Utrecht University

MSc thesis in Game and Media Technology
Utrecht University - Faculty of Science

Improving open text matching in a communication serious game

First supervisor:
Dr. Raja Lala

Second supervisor:
Prof. dr. Johan Jeuring

Author:
Andrei Ungureanu
7003218
a.ungureanu@students.uu.nl

November 22, 2022

Abstract

I analyze and implement state-of-the-art natural language processing models for open text understanding to improve the matching of open text input in a serious game that uses custom scenarios created for training communication skills, called Communicate. Previous work in matching open text input in this serious game used a scenario specific corpus, a corpus containing all the words used in the particular scenario, to match open text input to a scripted statement. This scenario-specific corpus contains mathematical representations for each word appearing in the scenario. The goal of this thesis is to expand on this previous work by exploring state-of-the-art word embeddings and implementing relevant models that use scenario specific information to try to improve the open text matching process.

Acknowledgements

I would like to thank my first supervisor, Dr. Raja Lala, without the help of who I would have never finished this Master's thesis. His endless patience with my unorganized style of work, and feedback helped me push through very confusing times and ultimately finish the project. It was quite disappointing not to be able to meet more often in person, however, the wisdom of his advice was transmitted digitally quite well.

I would also like to thank my second supervisor, Prof. dr. Johan Jeuring who gave me the opportunity to work on this project in the first place. In an uncertain time during my Master's he offered me this project to work on despite my limited experience in the field of NLP.

Many thanks go out to my friends back in Romania as well as the ones I made in the Netherlands during my studies, for the company and advice throughout the lockdown and after. Special thanks to my good friend and neighbor Marco Pellegrino for all the quality papers he offered me to understand how a real thesis should be written.

Last but not least, I would like to thank my parents for their support and patience in this journey of mine in figuring things out.

Table of contents

1 Introduction **4**

2 Related work **9**

3 Research Approach **11**

4 RQ1: Word embedding models **12**
- 4.1 RQ1 Method 12
- 4.2 RQ1 Results 12

5 RQ2: Testing word embedding models on the golden dataset **20**
- 5.1 RQ2 Method 21
- 5.2 RQ2 Results 22

6 RQ3: Implementation of the matching service **23**
- 6.1 RQ3 method 23
- 6.2 RQ3 Results 26

7 Discussion **29**

8 Conclusion **30**

1 Introduction

A serious game is an application used for training, education, or research and has other purposes than only entertainment. Alvarez et al. [10] define serious games as applications created with the intention of combining aspects of teaching, learning, and communication with entertaining elements found in modern video games. Serious games have a larger scope than simple entertainment, becoming a tool for teaching or raising awareness on a certain subject. These tools, compared to traditional learning mediums, might employ gamification techniques to improve the attention of the users, and to make the learning experience more entertaining as a whole. Hamari et al. [27] describe the process of gamification as "a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation". They argue that the goal of gamification is to improve the experience provided by a service by making it more entertaining. Gamification is not clearly defined as a set of methods or steps, but rather as a broad concept that increases the likelihood that gameful experiences might emerge in an application.

Alvarez et al. [10] classify serious games by three criteria: gameplay, which provides information on the type of interactions used between the user and an application, purpose, which describes the ulterior scope of the application, such as the learning goal, and sector, the public a serious game is aimed for, or the domain in which it aims to bring improvements. By using these criteria, designers can reflect on the "playful" and "serious" dimensions when creating a serious game. However, games can be used for other purposes than the ones intended by their designers, and players can also use a game designed purely for entertainment as a serious game.

Goh et al. [26] attribute the positive associations with video games to the fact that serious games offer covert learning, bypassing the psychological aversion to traditional educational methods.

The positive attitude users have towards games and the fact that players feel much more in control when playing a game have made serious games a much more accepted platform for learning [34] [16]. Prensky [55] notes that fun added to the learning process creates relaxation, which in turn enables students to put in effort without resentment.

Michael & Chen [48] state that serious games put the player in a more active situation than conventional training or educational methods. Michael & Chen also describe the importance of measuring the results or effects of a serious game. Serious games used for science education are very important since they enhance the learning process [17]. Many serious games that tackle interpersonal communication, such as TLCTS [33], or VECTOR [47], allow learners to explore foreign towns and practice speaking Arabic and English respectively while learning culturally appropriate gestures. Other serious games focus on cultural social conventions [41] or facilitate interpersonal communication, such as SimParc [14] or DeLaryous [63].

Laamarti et al. [35] state that serious games with the learning goal of training communication allow the player to experiment and make mistakes without having to fear the consequences of not having experience handling some situations.

Communicate [31] is a serious game created by Utrecht University to train students in communication skills. Afterward Communicate was rebranded as DialogueTrainer and used more widely, offering clients online professional communication training.

The goal of Communicate is to simulate specific scenarios that can be used to teach important interpersonal communication skills between a healthcare professional and a patient or between an employer and their employee. The game provides a player with prewritten statement options from which they can choose, leading to a conversation between them and a digital agent. A teacher can use a scenario editor to create their own personalized scenarios for players to practice communication skills on. This offers a platform for experiencing communication scenarios tailored specifically to the needs of a player. Lala et al. [39] describe a scenario as 'a series of interactions between a player and a virtual character for one-to-one communication skills training, where at each step a player is faced with a choice between statements' and specifically, in the serious game Communicate, as a directed acyclic graph of statements. These graphs can be easily created by a teacher, who can have fine control over the form and structure of the scenario and annotate each prewritten statement option in the scenario with scores on the desired learning goals, emotional effect on the player, and player feedback. This allows a teacher to easily create scenarios without needing programming skills to implement them in the game. Engström et al. [23] discuss the importance of tools for designing stories in video games. The authors present software used for this kind of development, however, state that often times these tools restrict the expressiveness of the designers. These tools should prioritize expressive freedom while offering very easy-to-use interfaces that reduce the overhead of such work. The scenario editor in Communicate follows this, providing a simple-to-use UI while offering a teacher all the freedom needed to create the scenarios.

The importance of open text input option comes from a study by Krathwohl, cognitive process taxonomy [12]. Here, 6 cognitive process dimensions are defined: remember, understand, apply, analyze, evaluate and create. Prewritten statement options test the first five of these cognitive process dimensions, but not the sixth. While the prewritten statement options of a scenario are easy to score, they do not demonstrate that a player can use elements from a scenario to create new structures. On the other hand, open text input is harder to score but requires complex thinking, something that is desired from a serious game.

The process of creation is defined as 'putting elements together to form a novel coherent whole or make an original product'. This process is tested with the introduction of open text input. A learner is required to analyze the information that is given to them, and create a coherent answer that furthers conversation and adds to it, rather than choosing between prewritten statements.

In previous work Lala et al. [32] added an open text input option in Communicate so that a player can input open text instead of selecting one of the prewritten statement options. To achieve this Lala et al. introduced the scenario-specific corpus method (SSCM) to match a player's open-text input to available prewritten statement options. SSCM takes an open text input and returns a match-score for each prewritten statement at a step of the scenario. The match-score determines how similar the open text input is to each of the prewritten statement options. In case all of the match-scores fall below a certain threshold, the input utterance is considered unmatched.

In order to find the ideal threshold and compare the SSCM to other matching methods, a dataset is created [32]. Open text statements are gathered from students in communication workshops using a scenario named "SamenwerkenOT". At each step in the scenario, students are given a sentence by the digital agent and are tasked to give an open text input, instead of choosing one of the scripted statement choices. After providing an open text input the available scripted statement choices at that step are displayed and the students are tasked to select the statement most similar to their open text input. In case none of the prewritten statement choices match the open text input, the student may select an option indicating there is no similar scripted statement. The students do this for every step of the scenario "SamenwerkenOT". These pairs of open text input and scripted statement choices, as well as the annotation of the students, are used to create the dataset.

