

Low-Level Design Report
February 12, 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

Low Level Design 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 low-level design is also detailed.

1.1. Purpose of the System

Current etymological analyses rely on pattern matching or tracings between different languages by experts in linguistics [1], yet it may be cumbersome or even improbable to detect word origins in situations where direct links cannot be observed between two different words. In this case, Etymøn will pose an advantage as it will be using a large corpus of data in order to match words in any given language.

Various studies carried out by linguistic experts and historians improved the understanding of language and its origins [2]. However, there is still "room for improvement" in the field. Currently, most of the studies target the Proto-Indo-European language family [2]. There is sparse research done for other languages, and there is not a single, unified resource for this information. Most of the information is scattered online or among other forms of literature.

Since there is no similar project in the market yet, our software will be designed from scratch, which would make it a greenfield project. However, we will use other existing algorithms to build our software, such as deep learning algorithms among others that will be specified further in the report.

1.2. Object Design Trade-Offs

Our system relies on working Augmented Reality software, Object Recognition software, and word databases online. We will be reusing existing packages and libraries, since our system is complicated and building these will take more time than is allotted to us to meet the project deliverable deadline. The trade-off is that it may not be completely compatible with our system, which may lead to data inaccuracies or incompleteness.

1.3. Interface Documentation Guidelines

Naming conventions will be used to make the development and design phases cohesive and understandable by all stakeholders in the projects. The following table will detail the naming and usage conventions that will be used.

Identifier Type	Rules for Naming	Examples
Packages	Package names are always written starting with a capital letter. If package names are a combination of multiple words, each word starts with a capital letter. Abbreviation letters are all capitalized. Package names are not lengthy but concise and meaningful.	package EtymonUI;
Classes	Class names follow the same naming conventions as package names. Abbreviations are only used when universally understood but are generally avoided to convey meaning clearly. Class names can be a combination of numerals and letters but not special characters.	class WordCloud;
Interfaces	Interface names follow the same naming conventions and class names.	interface LanguageSea;
Methods	Method names are verbs unlike package, class, and interface names. Method names start with a lowercase letter with first letter of internal words capitalized. Method names also need to be meaningful and concise and appropriate to understanding the method's purpose.	createCloud();
Variables	Variable names are also generally nouns. They start with lowercase letters, as methods and internal words start with an uppercase letter. Variable names may contain special characters but must not start with special characters. Variable names too must be meaningful but concise. Variables are all be initialized to avoid potential errors.	int wordCount; ArrayList<String> wordList;
Constants	Constants names have letters all capitalized and initialized. Internal words are separated by underscore character.	static final int MAX_WORD=100000;

Whitespace	In general, whitespace is used when needed to improve clarity of code. For example, the next line is used after the introduction of a branching statement. Spaces are used after an operator but can also be put before an operator.	for(int i = 0; i < 100; i++)
		{
		createWordCloud();
		}

Indentation	Indentation in our code is 4 spaces. If code is contained within a larger element, it is indented for code clarity.	
Brackets	Brackets are put on separate lines to help visual clarity on reading and writing code segments.	

1.4. Engineering Standards

The engineering standards we will be using are as follows:

- To model our software, Unified Modeling Language (UML) is used.
- IEEE engineering standards are used throughout the implementation and design of the project.
- The ACM Code of Ethics and Professional Conduct will also be enforced. Suggestions on the enforcement plan were given in the Project Specifications Report.

1.5. Definitions, Acronyms, and Abbreviations

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.

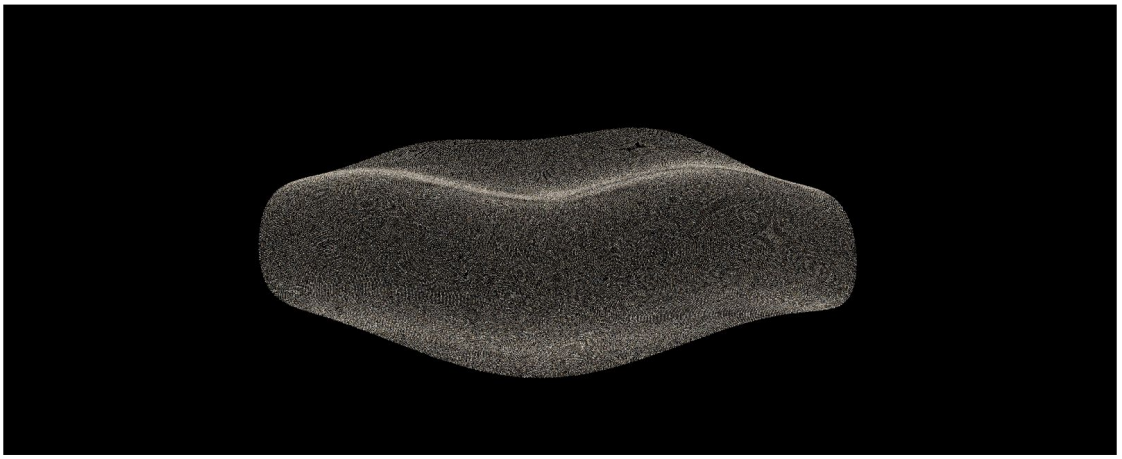

Figure 1 This figure depicts a wave-like pattern that will be like the Language Sea. [3]

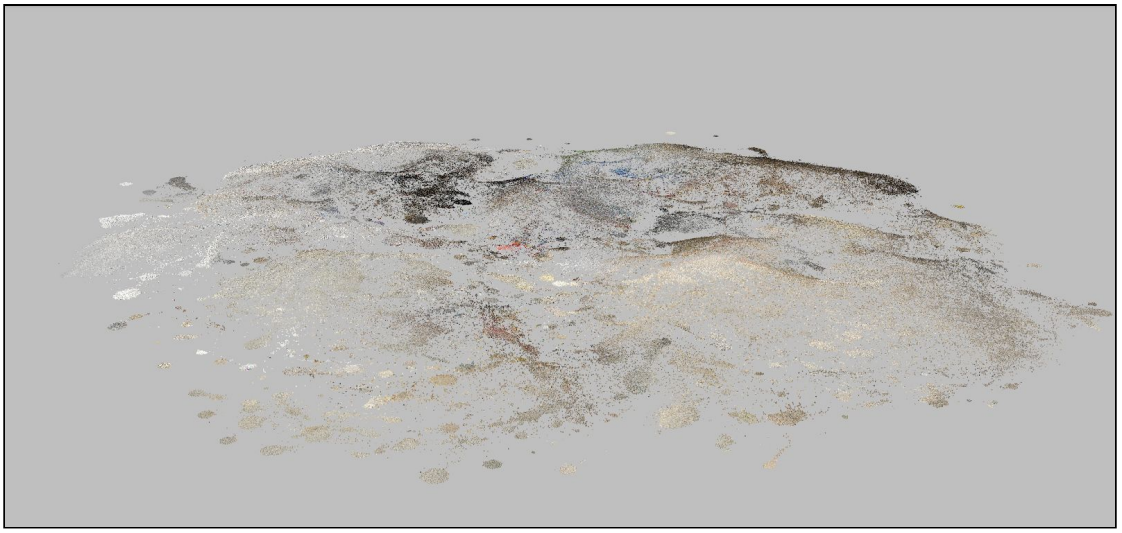

Figure 2 This figure is another clustered space that will be like the Language Sea. [3]