Each sample of this dataset is a pair of player open text input, and scripted statement options, as well as annotations from the student who wrote the input and communication experts, indicating which prewritten statement option is most similar to the open text input. If the majority of annotations of a sample agree on the same prewritten statement option as being the most similar to the open text input, the sample receives a label, called the goldenMatch, indicating to which of the prewritten statement options the open text input is most similar, and the sample is called matched. If the majority of annotations agree that none of the prewritten statement options are similar to the open text input, the goldenMatch is Null, and the open text input of the sample is considered not being matched to any of the statement options, the sample being called unmatched. If there is no majority decision regarding which prewritten statement option is most similar to the open text input, the sample is discarded.

Using this dataset, called the golden dataset, the ideal threshold for the SSCM is determined by running the method on a training split of the dataset and comparing which prewritten statement option the SSCM considers most similar to the input text with the goldenMatch. The ideal threshold is found starting from the value of 0.0 and incrementing it by 0.025, calculating the number of correct matched and unmatched predictions made using the SSCM. Afterward, the SSCM is evaluated against other NLP methods using a testing split of this dataset. The results show that SSCM outperforms generic word embedding methods for matched text but underperforms for unmatched text.

In an experiment [37], Lala et al. take steps to improve the open text input of the game by introducing scaffolding, as presented by Realdon et al. [57]. Scaffolding is used to handle matched and unmatched input by highlighting matched open text input and giving hints for unmatched input. They show that this method has a limited effect in Communicate and the results can be generalized to any serious game that uses a dialogue graph, has no extensive dataset, and uses open text input.

The serious game Communicate is separated into two main components. The first component, named the client service, contains the scenario editor, where a communication skills expert or scenario writer creates scenarios, and the scenario player, where the actual "gameplay" takes place. The scenario player is where the player interacts with the scenario, going through the scenario and providing open text input that determines the outcome of the scenario. The second component of Communicate is the matching service. Following the work of Lala et al. [32] the matching service uses the scenario specific corpus method (SSCM) to compute a list of similarity scores between each prewritten statement option and the input text from a player. This matching method can be replaced with a different matching method, as long as the matching service receives the same input information from the client service, and outputs the same list of scores back to the client service, in order to keep the working pipeline of the application.

In order for computers to understand and perform various tasks using human language, words need to be represented in a mathematical way. A word embedding is a representation of text that can be used in a meaningful way in various natural language processing algorithms. Almeida et al. [8] define word embeddings as "dense, distributed, fixed-length word vectors, built using word co-occurrence statistics as per the distributional hypothesis". The distributional hypothesis was suggested by Harris [28] as long ago as 1954 and it is the assumption that words with similar contexts tend to have similar meanings. Turian et al. [62] state that word embedding methods provide vector representations of words. Generally, a word representation is a vector that is mathematically associated with a word, such that the word keeps contextual information. The relationship between two vectors produced by a word embedding model should mirror the relationship between the two represented words on a linguistic level. The vector representations of the words are multi-dimensional, each dimension representing a hidden feature of the original word. In this manner, useful syntactic and semantic properties are captured, to be used in various NLP tasks. While most vector representations were used together with Neural Network Language Models, by first projecting the embedding onto an input layer of the network, called the embedding layer, researchers realized that word embeddings can be used on their own for solving many NLP tasks.

The simplest word embedding model that can be used as an example for understanding the underlying logic behind how they function is called One-hot encoding. The general idea of the algorithm is converting each unique word from a set dictionary or corpus into a binary vector. The length of these vectors is equal to the size of the vocabulary. All the elements of the vector are equal to 0, except for the position of the word in the vocabulary. For example, for the sentence: "This is the corpus", I create the following table:

	This	Is	The	Corpus
This	1	0	0	0
Is	0	1	0	0
The	0	0	1	0
Corpus	0	0	0	1

From this table we can see the vector representation of each word: "This" = [1,0,0,0], "Is" = [0,1,0,0], "The" = [0,0,1,0], "Corpus" = [0,0,0,1]. In practice, when the original sentence is transformed into the corpus, duplicate words are removed, and sometimes even intermediary words, such as "the" or "a". These words do not contain useful information and can be removed. The algorithm assigns a unique vector representation to each unique word. It is important to remove duplicate words since the same word appearing in multiple locations in the corpus would have different representations after embedding, making the representation nonunique. One-hot encoding is a useful representation of data but does not retain any contextual relationship between words. More recent word embedding models such as GloVe [53] use word-context matrices to generate the vector representations of words. In section 4 I further describe the GloVe model and the way it generates word embeddings. These advanced methods of creating word vectors produce numerical representations that capture contextual information. With this information, language models can perform tasks such as determining the similarity between two words or sentences, or sentiment analysis.

In the field of NLP noise is considered a complex concept and is hard to define [60]. It is mostly used when describing a dataset containing non-standard samples. Sharou et al. [7] categorize noise as harmful, such as incorrect punctuation in translation tasks, and useful noise, like punctuation patterns in sentiment analysis tasks. In our application, noise can be defined as the matching of open text input to an answer variant when there should have not been a match.

The aim of this MSc. thesis is to introduce contextualized word representations combined with a scenario specific corpus for a scenario in the serious game Communicate [31], improving the matching of open text inputs and reducing noise. I aim to find the combination of techniques that offers the best accuracy for open text matching in the application. This study will improve on the already existing framework, and further its performance.

Main research question

The final goal of this work is to improve the serious game Communicate by implementing state-of-the-art NLP methods that are relevant to solving the given task, that produce better results than previously implemented methods, and that integrate seamlessly into the existing framework of the application. I want to use the current architecture of the application as a starting point for improving it. This architecture is the client and matching service structure.

I elaborate on the constraints of the application and the proposed implementation in Section 6. Furthermore, the previous matching method, SSCM, used scenario specific information to create word vectors used for matching. This information is available from the moment the scenario is authored. It is relevant to determine if this information might be useful in increasing the accuracy of other matching methods as well. Considering this, I propose the following main research question:

How can we improve the matching service in the application Communicate using word embeddings and scenario specific information?

The main research contribution of this thesis is applying word embeddings in a new context, specifically a scenario driven serious game with an open text input option. I break down the main research question into smaller individual research questions that will be tackled in the Method section, with the results later being presented in the Results section. The research questions are:

- Which word embedding methods are suitable for sentence similarity in an application for training communication skills with an open text input option?
- What is the performance of the selected word embedding models on matching open text input to prescripted statements?
- Can the implemented matching service of the application be improved with scenario specific information?

In the following section, I present some related work in the field to better understand the context of the thesis. Section 3 contains the research approach for each of the research questions. In Section 4 I chose a set of suitable embedding models for the task based on a literature review. In Section 5 I test the chosen embedding methods on the same metrics as the previous paper concerning open text input in the game Communicate, and, based on the results, choose one model to implement in the matching service of the application. Section 6 describes the implementation of the model into the matching service, the testing of the model on a test scenario, and the usage of scenario specific information to improve the model. Section 7 presents a discussion of the results and finally, Section 8 contains the conclusions of this thesis and future work.

2 Related work

In this section, I introduce some papers that describe various serious game applications used for training communication skills. All of these applications use virtual environments and agents to simulate conversation scenarios and task the user in navigating through them, similar to Communicate. While their learning goals might be different, the approach and results are similar.

While most of the related work does not describe the technical details of the technologies used for handling NLP tasks, it shows the pedagogical benefits of open text input understanding. Westera et al. [65], state that the usage of NLP techniques in games is scarce because of the large amount of preprocessing and postprocessing, as well as the expertise in implementing these NLP methods.

Van der Lubbe et al. [44] introduce speech analysis to a serious game for training against doorstep scams. Unfortunately, elderly people are at a high risk of being victims of such scams and require a tool for training against them. Verbal skills are very important in preventing them, but without the necessary education, people lack the assertiveness needed to refuse potential scammers. The paper introduces a serious game that focuses on what to say to a scammer and how users should use their voices in an assertive way. In the game, the player, representing the person getting scammed, has to audibly utter a chosen response. The assertiveness of the response is analyzed and it is classified as either dominant or submissive. The simulated character offers a prewritten answer depending on the classification. Participants that performed an experiment on the application stated that the voice analysis clearly had added value in the context.