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

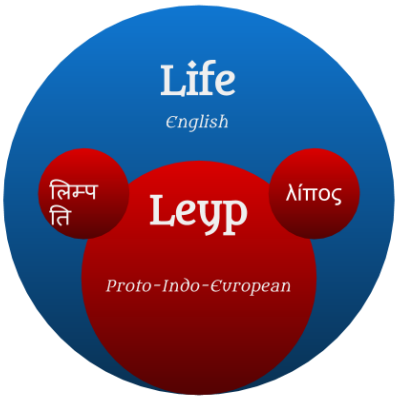

Figure 3 This figure shows a local graph for the English word, "life". Its origin is identified to be "leyp" in the Proto-Indo-European language family, and two descendent words —one in Sanskrit and one in Greek— are given next to the origin word.

2. Packages

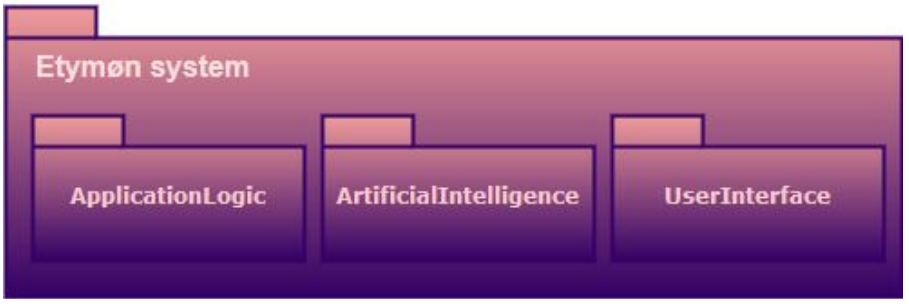

Figure 4 This figure is the UML diagram for the packages of the Etymon system.

User Interface package contains the WebGL and other graphics component codes. It includes specifically the mobile interface, web interface. Artificial Intelligence package represents the artificial intelligence portion of the whole system. It includes the Word Database and the Machine Learning related classes. Application Logic package represents the application logic of the Etymon system. It includes Augmented Reality Management, Query Management, and Object Recognition.

3. Class Interfaces

In this section of the report, we will be presenting the in-depth look to individual classes of the Etymon system. Their functions and attributes are described. Several design patterns are also described in the following paragraph. Final class diagram can be seen in Figure 5 and 6.

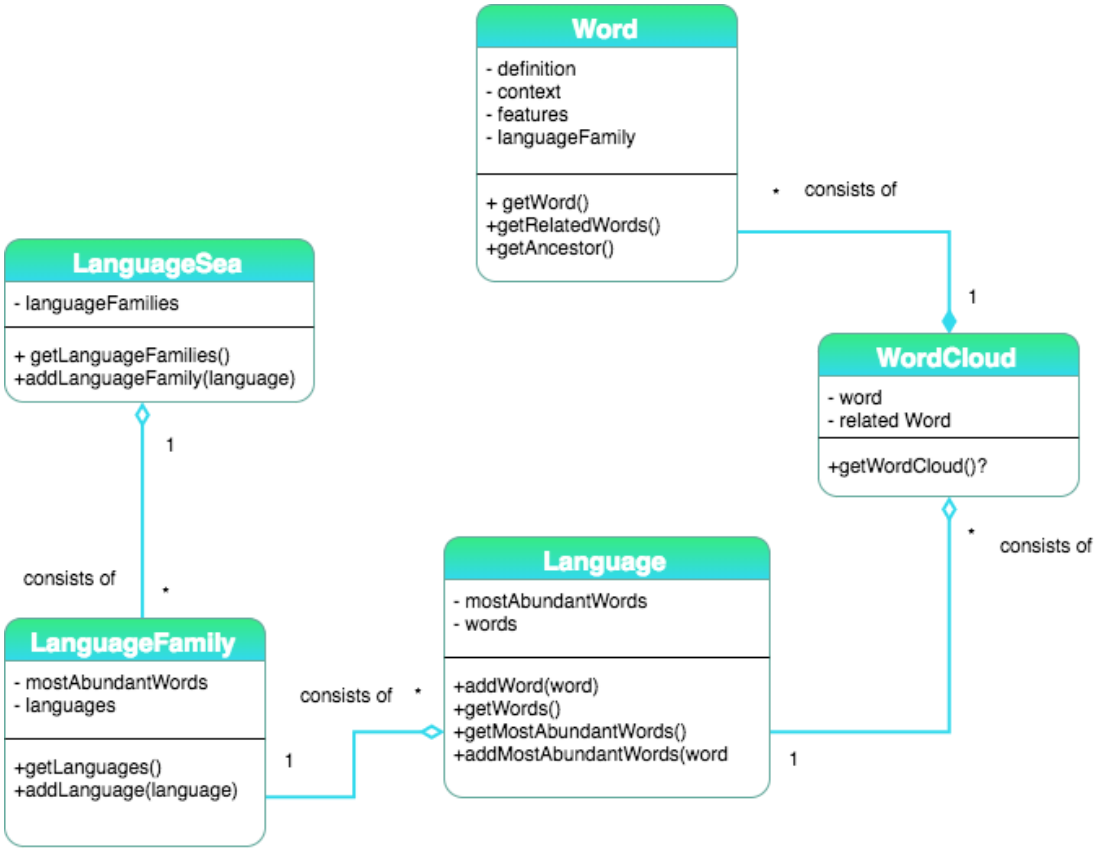

Figure 5 This figure shows the UML diagram for the entity classes of the Etymon system.