Ochs et al. [51] propose using serious games for training doctors to break bad news to patients. This is relevant because the delivery of bad news has important psychological effects on patients, as well as effects on the doctor-patient relationship. The player assumes the role of the doctor in the virtual reality simulation. The speech input from the player is analyzed and matched to one of the prewritten verbal and non-verbal responses. This response is given to the player as an answer from the patient. After an initial experiment with experts and naive participants, the users were asked different questions that were meant to measure their sense of presence and copresence. Results show that the experts tend to be more involved than the naive participants. The experiment shows that the virtual reality room and headset are the most appropriate virtual reality tools for training doctors in such a scenario.

Façade [46] is a real-time simulation of a marital conflict. The player can input responses to the presented conflict. These open text responses determine the reactions of the simulated characters. The application uses author intensive techniques to understand natural language typed by the player, meaning that a large number of rules are hardcoded. If the system still does not understand the utterance, failsafe behaviors are triggered, such as deflection-type responses, or an abrupt change of subject. The system might also misunderstand utterances, giving a false positive.

Anderson et al. [11] develop the TARDIS project, a serious game intended to simulate interview situations for young people not in employment, education, or training. The players interact with virtual recruiters, in order to experience real-life interviews without the risk of failure. The application detects its users' emotions and attitudes through voice and facial expression recognition. With this information, it is able to better adapt the conversation to individual users. Furthermore, it allows experts in the field to design specific scenarios and meaningfully measure a user's emotional regulation and social skill acquisition. This information is useful for furthering research in the field and creating more specific testing scenarios.

My Automated Conversation coacH, MACH [29] is an application for training social skills. It displays a conversation with a virtual agent that reads the facial expressions, speech, and prosody of a user and responds with verbal and nonverbal behaviors in real time. The application then provides visual feedback on the user's performance. The interview data is provided by real-life interview sessions, such as seeking employment or counseling. The user's face is tracked by the application which detects smiles and head movements in every frame. Pauses, loudness, and pitch variations are also tracked in order to assess the expressivity of the user. The application recreates the entire transcript of the conversation, however, it does not perform any natural language understanding, the main focus being nonverbal training. An experimental study was performed with 90 students and two professionals to validate the effectiveness and usability of the application in an interview scenario. Results show that participants learned something new about their behaviors and were inclined to use the application in the future.

Tartaro et al. [61] present an application for children with social and communication deficits in which they engage in a collaborative conversation with a virtual character. Collaborative conversations require the children to work together in order to create a story. In this study, children engage in conversations with human and virtual characters using this application, and the use of contingent conversation is compared between the two cases. The authors find that contingent conversations increased over the course of interaction with the virtual character, while with the human character they did not. Moreover, introducing new topics or maintaining the current topic was more common in the conversations with the virtual character.

Serious game applications for training communication skills often use open text input from a player to increase immersion. However, these open text utterances are gathered together with other information, such as facial expressions [11] or speech samples [29] to better understand the emotions of a player. Moreover, applications that only gather open text utterances do not use state-of-the-art embedding models to analyze the text, and rely on worse-performing methods, or hardcoding rules for understanding [46].

3 Research Approach

In this section, I describe the solution for each research question presented previously. The research questions follow each other so I present the results of each method after stating it. Each subsection in this chapter follows the results of the previous one.

Firstly I need to determine what kind of NLP task is the current version of the matching service of Communicate trying to solve. Possible NLP tasks range from annotation, text classification, and text summarizing to question and answering or text generation. This can be determined by studying the input and output of the current application and choosing the task that provides the most similar results. Next, having the NLP task selected, I look at the state-of-the-art models that are used for solving this task. This is done through a literature review of relevant papers in the field. Then, I apply the new method to the existing framework and test it using the same metrics as the previous paper that tackled the open text input in Communicate. This means using the same datasets to determine a threshold for cosine similarity, and computing the same metrics. Finally, after determining if there was an improvement to the original paper just by implementing a state-of-the-art pretrained model, I determine if using scenario specific information can improve the matching service of the serious game Communicate.

4 RQ1: Word embedding models

The starting point of this thesis is choosing a set of word embedding models that perform best on the language task that the matching service is solving. The task is a sentence similarity problem since an input sentence is provided by the user and the service needs to match it to the most similar prewritten statement option. Sentence similarity is a measure of how similar the meaning of two sentences is. This is a hard problem in the NLP field since the meaning of utterances is given by the context they are in as well as the lexical information [45]. In our case, after each sentence is converted into a word vector by an embedding model, their similarity is measured by the cosine similarity of the two vectors.

In this section, I describe the method of choosing this set, the resulting set as well as the reasoning behind the choices. By the end of the section, I have gathered the embedding models needed for testing in the next section.

4.1 RQ1 Method

In order to better understand the evolution of word embeddings and natural language processing models I conduct a literature review of relevant papers in the field that describe word embedding models, their use, and implementation. I use these papers to determine what models are relevant and how they evolved since the original Communicate paper [31] has been published. Since the framework in which I am implementing the models is a serious game, computation time is also an important factor to keep in mind.

4.2 RQ1 Results

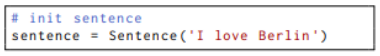
Literature review

In this section, I conduct a literature review on state-of-the-art NLP models. By doing this I aim to gain a better understanding of what models are relevant and which ones I can use to improve open text input understanding. I choose the papers discussed in the following literature review based on the popularity of the NLP methods, making sure that they are as recent as possible. I use snowballing [66] as a way to search for relevant literature by finding other useful material. Snowballing is the process of finding related papers by using the reference list and the citations of papers in a certain start set.

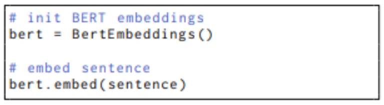
The start set I used for this literature review is composed of state-of-the-art NLP models such as Glove [53], Bert [22], and Elmo [54]. These models are important in the field of NLP because they introduce contextualized word embeddings, representations that retain the context of the sentence they appear in. This increased the performance on many NLP tasks such as question answering, language inference, and similarity tasks, and future developments of word embedding models refer back to these models. From these I perform backward snowballing in order to gather papers that explain the evolution of an NLP model, to gain a better understanding of why it is state-of-the-art. I also perform forward snowballing to gather works that expand on the model and improve its performance.

FLAIR

Without experimental data, it is hard to determine the best combinations of embedding models and language model layers for a specific application. The implementation of these embedding models often requires a rework of the entire language model used for solving the NLP task. Since testing the performance of all the combinations of embedding models and language models becomes exponentially difficult with the increasing number of embedding models, Akbik et al. [6] introduce a novel method of mixing and matching various embedding models with language models easily. With the help of this framework, a researcher can focus solely on finding the optimal embedding model and language model combinations. Language models often have parameters that can be tuned by users such as the learning rate in optimization algorithms or the number of epochs the model trains for. These "hyperparameters" need to be chosen experimentally and are specific to the problem the model is designed to solve. The framework also allows easy hyperparameter tuning directly in the interface. The framework also allows easy fetching of public NLP datasets that can be used for setting up experiments. The framework automatically reads them into data structures that can be used for training or testing purposes. Finally, Akbik et al. are working on introducing more embedding approaches into the framework, word embeddings such as transformer embeddings [56, 19], InferSent representations [18] and LASER embeddings [13]. This paper provides a useful framework that can be used to quickly compare embedding models. As an example of how the framework simplifies the process of testing NLP models, to use BERT embeddings to embed a sentence the user has to instantiate the embedding with one line of code and embed the sentence with another single line of code:



```
# init sentence
sentence = Sentence('I love Berlin')
```



```
# init BERT embeddings
bert = BertEmbeddings()

# embed sentence
bert.embed(sentence)
```

Transformers

Many embedding transformation models use recurrent or convolutional neural networks including an encoder and decoder. The encoder maps an input sequence of symbol representations to a sequence of continuous representations, while the decoder generates an output sequence of symbols one element at a time. At each time step in this process, the model uses the previously generated symbols for the generation of the next. In the best performing models, these encoder and decoder elements are also connected through an attention mechanism. This allows the modeling of dependencies without regard to their distance in the input or output sequences. Vaswani et al. [64] propose a new network architecture that uses only this attention mechanism, getting rid of the recurrence and convolutions. Vaswani et al. show that the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. Many embedding models discussed further are based on the transformer architecture such as BERT [22] or GPT [15], and the paper is important for understanding how they work, and how they can be implemented in our specific application.