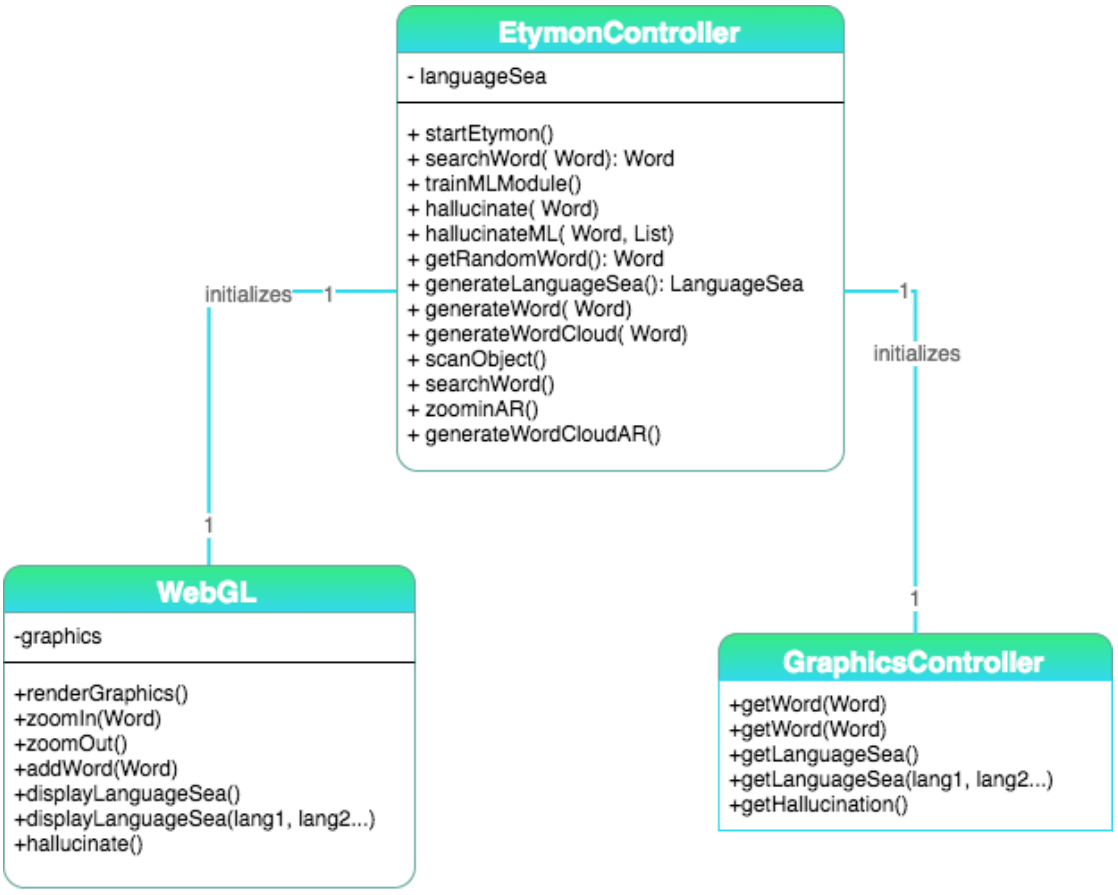

Figure 6 This is the UML class diagram representing the controller classes of Etymon.

Certain design patterns are chosen for objects of the Etymon. EtymonController and WebGL will be singleton classes as they will be initialized only once. LanguageSea object will follow an object pool pattern. Word class will be a prototype class and the GraphicsController will be an adapter class.

Class name	`LanguageFamily`
Attributes	`mostAbundantWords:` The most used words in this language family `languages:` Languages that are part of the specific language family.
Functions	`getLanguages():` Returns the languages within the specific language family. `addLanguage(newLanguage):` Adds a new language to the specific language family. `getMostAbundantWords():` Returns the most abundant words within the specific language family. `addMostAbundantWords(word):` Adds a new word to the most abundant words of specific language family.

Class name	Language
Attributes	words: all the words present in this language mostAbundantWords: The most used words in this language
Functions	addWord(word): Adds a new word to the specific language. getWords(): Returns the words within the specific language. getMostAbundantWords(): Returns the most abundant words within the specific language. addMostAbundantWords(word): Adds a new word to the most abundant words of specific language.

Class name	WordCloud
Attributes	word: The specific word that the word cloud is built upon. relatedWords: Related words to the specific word
Functions	getWordCloud(word): Returns the related words, ancestor and descendant information of a given word.

Class name	LanguageSea
Attributes	languageFamilies: The language families that are currently included in the language sea.
Functions	addLanguageFamily(language): Adds a new language to currently displayed language sea. getLanguageFamilies(): Returns the language families.

Class name	Word
Attributes	definition: Dictionary definition of the word. context: Context of the word. features: Certain features extracted from the word for machine learning purposes. languageFamily: Language family which the word is part of.
Functions	getWord(): Returns the word itself getRelatedWords(): Returns the related words for the specific word. getAncestor(): Returns the ancestor word of the specific word.

Class name	EtymonController
Attributes	languageSea: The language sea consists of all the languages and words that are connected to each other.
Functions	startEtymon(): Initiates the program, connects to databases searchWord(Word): Queries a word search in Etymon database and displays the results trainMLModule(): Initiates the clustering algorithm on database hallucinate(Word): Creates new words hallucinateML(Word, List): Runs the machine learning in hallucination mode, using generative LSTM. getRandomWord(): Chooses and displays a random word from the Etymon database. generateLanguageSea(lang1, lang2...): Generates and displays a language sea with the given languages. generateWord(Word): Runs the new word through the machine learning algorithm. generateWordCloud(Word): The newly added word is run through the ML algorithm to generate connection with other words and form a cloud. scanObject(): Scans the object using camera and returns its name. Will be used to search a word using the object recognition algorithm. zoominAR(): Zooms the image in AR. generateWordCloudAR(): Generates and shows the

	word cloud of a specific word as AR.

Class name	WebGL
Attributes	graphics: The actual graphics to display (language see, zoomed-in word cloud, etc.)
Functions	renderGraphics(): Draws the language sea on display. zoomIn(Word): Finds and zooms in the the queried word. zoomOut(): Display returns to the initial state. addWord(Word): User requests a new word to be added to the Etymon database. displayLanguageSea(): Displays the language sea. displayLanguageSea(lang1, lang2...): User chooses which languages will be displayed in the language sea. hallucinate(): generates the graphics for hallucination mode.

Class name	GraphicsController
Functions	getLanguageSea(): Gets the language sea from Etymon database to be used in WebGL. getLanguageSea(lang1, lang2...): Retrieves some of the languages that will be displayed in the language sea. getWord(word): Returns the word data on a scpecific word. getWordCloud(): Gets the word cloud data from database. getHallucination(): Retrieves the hallucination data.

4. Machine Learning Model Design

In this section four different algorithms that can be used as the machine learning components of Etymon will be discussed. Each of their advantages and disadvantages will also be analyzed. Logistic analyses of all the models is also included to decide on which algorithm to deliver in the end product.

4.1. Deep Learning Approach

Using Deep Neural Networks (DNN), ancestor words of current words from different languages of today can be found. This is a generative model, as it learns the training information and then generates a learning method by itself to be applied to other cases. In the end, this can create proto-words—ancestor words—can be created using Long Short-Term Memory (LSTM) models.

In the training part of the DNN, labels will be ancient words that are already known and the training set will consist of their descendant words in several different languages. All of the word data will be downloaded from wiktionary Proto-Indo-European word list [4].

The main advantage of this model is that it creates a simple implementation of the machine learning part. Furthermore, no manual feature extraction is necessary with this approach, as the words would be provided to the DNN and the output will be received after the training and testing periods. The main disadvantages are that the data classes are unbalanced and there is a small amount of data, which may reduce the accuracy of the DNN. Even more, the training period would be long for a DNN.

4.2. Graph Alignment Approach

Recent research in applied graph theory is focusing on graph alignments of different social media to identify missing components in another graph. This same concept can be used in identifying missing words in different languages and in identifying ancestor words.

Using wiktionary data for nearly five million words in the English section, graphs can be generated for multiple languages. Edges in the node will be connecting different words in the same language with which the current node has a link in the wiktionary page. Then the machine learning model can look at links between two graphs after alignment in order to see the word resemblances

Even though this model can identify the links between words of different languages, it does not fully create a proto-word, which would be a shortcoming as Etymon is also responsible for giving root information. Furthermore, the wiktionary data may not depict correct linguistic links in all of its word entries

4.3. Dynamic Programming Approach

In regular linguistic epigenetics, a method called comparative method is utilized to find the ancestry information of a language. This method is generally automated by using a dynamic programming strategy. This approach also borrows from the bioinformatics community by using the "edit distance" solution of dynamic

programming [5]. With this approach, two words in two languages can be compared at the same time with one another. One language can also be a proto-language. Hence, a word's edit distance can be found from its proto-word to a real word in today's languages. Using this strategy, we will be generating multitudes of dynamic programming tables. Hence, we can use the patterns found in these tables as machine learning features.

One drawback of this approach can be that it can only work with borrowed words in different languages, or it can favor borrowed words than any other word as the characters will have very small edit distances. Furthermore, every new word would be needed to be compared with every other word in the other language.

4.4. Word Embeddings Approach

Like with the graph alignment strategy, each word can be clustered according to their meanings and can be transformed into a vector using a tool like word2vec [6]. This would create a hyperdimensional space, and these spaces can be created for two different languages. In the end, the differences between these two spaces can be used to predict the ancestor words.

The main advantage of this method is that it successfully transforms words into vectors and multilingual forms of the word2vec can be employed.

All of these approaches are possible, yet doing them all at the same time would require a lot of time that can exceed the delivery time of the project. As a result, currently the most feasible approach—pragmatic logistics-wise—is the deep learning approach.

5. References

[1] E. P. Hamp and J. Lyons, "Linguistics," Encyclopædia Britannica, 10-Mar-2017. [Online]. Available: https://www.britannica.com/science/linguistics/The-comparative-method. [Accessed: 12-Feb-2018].

[2] "About," Ethnologue. [Online]. Available: https://www.ethnologue.com/about. [Accessed: 12-Feb-2018].

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

[4] "Appendix:List of Proto-Indo-European roots," Appendix:List of Proto-Indo-European roots - Wiktionary. [Online]. Available: https://en.wiktionary.org/wiki/Appendix:List_of_Proto-Indo-European_roots. [Accessed: 12-Feb-2018].

[5] M. P. Oakes, "Computer Estimation of Vocabulary in a Protolanguage from Word Lists in Four Daughter Languages," Journal of Quantitative Linguistics, vol. 7, no. 3, pp. 233–243, Jan. 2000.

[6] Dav, "dav/word2vec," GitHub, 18-Jan-2018. [Online]. Available: https://github.com/dav/word2vec.git. [Accessed: 12-Feb-2018].

Bilkent University

Department of Computer Engineering

Senior Design Project

Etymøn: A Data Visualization & Deep Learning Application for Etymological Clustering of Words

Final Report

Nashiha Ahmed, Mert İnan, Cholpon Mambetova, Utku Uçkun

Supervisor: Prof. Mehmet Koyutürk
Jury Members: Prof. Uğur Doğrusöz and Prof. Çiğdem Gündüz Demir

Final Report
May 3, 2018

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

Final Design Report

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

1. Introduction

Etymøn is an analysis and tracing tool for word origins in all languages. It is also a data visualization application that demonstrates network graphs of links between words in different languages. Etymøn can be considered as an interactive art experiment as well, with its aesthetic visuals. It also contains a "hallucination" section, where a Long Short-Term Memory (LSTM) neural network generates new connections for given words.

2. Final Architecture and Design

As mentioned in the previous reports, Etymøn uses client-server architecture. However, instead of the object-oriented class structure, modules based on the functionalities of the html structures are created. Figure 1 shows the UML diagram of the server side and Figure 2 shows the package structure of Etymøn client side.

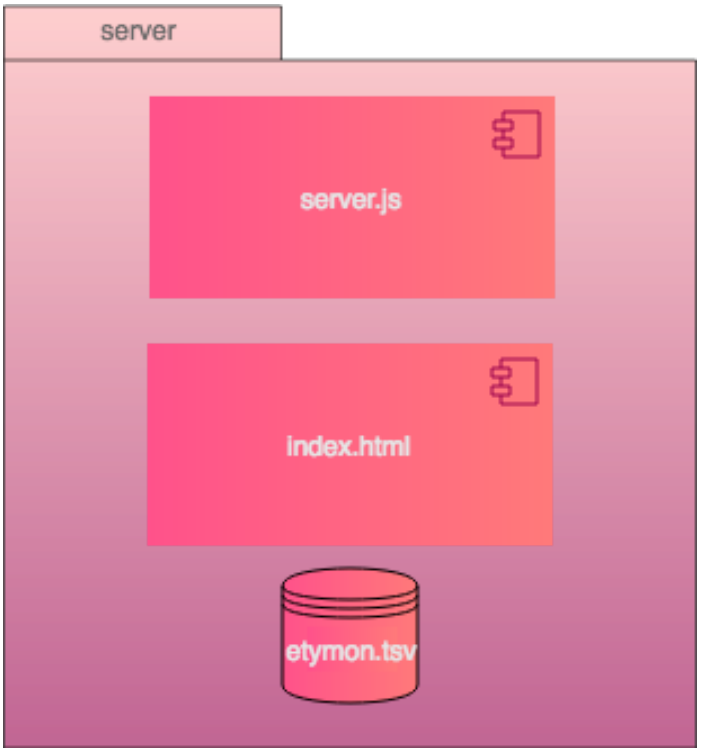

Figure 1 This figure shows the components of the server side of Etymøn.

Figure 2 This is the UML diagram for the components and packages of the client side of Etymøn.

These UML diagrams show main javascript files of the packages. Contents of these files are available in the GitHub repository of Etymøn.