Wor2Vec

Mikolov et al. [49] present a method for creating vector representations of words from a corpus. This vector is placed in a vector space, where similar words are placed closer together. The model uses a neural network architecture. Mikolov et al. present two learning models, Continuous Bag-of-Words, which uses the context words to predict a target word, and Continuous Skip-gram being the opposite, using the target word to predict the context words. Context words are the words closest to the target word and are gathered by having a moving window over the corpus of text. By having this moving window, training pairs are created, consisting of target words as well as context words. The one layer neural network is then trained on these pairs of words. By using backpropagation and gradient descent, the model tunes weights that are the vector representation of words. In a follow-up paper [59] Xin Rong explains the use of weight tuning.

GLoVe

Pennington et al. [53] present a new method of creating vector spaces for words by using not only local information about the words, such as local context information but also global statistics, such as word co-occurrence. By using global and local statistics of a corpus, GLoVe captures some of the semantic relationships between words. The model uses a word-word co-occurrence count, whose entries note the number of times a word occurs in the context of another word. The GloVe model performs significantly better than other baselines, often with smaller vector sizes and smaller corpora.

LSA

Latent semantic analysis (LSA) [40] is a natural language processing technique used to create representations of text data in terms of some hidden features. LSA relies on the fact that once words are embedded into a vectorial space, terms with similar meanings are closer to each other. And as such, words with similar meanings usually are present in similar contexts. The way LSA works is that input is gathered from a training corpus and a word-by-document co-occurrence matrix is constructed. This matrix contains the occurrences of each word in each document. The authors perform a dimensionality reduction by using a method called Singular Value Decomposition (SVD) [25]. The method is similar to principal component analysis [67], projecting every word in a subspace with a predefined number of dimensions. Finally, the semantic similarity between words that have been embedded in such a vectorial space can be obtained by the cosine angle between them. Altszyler et al. [9] compare LSA, a counter-based model, and Word2Vec, a prediction-based model, performances on a small corpus. They argue that often there are situations where embeddings have to be created with only scarce data. Altszyler et al. compare the capability of the models to represent semantic categories, as well as analyze and disambiguate the content of dreams. The results of the paper show that Word2Vec performs better when there is substantial training data, however, when this is reduced, LSA becomes the more viable model.

ELMO

Previous word embedding models such as Word2Vec and GLoVe would generate a single embedding vector for each word. Because of this, it is impossible to determine context just from the word embeddings. Peters et al. [54] introduce a deep contextualized word representation, able to capture context as well as syntax and semantics, called ELMo (Embeddings from Language Models). Essentially, ELMo is a function that provides a contextualized embedding for words, getting a sentence as input and outputting the embedding vector. Elmo uses all the internal states of a bidirectional language model to generate these embeddings. Language modeling (LM) is the use of various techniques to determine the probability of words occurring in a sentence. Language models determine word probability by analyzing text data. Bidirectional models(biLM) analyze text in both directions, backwards and forwards. The main advantage of using language modeling for an NLP model is you don't need additional labels for training data, and just a sufficiently large dataset to predict previous and future words. When ELMo is being trained, the LM predicts future words, having been given previous words, while the backward LM is trained to predict previous words given future words. This helps to generate more accurate word embeddings, by capturing context information from multiple passes through the given text. While other state-of-the-art embedding models at the time use only the last layers for output, ELMo uses all of the internal states of the biLM. The authors found that the higher layer outputs capture more context information, while the lower layer outputs capture information about syntax in the generated word vector. When training, the LM is given as input a sentence. Each word is converted into a character embedding, and it is given to the first Long Short-term Memory (LSTM) cell of the model. The authors use character embeddings because the initial layer embeddings should be context independent. Furthermore, to be able to compare the proposed method with pretrained word embeddings, these could not be used in the training of ELMo. Deeper layers of the LSTM start outputting context dependent embeddings. In order to combine the features of the forward LM and backward LM, Elmo concatenates vectors between each internal state. Afterward, each vector is multiplied with a weight, which is tuned in the training process. Finally, all vectors from all the layers are summed up and represent the final word embeddings. Peters et al. prove that summing up all the layers resulted in better scores.

BERT

Devlin et al. [22] describe a new method of embedding words that builds on the advantages of previously used methods. At the time of publishing this paper, there have been two main strategies for applying pretrained language representations: feature-based and fine-tuning. For feature-based language representations, I present ELMO, which uses task-specific architectures that include the pretrained representations as additional features. Fine-tuning language representation, such as the Generative Pretrained Transformer (OpenAI GPT) [56] use minimal task-specific parameters, and are trained on the downstream tasks by simply fine-tuning all pretrained parameters. The BERT language representation presented by Devlin et al. [22] combines these two strategies to achieve a model that can solve many NLP tasks with much lower training times. There are two steps in the framework: pretraining where the model understands the general semantics of the language and context, and fine-tuning, where the model learns to solve a specific task. BERT: Bidirectional Encoder Representations from Transformers is pretrained by using a "masked language model" (MLM) which masks some of the tokens from the input, and the objective is to predict the original vocabulary ID of the masked word based only on its context. Each token is converted into a word embedding using pretrained embeddings. This helps the model understand bidirectional context within sentences. For pretraining, the model also uses a "next sentence prediction" task that jointly pretrains text-pair representations. Given two sentences, the model determines whether the second sentence follows the first, and helps the model understand the context across different sentences. For the fine-tuning phase of the model, the output layer is replaced with a specific set depending on the task, and only that layer is retrained. That is what makes the training of BERT fast, independent of the NLP task. BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the previously discussed transformer architecture [64]. The BERT Transformer uses bidirectional self-attention, while the GPT Transformer uses constrained self-attention where every token can only attend to the context to its left. The BERT model is a combination of the state-of-the-art language representation methods at the time. It offers the advantages of both feature-based architectures and models based on fine-tuning. It is also a precursor of XLNet [69] and useful for understanding the evolution of language models.

XLNet

Yang et al. [69] present an extension to the BERT architecture discussed previously. The authors describe a model that combines two kinds of language modeling techniques. The autoregressive approach predicts each timestep using previous timesteps but has the limitation that even when using bidir lSTM , it never truly captures the entire context of the given text. The autoencoding denoiser approach (as seen to great performances in BERT [22]) is based on the transformer architecture and is good at understanding context from all the text. However, the limitation of this technique is that when training with a masked language model, masked tokens are in the input of the model and it is possible that these tokens are interdependent. XLNet combines these two approaches and does not use an LSTM, but is entirely based on self-attention like the transformer [64] models. While BERT predicts all masked tokens at once, XLNet predicts the words one at a time in order to add to the context of the entire sentence autoregressively. XLNet uses a permutation LM instead of a bidirectional LSTM to train on all possible permutations of a given text. It uses a sinusoidal signal for positional encoding as seen in the transformer architecture but it also integrates relative positional encoding from TransformerXL [20]. The model outperforms BERT on 18 out of 20 NLP tasks and this is why it is relevant to research.