3. Final Status

Etymøn functionally accomplishes all the requirements that were mentioned in the project specifications for search and hallucination parts. It does not contain the object recognition module yet, as it was deemed to be a low-priority sub-functionality of the project. Search functionality searches for an enquiry and demonstrates the word cloud (graph of etymologically-connected words) using cytoscape.js. Hallucination functionality uses LSTM to generate a theoretical word cloud for the inquiry. Screens of Etymøn can be seen in the following figures. Etymøn also strives for a good aesthetic experience for the user, hence it accomplished that in addition to the previous project specifications and it is still possible to improve the aesthetics of it.

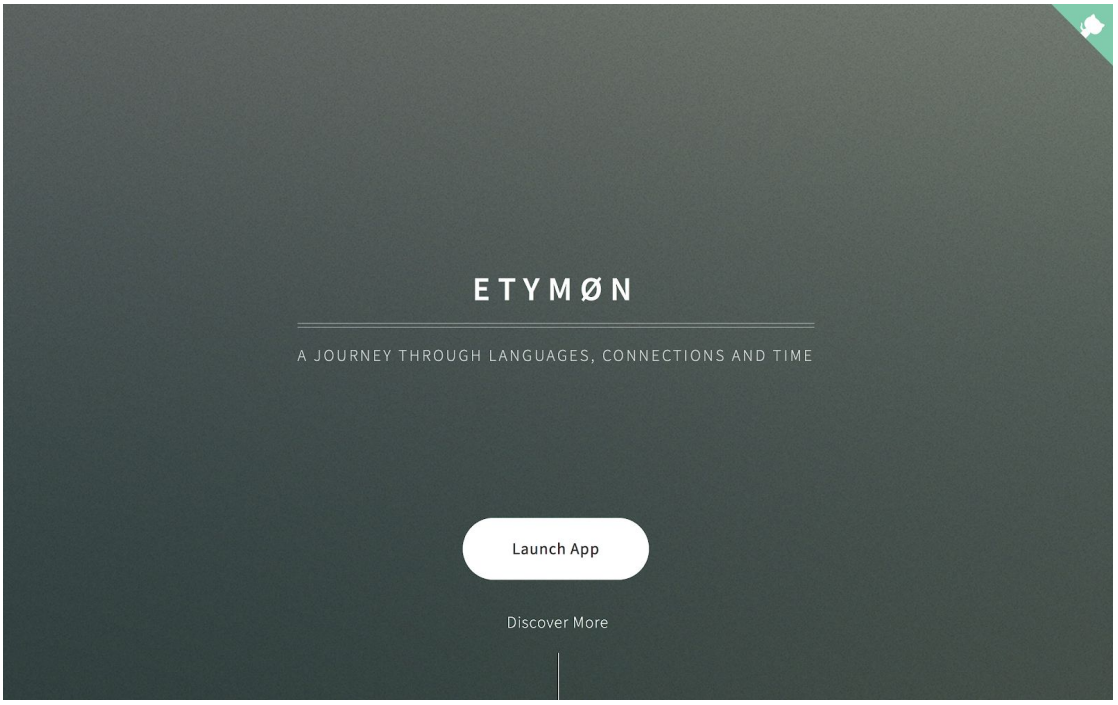

Figure 3 This figure is the landing screen of etymon.org.

Figure 4 This is the main screen of the Etymøn app, where the user is greeted by the moving language sea in the background.

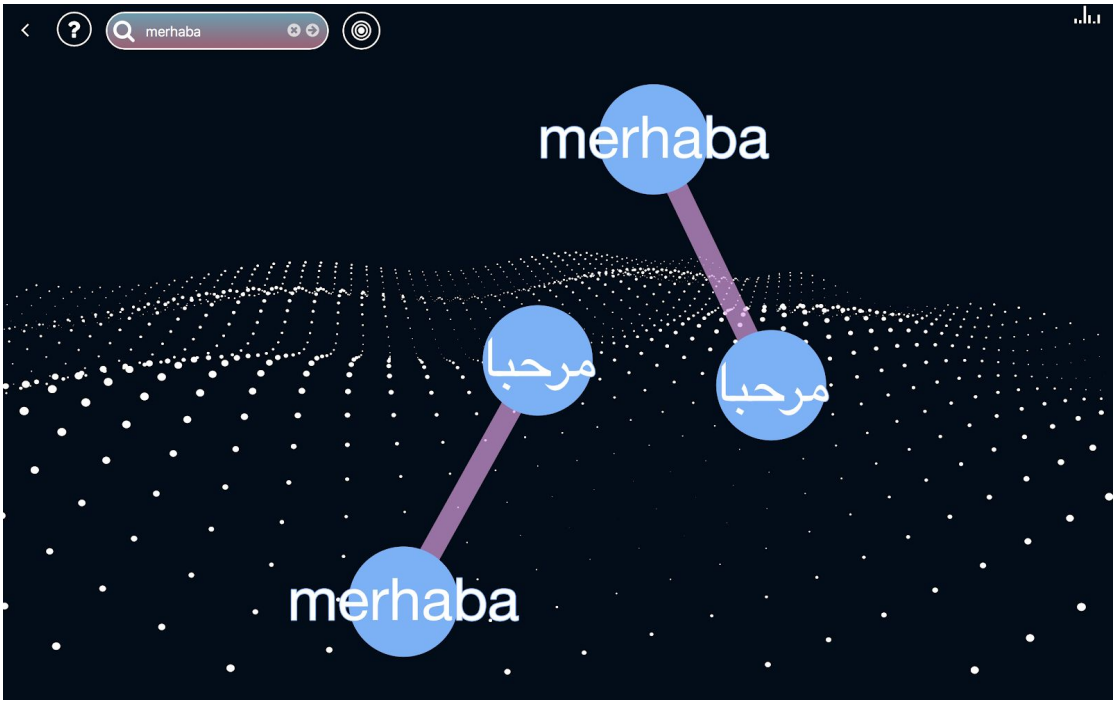

Figure 5 This figure shows a word cloud of the word “Merhaba” in Turkish when a user searches the word merhaba.