Comparing the performance of NLP models is very important to determine which models are best suited for various NLP tasks, and which models can be improved. Calculating accuracy, precision, recall, and F1 score will give a good rough idea of how these models perform. Akbik et al. [6] provide a framework for easy hyperparameter tuning and combining models. A study on topic segmentation using word embeddings [50] that were state-of-the-art at that time (LSA, Wor2Vec, and GloVe) compared the methods on two corpora containing computer science, economy, politic, and health topics. Based on the corpora, the authors created training and testing collections. They evaluated the performance of these methods using WindowDiff metric that measures the error rate by using a sliding window. This work is useful for understanding the methodology of comparing word embedding models.

Kawin Ethayarajh [24] compares three contextual embedding methods (ELMo, BERT, and GPT-2). Here training data is taken from the SemEval Semantic Textual Similarity tasks from years 2012 - 2016 [3, 2, 4, 5]. The reason for choosing these datasets is that they contain many homonyms, words that have different meanings depending on the context they are in.

In this article [30] Horan Cathal compares BERT, ElMo, XlNet, and USE on a sentence similarity task to determine which model is better suited. The author stresses the fact that while some embedding models are considered in general better than others, on specific tasks they may perform worse. Moreover, it is desirable to use pretrained models rather than training them from scratch, since this takes very large datasets and a considerable amount of training time. From the comparison, I can see that BERT is well suited for tasks that involve determining which sentence from a given set is more similar to another sentence.

Achananuparp et al. [1] write about evaluation criteria for determining the efficiency of sentence similarity models. They define six evaluation metrics to use; Recall, Precision, Rejection, Accuracy, F1 Score, and f1 Score defined as the uniform harmonic mean of rejection and recall.

In a similar study on integrating Embeddings from Language Models (ELMo) and the Bidirectional Encoder Representations from Transformers (BERT) with the transformer encoder, Laskar et al. [42] propose using mean average precision (MAP) and mean reciprocal rank (MRR) to evaluate the models.

The criteria for selecting a word embedding model were ease of implementation into an existing application, having pretrained embedding models in order to run the application while the user is playing the scenario. From the previously presented literature review, I conclude that the transformer architecture is the most suited for open text matching in our application. The transformer architecture offers, at the moment of writing, state-of-the-art word embedding models. I look into libraries that offer transformer based word embedding models. Furthermore, I determine that the programming language Python offers the fastest methods for prototyping and testing implemented word embedding models, so the service I will develop will also use Python.

A paper [68] I came across while performing the literature review described a state-of-the-art natural language processing library based on the transformers architecture, called Huggingface. The library contains a large number of ready-to-use, pretrained embedding models that are state-of-the-art in various NLP tasks.

Another available python library that provides pretrained models is the SpaCy library. Similarly to Huggingface, this library offers an interface for implementing state-of-the-art models into any python application, bypassing the time time-consuming steps of implementing and training the models. However, in a previous experiment by Lala et al. [38], matching free text inputs with methods provided by SpaCy resulted in lower scores than the currently implemented SSCM method.

Patel et al. [52] compare SpaCy, Huggingface, and fast.ai, a less developed and popular library option than the other two. While SpaCy has been available longer than Huggingface, until last year the NLP models it has offered relied on the recurrent neural network architecture rather than the state-of-the-art transformer-based one. Huggingface on the other hand focuses mainly on NLP tasks with its transformer library, being optimized for speed when performing these tasks, and has seen huge popularity and funding over the last few years. This means that using the library might become a standard in production and development, an important aspect when developing an application with long-term support.

Another advantage of Huggingface is that it offers an easy-to-use class for finetuning the pretrained models using custom datasets. This is important for my use case since the problem I tackle in the third research question is finetuning the model on scenario specific information. This means a fast and easy-to-use method of finetuning is an important criterion in choosing the library of models to go forward with.

I decide to use the Huggingface library for the following reasons. Firstly, the models can be used directly by importing them with code and there is no need for reimplementing them from scratch. This is a very important criterion since the main contribution of this thesis is to bring improvements to the open text input understanding of a serious game. Considering this, the goal is not to implement a model from scratch, but to find suitable models that can be used in the matching service. Secondly, considering the online nature of the application, the models need to be pretrained in order for the matching to be performed relatively fast for practical use in a real-time application. The models in the Huggingface library are already pretrained on very large datasets, something that would not be feasible and efficient for me to do from scratch. Furthermore, the models I want to select for testing need to be state-of-the-art in the field of word embeddings. The goal is to use the fastest and best performing models for the task while keeping computation times low. The Huggingface library offers these state-of-the-art models, as well as a list of models for each task, ranked based on their performance scores on a standard validation set.

The task I am trying to solve can be presented as a sentence similarity task: A sentence is provided by the user and the matching service is tasked with calculating the similarity between this provided input sentence and some prewritten statement options. The statement with the highest score and also above a predetermined threshold is chosen as a suitable interpretation for the open text input. The threshold is determined by evaluating the model on the golden dataset created previously from communication workshops using Communicate. This golden dataset contains multiple samples. Each sample contains an input sentence written by a student in the workshop and one or more prewritten statement options gathered from the scenario. The sample also contains one label determined by the user on which statement option best matches the input sentence, as well as two other labels determined by experts. If at least two out of the three labels are the same, the golden match of this sample is equal to that label. If there is no majority decision on the labels, the sample is unmatched and labeled as 'null' in the golden match.

Finally, since the application I am working on is mainly for Dutch language use, the models need to either be specifically trained on Dutch datasets or be a multilingual model that can be used for Dutch specific language tasks. Reimers et al. [58] present a method to make sentence embedding models trained on one language perform well in another language. This was done using multilingual knowledge distillation. They show that models trained in a high-resource language, like English, can be extended to support multiple languages. Their method uses a teacher model and a set of translated sentences from the source language to the desired language. Training a student model on the translated sentences such that the final embedding vector is similar to the vector of the teacher model.

One of the multilingual models chosen from the Huggingface transformer library is called "clips/mfaq" [21]. The model is based on the BERT architecture, specifically using the RoBERTa base model. RoBERTa is a reimplementation of BERT that focuses on the significant impact that hyperparameter tuning has. Liu et al. [43] determine that the original BERT model was significantly undertrained, and the RoBERTa implementation performs significantly better can matching or exceeding every model published after it. The "clips/mfaq" model I am using for this application is pretrained on the "MFAQ" [21] dataset, which is a multilingual corpus of Frequently Asked Questions parsed from the Common Crawl. Common Crawl is an open-source database of web crawling data. "Clips/mfaq" 6 million pairs of questions and answers in 21 different languages, including Dutch and English.

Finally, considering everything mentioned previously I choose to further test a set of 5 word embedding models. This set contains models from the Huggingface library. The models are "clips/mfaq", which is a multilingual RoBERTa based word embedding, "multi-qa-mpnet-base-dot-v1" a model used for semantic search and "all-mpnet-base-v2", "all-distilroberta-v1" and "all-MiniLM-L12-v2", which were trained on all available training data and are designed as general purpose models. These models have performed best on various NLP tasks including sentence similarity and semantic search, and all have different architectures. They are considered the best general purpose models, however, that does not indicate if they are best suited for our specific task. We perform these tests to determine which one performs best. Due to the amount of data and complexity of testing all possible threshold values for all possible models, I choose to only perform tests on the top 4 general purpose models and the only multilingual dutch specific model on Huggingface, "clips/mfaq". Further work may include testing other models to see if they perform better. In the next section, I describe the research approach for the second research question, test and choose the best model for our specific task, as well as present the results of this testing.

5 RQ2: Testing word embedding models on the golden dataset

In this chapter, I focus on answering the second research question, specifically which of the selected word embedding models from the previous chapter performs best in my specific case.