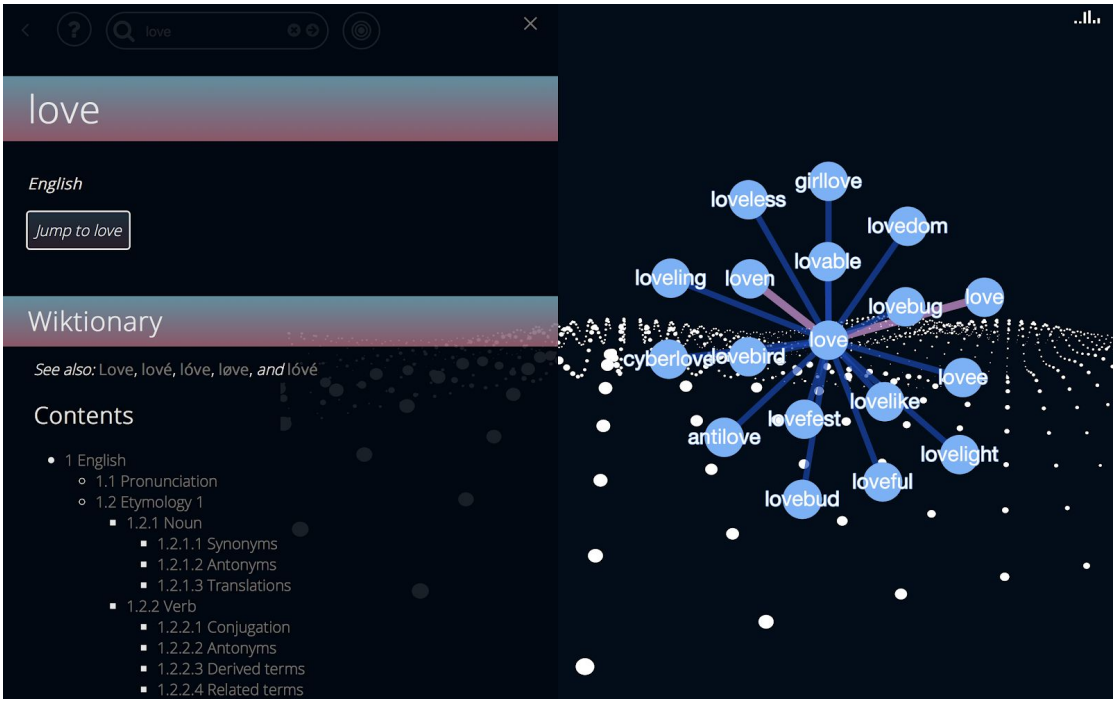

Figure 6 This figure shows the Wiktionary information of a word when it is clicked. It is located on the left side of the page in form of a panel. The user can jump to the word "love" by clicking the button on the top of the panel "Jump to love."

4. Impact of Engineering Solutions

4.1. Dataset

One of the functionalities Etymøn provides to the user is collecting and presenting already existing etymological information from web. We did not choose to do this lookup operation dynamic because we were not sure whether these websites would be available in the future. Thus, we downloaded and stored all the etymology related information from these websites. After parsing and extracting the useful informations from these files we decided to store this information in text format. We knew we had to store lots of words and relations between them and storing in text format was the most space efficient solution.

We also kept the dataset and backend logic as independent as possible. This way altering dataset become very easy. Etymology domain is very dynamic and we wanted to make Etymøn flexible.

4.2. Three.js and Cytoscape

Another functionalities Etymøn provides is allowing user to choose word relations in either cytoscape or three.js. We decided to have these options, since three.js despite looking beautiful is demanding. For those who are using Etymøn for research purposes, it may be more convenient to use Cytoscape, which is less demanding on the machine running Etymøn.

5. Contemporary Issues

5.1. Machine Learning

Currently Etymøn uses an LSTM model to create the graph itself. However, it may be useful to generate the word cloud based on specific words instead of the graphs. As, machine learning is of contemporary importance, it would be appropriate for Etymøn to use a different model, and improve its usefulness in the etymological origin-finding process. This can be accomplished by showing the confidence levels for each generated word when another model is used.

5.2. Intellectual Property and Legal Issues

Each team member and the system itself was responsible with the data or code that they used in the project and gave credit in order to protect intellectual property rights of the users or experts in linguistics. The data we retrieved from websites were attributed and used for academic purposes. We also have not monetized our software, therefore there can be no legal disputes.

5.3. Natural Language Processing

Even though Etymøn uses already present etymological information, natural language processing is relevant in several aspects. First of all, while scraping data from websites, natural language processing can be employed to better extract information from the websites. Secondly, it would prove to be useful in generation of hallucinated origin words in the LSTM, as certain key features of words can be extracted using natural language processing and used in machine learning.

5.4. Linguistics

Relations between languages and language families are changing with new archaeological discoveries. As new literature pieces, alphabets and cultures are unearthed etymological origins of languages and words become open to reinterpretations. Therefore, there can be variations between etymological origin information of the same word among different sources. We experienced same issue in our application as well. Some of the sources we used had contradicting results and we are aware that further discoveries may nullify some of our work. On the other hand, Etymøn dataset is easy to change and these new discoveries can be integrated without much effort.

6. New Tools & Technologies Used

6.1. Cytoscape.js

Main data visualization of the network graphs is undertaken by cytoscape.js. As cytoscape.js is a client-side application that renders graphs using source and target nodes and edges, it is used to demonstrate the word clouds.

Different colors of edges are used for different languages. Edge and node information is received from the server side and processed inside the client-side javascript code.

Cytoscape has the flexibility of adapting to the inputs. It also contains algorithms to perfectly organize a given graph. CoSE Layout is used to create nodes which have the searched word in the center and the relations around the word arranged

in a circle. This enables the user to clearly see the relations based on their connections to the searched word.

6.2. Three.js

As Etymøn is also an art experience, quality animations are necessary to attract users' attention. This is accomplished by the use of client-side 3D rendering javascript library called three.js.

Three.js allows smooth movement of particles. This idea and several examples on the three.js documentation website enabled Etymøn to have a language sea. This sea contains white particles that are aesthetically moving in waves when Etymøn is first opened.

Full transition from cytoscape.js to three.js is envisioned as such a transition would improve the aesthetic quality and smoothness of Etymøn. Yet, due to heavy processing load of particles in three.js, this transition may not preferred, as certain users may use Etymøn just to search for etymological purposes rather than aesthetic experiences.

6.3. Node.js

Node.js is the javascript library for server-side setup. This library is used to deploy the application on Google Cloud. It is also employed to test Etymøn locally on the computers for its projected behavior on the server.

Certain libraries of node.js were necessary during the production of the server. These include python-virtualenv, express and body-parser. Python virtual environment package was used for machine learning code. Express and body-parser were used to set up http requests.

6.4. Express.js

Express.js is a framework that handles http requests such as POST, SEND and GET. Etymøn's server uses two different POST requests: one for regular word search and another for machine learning inquiries. It is used in company with node.js.

6.5. Keras

Keras is an abstraction library of neural networks on the tensorflow library. It is used in Etymøn to create a sequential model of a neural network. The model contains an LSTM layer. Ease of use of Keras makes it suitable for small applications of neural networks, hence it was the best choice to be used in the project.

Keras is used inside a python virtual environment in combination with tensorflow. This integration was necessary to be used in Google Cloud platform. LSTM model uses the tab-separated file as input to produce etymological connections.