In previous work [32], a dataset is created from open text utterances gathered in communication workshops. Each sample in the dataset contains an open text input from a student, prewritten statement options gathered from the scenario, and a value indicating which statement option is most similar to the open text input. The value is given by the student who gave the input and two experts. If the majority agree on the same value, the sample is considered matched. Alternatively, if the majority agree that none of the statement options are similar to the input, the sample is considered unmatched, and the value is Null. In the samples, the value is called the goldenMatch. The dataset is split into smaller training, testing, and validation datasets. In the previous experiments, the validation dataset was used to evaluate the best performing SSCM configurations, as well as other NLP methods used as performance comparisons. The method proposed in this thesis does not separate testing and validation. This is because, at this step, the word embedding models are already pretrained and do not need further training or validation. Instead, I use the training as well as the validation set from the previously mentioned dataset to test the models chosen in the previous section, since the increased number of samples will translate into more accurate results when finding the threshold. For evaluation, I want to compare the chosen word embedding method with the SSCM. Previous work [36] used only the training dataset when comparing the SSCM to other methods. Similarly, after finding the best model from the set, in order to fairly compare it to the results from the previous experiment [36], I also use only the Training dataset for evaluation.

5.1 RQ2 Method

In the context of our application, open text inputs can be either matched to a prewritten statement option or not matched to any of the statement options, considered unmatched. We define noise as the amount of incorrectly labeled unmatched open text inputs.

One of the goals of this thesis is to improve the correct detection of unmatched samples while keeping the correct detection of matched samples still high. Lowering the threshold of matching does increase the detection of unmatched samples, however, it also lowers the detection of matched ones, so this method is not viable. We determined a threshold where the ratios of matched/unmatched samples to their number of occurrences are closest to each other. From here I need to improve the embedding model to correctly determine samples that have no match.

In order to determine which model performs best for the use case of the application, I use a testing bench. Here I calculate the best threshold for each model.

In order to maximize correct labeling on both the matched and unmatched samples, the ideal threshold should be chosen when the percentages of correctly detected samples from both groups are the highest. This is because, in the dataset, the total number of matched and unmatched samples are not equal. In order to get the percentages of correctly matched and unmatched samples, we divide the number of correctly detected samples by the total number of samples in the dataset, for each group.

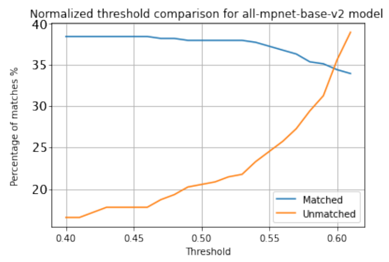
I perform this test on the combined training and validation datasets, since choosing a threshold does not require validation, the testing step will suffice. To reduce the number of computations needed to perform this search, I end the search when the absolute value between the relative number of found matched and unmatched samples reaches a minimum. At this threshold, the two values are both as high as possible, and while one might increase after that point, the other one would decrease. I start the search at a low threshold value to ensure that I do not miss the point of intersection.

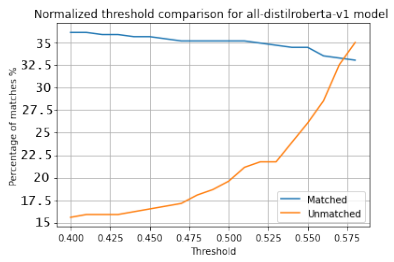
Using the thresholds I find, I run the models on the testing dataset and compare the results.

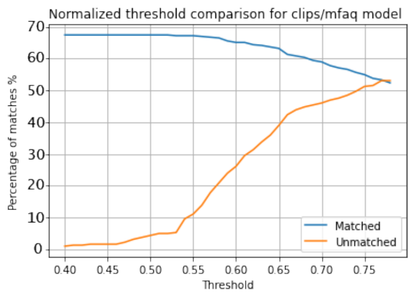
5.2 RQ2 Results

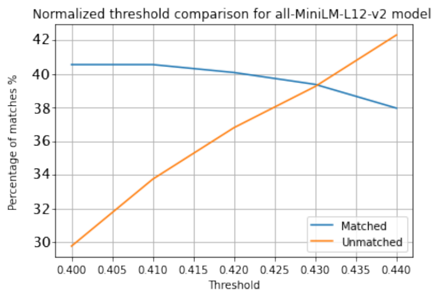
In this section, I present the results of the threshold testing and show the ideal threshold for each model.

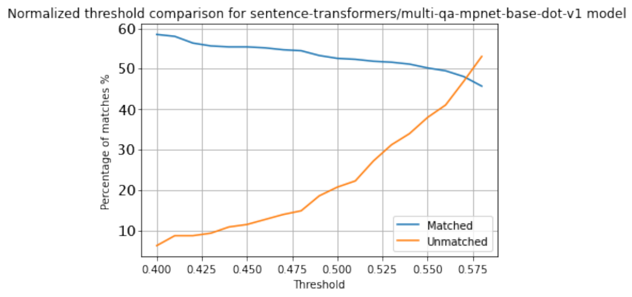
The following figures show the evolution of the correct matched and unmatched results while increasing the threshold for each model. The table shows the threshold that gives the best results for each of the tested models.











	Ideal Threshold	Correct matched results		Correct unmatched results		Total correct results	
Totals		**424**		**326**		**750**	
Model		#	%	#	%	#	%
Clips/mfaq	0.77	226	53.301	173	53.067	399	53.2
multi-qa-mpnet-base-dot-v1	0.52	204	48.113	153	46.932	357	47.599
all-mpnet-base-v2	0.6	146	34.433	116	35.582	262	34.933
all-distilroberta-v1	0.57	141	33.254	106	32.515	207	27.6
all-MiniLM-L12-v2	0.43	167	39.386	128	39.263	295	39.333

Table 1: Best thresholds found evaluating on the Training + Validation dataset

Results show that the ”clips/mfaq” model had the best performance out of all the tested embedding models. This shows that while the multilingual models have a very wide functionality and can be used in many different languages, they do not perform as well as the model trained specifically on the language used in the task.

I use the ”clips/mfaq” embedding model to answer the next research question.

6 RQ3: Implementation of the matching service

In this chapter, I present the current architecture of the serious game, the limitations that need to be taken into consideration, and the method of implementing the chosen embedding model from the previous step. Furthermore, I present a method for fine-tuning the model on scenario specific information taken directly from the scenario editor. I show the results before and after applying this fine-tuning.

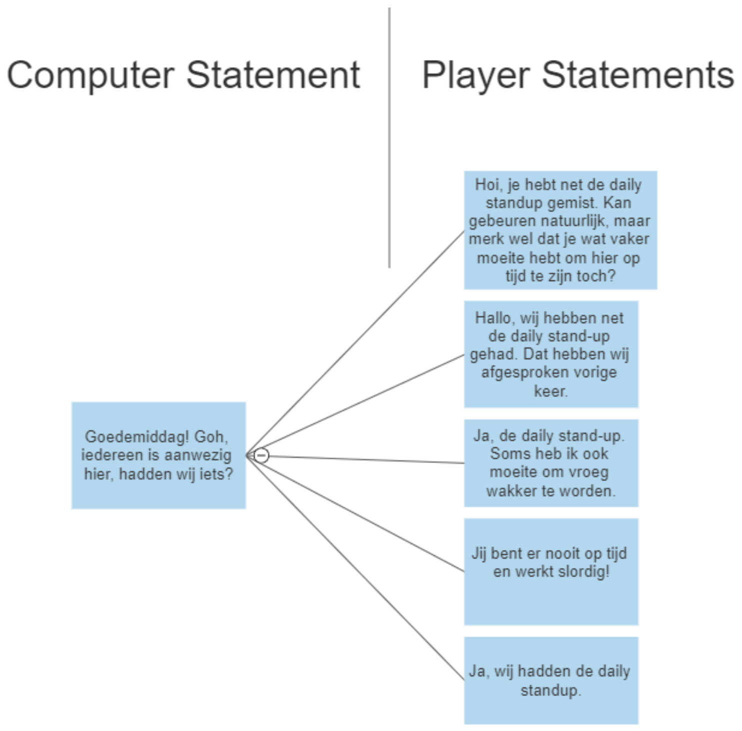
6.1 RQ3 method

The matching service of Communicate where the SSCM is used is a C# application running on a server. The matching service can be modified as long as the output result and the received input data maintain the same format. A newly created python matching server can be easily integrated into the existing architecture. I test the new matching method on my local machine before uploading it to the server. The scenario selected for testing the compatibility of the matching service with the client service is called 'Advanced Game Merchant Dialogue' and is a scenario specifically created for testing the original SSCM. It contains a dialogue tree with an end goal as well as an option to return to a previous stage of the conversation, in order to simulate all possible interactions that can happen in the application. When testing on my local machine, the game client is modified slightly to send the packages to a local server, instead of the official Communicate server. The packages contain the input that the user has put into the open text input box and the prewritten statement options obtained from the scenario tree. The packages also contain the scenario ID and information about the matching method. On the local server, the sentences in the package are separated into pairs. These pairs contain a question field, which is the input sentence from the users, and an answers field, which contains all the prewritten statement options. The word embedding model runs on these pairs and computes similarity scores between each statement option and the user input.

Using the best threshold I found for the "clips/mfaq" model in the previous chapter, I run the model on the same dataset as the SSCM, to compare the results and see if there is a relevant improvement. SSCM was run on the Testing dataset.

In this context, finetuning means further training the already pretrained model on scenario specific information. The goal is to examine whether training the model on prewritten statement options improves its performance when evaluating it on the golden dataset.

To finetune the model I will reuse some of the code that was used for the original training of the "clips/mfaq" model [21]. The pretrained "clips/mfaq" model is available in the transformers library and can be initialized directly in code. Instead of training the embedding model from scratch, I will load the pretrained embedding model and train it further on the scenario "SamenwerkenOT". This scenario was created for testing the SSCM and was used to create the golden dataset. Communicate allows the download of an XML file containing the scenario information. This XML is structured such that every dialogue option points to a possible statement option that follows it. In this way, a tree structure can be created, where every statement of the computer can have one or more prewritten statement options for the player to choose from:


Computer Statement
Player Statements
Goedemiddag! Goh, iedereen is aanwezig hier, hadden wij iets?
Hoi, je hebt net de daily standup gemist. Kan gebeuren natuurlijk, maar merk wel dat je wat vaker moeite hebt om hier op tijd te zijn toch?
Hallo, wij hebben net de daily stand-up gehad. Dat hebben wij afgesproken vorige keer.
Ja, de daily stand-up. Soms heb ik ook moeite om vroeg wakker te worden.
Jij bent er nooit op tijd en werkt slordig!
Ja, wij hadden de daily standup.

Using this tree structure I create a scenario specific dataset that can be used to finetune the embedding model. In order to reuse the training code for the ”clips/mfaq model”, I structure the data in the same format as the ”MFAQ” [21] dataset. The scenario specific dataset is structured as a dictionary containing pairs consisting of a question and multiple answers. For the data in the scenario XML, I consider the statement given by the computer as being the question and the prewritten statement options as the answers. This is because a statement option can only follow after a computer statement, while a computer statement can have multiple statement options following from it. The answers field of the pair contains all of the statement options that follow from the corresponding computer statement, separated by two newline characters.

Another configuration for this dataset could be having the pairs as single sentences in both the question and answers fields. However, because there are more statement options than computer statements, in order to include all of the statement options in the dataset, some computer statements will appear in multiple pairs, creating an imbalanced dataset, which may affect training. This is a modification to the training algorithm that will be explored in future work.

Reusing the training structure of the ”clips/mfaq” model is not an ideal way of finetuning the model. The structure was created for the particular ”MFAQ” dataset, while the scenario specific dataset is much smaller. Because of this, running the application takes a very long time and high computing power. I use this method to determine if the accuracy of matching can be increased, however, future work will develop a training algorithm specific to this dataset.

To finetune the pretrained "clips/mfaq" model I first load the pretrained model from the Huggingface library. The same tokenizer is used as in the original training of the dataset in order to arrange the data in a form that can be given as input to the model. The tokenizer creates the training samples as a touple of question, answer, and a label represented by a numerical id. The model is finetuned by matching the question to the answer that has the same label. After each training sample, the internal weights of the model are modified.

I first finetune the word embedding model on the scenario specific dataset created from the "SamenwerkenOT" scenario. To evaluate the finetuned model, I use the same validation algorithm as in the previous chapter. Firstly I validate the model on the Testing dataset, and then on the Training dataset. I only evaluate the model for threshold values between 0.65 and 0.75, since I experimentally determined that that is the range where the percentages of matched and unmatched results are closest.

The finetuning process is done in steps, at each step inputting a training sample into the model and changing the internal weights. Since the scenario specific dataset contains 23 training samples, the finetuning process will be done in 23 steps to pass through all of the data.

6.2 RQ3 Results

In this section, I first present the results of the "clips/mfaq" model before finetuning compared to the previously introduced SSCM. Afterward, I present the results after the finetuning process.

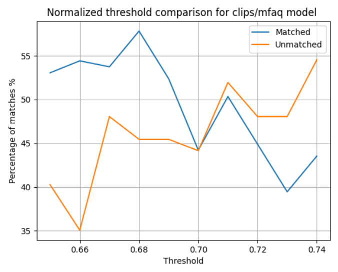
Running the model on the Testing dataset, similarly to how SSCM was compared to other word embedding methods, I find these results:

		Matched		Unmatched		Total	
TOTALS		147		77		224	
		#	%	#	%	#	%
SSCM	Best Matched	74	50	36	47	110	49
	Best Unmatched	35	24	67	87	102	46
	Best threshold found (0.61)	105	71.4285714	29	37.66233766	134	59.82142857
	Highest F1 score (0.748)	128	87.0748299	10	12.98701299	138	61.60714286
Clips/mfaq	Closest percentages	81	55.1020408	42	54.54545455	123	54.91071429
	Best Matched	129	87.755102	5	6.493506494	134	59.82142857
	Best Unmatched	6	4.08163265	77	100	83	37.05357143

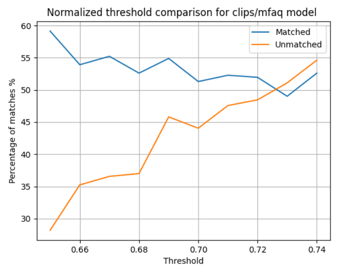
Table 2: Comparison between SSCM and non-finetuned Clips/mfaq on Testing dataset

As we can see, the model correctly finds more matched samples than SSCM for the threshold found on the training+validation dataset. However, it does not perform as well as the best unmatched SSCM on unmatched samples. Furthermore, when considering the threshold value for the closest percentages between the number of matched and unmatched samples, "clips/mfaq" correctly finds more matched and unmatched samples compared to the best matched SSCM, however not as many unmatched samples as the best unmatched SSCM. The conclusion I can draw from these results is that without finetuning, "clips/mfaq" performs slightly better on matched results, but slightly worse on unmatched results, than SSCM. Specifically, when it is not finetuned on scenario specific data, the method does not reduce noise in matching open text input.

Now I show the results of the model after it has been finetuned on the scenario specific dataset created from the "SamenwerkenOT" scenario. The evaluation is done on the Testing and Training datasets, in order to compare the results with the results obtained before finetuning, for threshold values between 0.65 and 0.75:



(a) Finetuned model evaluated on Testing dataset

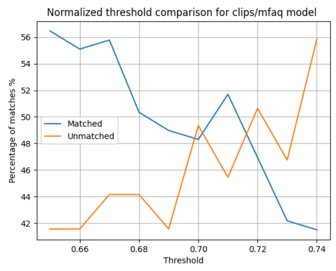


(b) Finetuned model evaluated on Training dataset

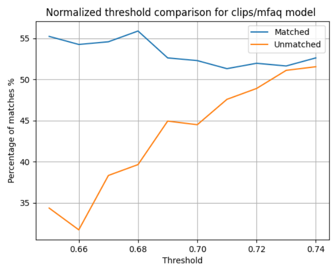
After the finetuning process, the model detects fewer matched and unmatched results overall. This shows that the finetuning process on the scenario specific data does not reduce noise. This may be due to the fact that the scenario specific dataset does not contain any unmatched training pairs. Specifically, all of the training pairs are created from the scenario tree, where the question field contains the computer statement and the answer field contains all possible statement options. Because of this, there are no training samples for unmatched statement options, so the model is not trained to find any unmatched results.