6.6. Tensorflow

Tensorflow is an open-source machine learning library developed by Google. It is used in the machine learning part of Etymøn. Keras is dependent on tensorflow

library to work. Hence, it is used to create the LSTM code for hallucinating new connections.

H5Py is used in combination with tensorflow to output the weight vectors of the LSTM model.

6.7. Google Cloud Console

Google Cloud Console is the location of the server of Etymøn. It holds the code that searches the enquiry through the dataset using bash commands and node.js. Google Cloud Console provides resources to analyze the activity of the server and has tutorials on how to get started easily, hence, it was chosen as the main node.js server. Free credits were used for this application.

6.8. Google Machine Learning Engine API

Google Machine Learning Engine API was used to train the LSTM model on the cloud, using faster processors. Google provides multiple sets of processors of arbitrary capacity to be used on machine learning model training. As the original input to the Etymøn machine learning module contains more than 300,000,000 characters, it is computationally expensive to run the model. However, as Google does not provide high end processors for free-credit users, it was decided afterwards to rely on subsets of the input data to be trained on local computers.

7. Use of Resources

Mainly online resources were used to cover multitudes of areas, ranging from website development to machine learning model creation. These resources were vital in understanding and developing on different platforms.

7.1. Resources for the Website

Main resources used for Etymøn's website are documentations about HTML, CSS and javascript. W3Schools and CodePen were appropriate resources on this matter. Three.js official website was also used for documentation and examples. The resources are described as follows:

- GitHub corners code that links to the GitHub repository of the code and has an animation of a cat [1].
- HTML5Up template for the HTML responsive website design [2].
- Sample code for the audio icon with bar animation [3].
- Sample code for filling animation of search and hallucination buttons on hover [4].
- Sample code for search button and hallucination button animation of enlarging on click [5].
- Resource about creating navigation bars that pop up from the side of the window [6].
- Documentation and examples of Three.js [7].

7.2. Resources for the Server

Server side code required javascript knowledge that was new. Hence, the following resources were useful in order to understand server architectures and creating http requests.

- Resource on executing unix commands in node.js [8].

- Tutorial on node.js server setup [9].
- Server sample code using node.js and express.js [10].
- Express.js documentation [11].
- Google Cloud Console quickstart with node.js [12].

7.3. Resources for Etymological Information

Etymological information was present on a diverse set of websites online such as etymonline.com and nisanyansozluk.com. These information had to be scraped from HTML. Here are the resources that were used while scraping:

- Nisanyan Sozluk Turkish etymology resource [13].
- Etymønline English etymology resource [14].
- Etymological WordNet resource [15].
- Wiktionary Resource to display etymological and meaning information [16].

7.4. Resources for Animation and Data Visualization

Animation and data visualization is also done in javascript in order to increase compatibility with the client-side code. Hence, javascript solutions such as cytoscape.js and three.js had to be learnt. The following resources were used while implementing in those libraries:

- Waves animation resource [17].
- Cytoscape.js documentation [18].
- Cytoscape.js twitter tutorial [19].

7.5. Resources for Machine Learning

Neural network creation is done using Keras and tensorflow. As these libraries have their own specific function uses, documentation and tutorials of them were useful. Furthermore, tutorials on implementing a whole LSTM network were used to create the model.

- Training LSTM Networks [20].
- Keras documentation [21].
- Tensorflow documentation [22].
- Google Cloud Machine Learning Engine documentation [23].

7.6. Miscellaneous Resources

These resources were of general help while developing python code and certain other API requests.

- Sample code on looking up a word using Wiktionary API [24].
- Python BeautifulSoup Documentation [25].
- Python regular expressions documentation [26].

8. User's Manual

8.1. General Information

8.1.1. System Overview

Etymøn is an analysis and tracing tool for word origins in all languages. It is also a data visualization application that demonstrates network graphs of links between words in different languages. Etymøn can be considered as an interactive art experiment as well, with its aesthetic visuals. It also contains a "hallucination" section, where a Long Short-Term Memory (LSTM) neural network generates new connections for given words.

Etymøn is currently available on web-browsers, such as Mozilla Firefox, Internet Explorer, Safari, and Google Chrome. Etymøn is also compatible with mobile phones and can be accessed through the phone's web browser.

8.1.2. Points of Contact

The points of contact (POCs) that may be needed for any troubleshooting, information, or assistance purposes are the creators of the website and team members of the Etymøn team. They are as follows:

- Mert Inan: mert.inan@ug.bilkent.edu.tr
- Utku Uckun: utku.uckun@ug.bilkent.edu.tr
- Nashina Ahmed: nashiha.ahmed@ug.bilkent.edu.tr
- Cholpon Mambetova: cholpon.mambetova@ug.bilkent.edu.tr

8.1.3. Definitions, Acronyms and Abbreviations

Some definitions of the 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.
- The *Word Cloud* is a local graph for words clustered close to one each other.
- *Word Hallucinations* are brand-new words created from another word using neural networks.
- *App* denotes Application
- *LSTM* denotres Long Short-Term Memory neural network

8.2. System Summary

As aforementioned, Etymøn allows users to search for the etymology of any word in all languages. It shows this etymology in a network diagram. In addition, Etymøn provides word definitions and explanations through Wiktionary. Etymøn also creates word hallucinations when a user enters a source word. The user can also view the different relationships between words in a network diagram explained in further sections.

8.2.1. System Configuration

Etymøn has platform support for all web browsers on desktop and mobile devices. It features one main screen for the app that can be accessed through the website etymon.org by launching the app. Users can search for the etymology of a word by clicking the search button. They can create word hallucinations by using the hallucinate button. Finally, users can view instructions by clicking the help button. Lastly, users can turn background music on or off by clicking the music icon on the top right hand corner of the screen. Users can zoom in and out of the animation and navigate the 3D animation using their cursor. Users can also move the nodes of the network diagram of an etymology word cloud using their cursor. They can jump to another word by clicking the 'Jump to Word" button in the Wiktionary information of a word.

8.2.2. Data Flows

Users can input a word using their keyboards. The word can be changed after it has been typed by clicking the search or hallucinate fields. The word cannot be changed after it has been submitted, but a new word can be entered into the search or hallucinate fields.

8.2.3. User Access Levels

All users can only view information but cannot directly add or edit new information to the system.

8.3. Getting Started

8.3.1. Accessing the Page

The app can be accessed through the website etymon.org by clicking the button "Launch App." Alternative, the app can be accessed via the following link: http://etymon.org/website/etymon/index.html. The app can be accessed from all web browsers on phone and desktop.

8.3.2. System Menu

8.3.3. Searching a Word

After a user has entered the app, the user can utilize the functionality of searching the etymology of a word by clicking on and entering a word into the search field by clicking the search button which expands on click.

The word can then be searched by using the enter button on the keyboard or clicking the right arrow button on the screen using a cursor. The word can be deleted or edited by using the backspace on keyboard or clicking on the cross button on the search bar.

- **The Word Cloud**

  When a word is searched and a network diagram, the word cloud, denoting the etymology of the word appears, some edges may be colored according to the relationships between words. The searched word is in the center of the cloud. Pink edges denote the source etymology of the word. Blue edges denote words that come from the searched word. Therefore the searched word (in the center of the word cloud) is the etymology of the word at the end of a blue edge. Green edges denote etymologically related words. Light blue edges denote similar words.

  When a word from a word cloud is selected the panel with information on the word appears on the left side of the screen.

- **Word Definitions**

  The panel shows the word definition that is real time scrapped from Wiktionary website. The definition is very detailed and shows all possible ways of pronunciation, definitions with example, etc. that is available on Wiktionary.

- **Jumping to another Word**

  After the Wiktionary information of a word on the left side of the screen appears via a panel, the user can jump to that word so that that word is searched directly on Etymøn and its word cloud will appear. The user can do this by clicking the "Jump to [selected word]" button in the top of the panel.

8.3.4. Creating a Word Hallucination

The creation of a word hallucination is a similar process to the searching of a word. The hallucinate button denoted with an icon that contains

concentric circles can be clicked. Upon clicking the button, the hallucinate bar expands so the user can enter a word to hallucinate. The word can be deleted by using the cross button in the hallucinate bar or by using backspace on the keyboard. The word can be entered to hallucinate by pressing the enter button on the keyboard or clicking the right arrow in the hallucinate bar.

8.3.5. Background Music

The background music can be turned off and on using the button on the top right hand corner of the screen. When the bars are animating, the music is on. When the bars are still, the music is off. The music is on by default.

8.3.6. Exiting the System

The system can be exited by simply closing the window or tab on which Etymøn is running.

9. References

[1] Tholman, "tholman/github-corners," GitHub, 26-Sep-2017. [Online]. Available: https://github.com/tholman/github-corners. [Accessed: 03-May-2018].

[2] "HTML5 UP," HTML5 UP. [Online]. Available: https://html5up.net/. [Accessed: 03-May-2018].

[3] "Audio Icon," CodePen. [Online]. Available: https://codepen.io/joannaong/pen/olkzh. [Accessed: 03-May-2018].

[4] "Button Fill Animation," CodePen. [Online]. Available: https://codepen.io/davekilljoy/pen/wHAvb. [Accessed: 03-May-2018].

[5] "Awesome Search Button with Input Animation," CodePen. [Online]. Available: https://codepen.io/ey_intuitive/pen/vlcgf. [Accessed: 03-May-2018].

[6] "How TO - Side Navigation," How To Create a Side Navigation Menu. [Online]. Available: https://www.w3schools.com/howto/howto_js_sidenav.asp. [Accessed: 03-May-2018].

[7] "three.js," Three.js official website. [Online]. Available: https://threejs.org. [Accessed: 03-May-2018].

[8] L. Pollard, "Execute A Unix Command With Node.js - DZone Web Dev," dzone.com, 15-Aug-2010. [Online]. Available: https://dzone.com/articles/execute-unix-command-nodejs. [Accessed: 03-May-2018].

[9] "Build a simple Weather App with Node.js in just 16 lines of code," codeburst, 20-Jun-2017. [Online]. Available: https://codeburst.io/build-a-simple-weather-app-with-node-js-in-just-16-lines-of-code-32261690901d. [Accessed: 03-May-2018].

[10] "Build a Weather Website in 30 minutes with Node.js Express OpenWeather," codeburst, 26-Jun-2017. [Online]. Available: https://codeburst.io/build-a-weather-website-in-30-minutes-with-node-js-express-openweather-a317f904897b. [Accessed: 03-May-2018].

[11] "Express - Node.js web application framework," Express - Node.js web application framework. [Online]. Available: https://expressjs.com/. [Accessed: 03-May-2018].

[12] "Quickstart for Node.js in the App Engine Flexible Environment | Node.js | Google Cloud," Google. [Online]. Available: https://cloud.google.com/nodejs/getting-started/hello-world. [Accessed: 03-May-2018].

[13] "Nişanyan - Türkçe Etimolojik Sözlük," Nişanyan - Türkçe Etimolojik Sözlük. [Online]. Available: http://www.nisanyansozluk.com/. [Accessed: 03-May-2018].

[14] "Online Etymology Dictionary," Online Etymology Dictionary. [Online]. Available: https://www.etymonline.com/. [Accessed: 03-May-2018].

[15] Etymological Wordnet. [Online]. Available: http://www1.icsi.berkeley.edu/~demelo/etymwn/. [Accessed: 03-May-2018].

[16] "Wiktionary," Wiktionary. [Online]. Available: http://wiktionary.org/. [Accessed: 03-May-2018].

[17] Mrdoob, "three.js," GitHub. [Online]. Available: https://github.com/mrdoob/three.js/blob/master/examples/canvas_particles_waves.html. [Accessed: 03-May-2018].

[18] M. Franz, "Cytoscape.js," Cytoscape.js. [Online]. Available: http://js.cytoscape.org/. [Accessed: 03-May-2018].

[19] "Graphing a social network," Graphing a social network · Cytoscape.js. [Online]. Available: http://blog.js.cytoscape.org/2016/07/04/social-network/. [Accessed: 03-May-2018].

[20] "Text Generation With LSTM Recurrent Neural Networks in Python with Keras," Machine Learning Mastery, 08-Jan-2018. [Online]. Available: https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/. [Accessed: 03-May-2018].

[21] "Keras: The Python Deep Learning library," Keras Documentation. [Online]. Available: https://keras.io/. [Accessed: 03-May-2018].

[22] "Installing TensorFlow | TensorFlow," TensorFlow. [Online]. Available: https://www.tensorflow.org/install/. [Accessed: 03-May-2018].
[23] "Predictive Analytics - Cloud Machine Learning Engine | Google Cloud," Google. [Online]. Available: https://cloud.google.com/ml-engine/. [Accessed: 03-May-2018].
[24] Nichtich, "Look up a word in Wiktionary via MediaWiki API and show the Wiktionary page," Gist. [Online]. Available: https://gist.github.com/nichtich/674522. [Accessed: 03-May-2018].
[25] "Beautiful Soup Documentation," Beautiful Soup Documentation - Beautiful Soup 4.4.0 documentation. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Accessed: 03-May-2018].
[26] "7.2. re - Regular expression operations," 7.2. re - Regular expression operations - Python 2.7.15 documentation. [Online]. Available: https://docs.python.org/2/library/re.html. [Accessed: 03-May-2018].