To alternatively test the finetuning process and determine what causes the drop in performance, I create new datasets for finetuning from the existing Training and Testing datasets. I call these datasets the finetuning Training and Testing datasets respectively. The finetuning datasets contain the player open text input, all of the prewritten statement options, and the golden match, which determines which statement option is most similar to the open text input provided by the player, or if none of them are.

The formatting of these finetuning datasets is similar, containing pairs with question and answer fields. However, unlike the scenario specific dataset, here the question field contains the player open text input, and the answer contains the prewritten statement option closest to the player open text input, given by the golden match. If the golden match is null, meaning that the player open text input is not similar to any of the statement options, the answer field of the pair has the value None. By structuring the finetuning datasets in this way, the embedding model should associate similar sentences together, and learn that some sentences can be associated with a None value, if none of the provided statement options are similar enough. If the model performs better after being finetuned on these finetuning datasets, it shows that the scenario specific dataset needs to contain unmatched samples as well as matched samples, like the Training and Testing datasets. These new finetuning datasets contain more training samples than the previously used scenario specific dataset. The finetuning Testing dataset contains 224 samples, and the finetuning Training dataset contains 533 samples. In order to pass through all the data, the number of steps in the finetuning process will be this sample number.



(c) Model finetuned on the finetuning Training dataset, evaluated on the Testing dataset



(d) Model finetuned on the finetuning Testing dataset, evaluated on the Training dataset

In the context of evaluating the word embedding model, I consider a true positive when the model correctly labels a matched sample, a true negative when the model correctly labels an unmatched sample as unmatched, a false positive when the model labels an unmatched sample as matched, and a false negative when the model labels a matched sample as unmatched.

As a final comparison, I will show the values of the true positives, true negatives, false positives, and false negatives in the following cases:

Firstly, when the model is evaluated on the Testing dataset, comparing being finetuned on the scenario specific dataset and the finetuning Training dataset:

Threshold	True Positives		True Negatives		False Positives		False Negatives	
Finetuning Dataset	Scenario	Training	Scenario	Training	Scenario	Training	Scenario	Training
0.65	78	83	31	32	72	63	43	46
0.66	80	81	27	32	72	60	45	51
0.67	79	82	37	34	59	61	49	47
0.68	85	74	35	34	56	64	48	52
0.69	77	72	35	32	57	65	55	55
0.7	65	71	34	38	66	60	59	55
0.71	74	76	40	35	53	57	57	56
0.72	66	69	37	39	60	52	61	64
0.73	58	62	37	36	59	56	70	70
0.74	64	61	42	43	53	50	65	70

Table 3: Model evaluated on the Testing dataset

As we can see, for a threshold value of 0.7, where the percentages of matched and unmatched samples are the closest, there are slightly more correctly labeled matched and unmatched samples for the case where the model was finetuned on the finetuning Training dataset. This can be attributed to the fact that the finetuning Training dataset has 533 samples to train, while the scenario specific dataset only has 23 samples. Furthermore, the finetuning Training dataset contains unmatched samples for the model to train on, while the scenario specific dataset does not.

Finally, I show the comparison between the evaluation of the model on the Training dataset, when it is finetuned on the scenario specific dataset and the finetuning Testing dataset:

Threshold	True Positives		True Negatives		False Positives		False Negatives	
Finetuning Dataset	Scenario	Testing	Scenario	Testing	Scenario	Testing	Scenario	Testing
0.65	181	169	64	78	264	261	24	25
0.66	165	166	80	72	250	264	38	31
0.67	169	167	83	87	254	250	27	29
0.68	161	171	84	90	251	234	37	38
0.69	168	161	104	102	215	227	46	43
0.7	157	160	100	101	225	225	51	47
0.71	160	157	108	108	215	216	50	52
0.72	159	159	110	111	207	208	57	55
0.73	150	158	116	116	206	197	61	62
0.74	161	161	124	117	186	194	62	61

Table 4: Model evaluated on the Training dataset

In this case, the unmatched results are slightly higher when the model is finetuned on the scenario specific dataset, while the number of correctly labeled matched samples remains the same. However, the differences are too small to be considered significant.

7 Discussion

In this section, I consolidate all of the findings from the previous chapters and discuss the results.

I was able to modify the matching service of the serious game Communicate to use a state-of-the-art word embedding model in the process of matching open text input to prewritten statement options. This word embedding model is called "clips/mfaq", and it is a RoBerta based word embedding model trained on a Dutch dataset. Comparing the results of this matching method with the previously introduced SSCM showed a slight increase in the number of correctly labeled matched samples, but a decrease in the number of correctly labeled unmatched samples, showing that noise was not reduced in this way. Future work for this section includes testing other state-of-the-art word embeddings that were not covered in the initial set. Word embeddings trained specifically on Dutch corpora are scarce, or the corpora are not large enough for good performance. Unfortunately, the method of finetuning the word embedding model with scenario specific information was not successful. This was most likely caused by the fact that the finetuning process was adapted from the original training method of the ”clips/mfaq” model, formatting the scenario specific dataset to resemble the original ”MFAQ” dataset used for training. Instead of this, in future work, the finetuning method should be created from scratch, and tailored to the scenario specific data. This would reduce computation time, since the scenario specific dataset is much smaller than the original ”MFAQ” dataset, and requires a much less complex method of finetuning.

Furthermore, the lack of noise reduction after the finetuning process might be attributed to the training dataset not having unmatched samples. This trains the model only on matched samples and does not leave an option for unmatched samples. There might be two possible solutions to this problem. The first solution is creating synthetic data for the finetuning process, that contains only unmatched samples to balance out the existing dataset. However, creating this data is not straightforward, since the sentences in the question part of the sample need to make sense in the context of the scenario, but should not be similar to any other sentences, to avoid unwanted matching. This is a complex task by itself and requires further research. Another solution for the problem is restructuring the data samples, so that an additional answer, an unmatched variant, is added to every sample. This means that the embedding model will always have the option to assess the similarity of a sentence to a null entry. However, this is problematic because in order to match with the null answer, the other answers in the sample need to be very dissimilar to the sentence, which is not always the case in the training data. This is also a complex process that needs further research.

8 Conclusion

In this section, I will state the conclusion of the findings of previous chapters, and I will elaborate on the future work that can be derived from taking this thesis as a starting point.

At the moment, state-of-the-art word embedding models can be integrated into the serious game Communicate, with comparable results to the previous methods, depending on the chosen similarity threshold. However, finetuning the models with scenario specific information requires further development, and possibly a rework of the format in which the data is preprocessed.

In answering the first research question, Which word embedding methods are suitable for sentence similarity in an application for training communication skills with open text input, after a literature review I chose a subset of 5 state-of-the-art word embedding models for further testing. Future work could include more models for testing on this particular NLP task, because, as discussed in Section 4, the general performance of word embedding models does not always correlate to their performance on a particular task. This means that while some other word embedding models are considered to generally perform worse, they might perform better on this particular sentence similarity task than the chosen set of 5. This process of testing a larger set of word embedding models from the Huggingface library could be automated, finding if other models have a better detection accuracy of matched and unmatched samples from the golden datasets.

Furthermore, instead of using the "clips/mfaq" model as a pretrained model, it is possible to retrain the model from scratch. Future work should explore this possibility. The serious game Communicate has a large number of authored scenarios that can be used as samples for a training process of the word embedding model. However, it is worth exploring if the model would perform better if it is trained on samples that contain, in addition to the prewritten statement options, an extra empty statement option. In this way, the model would be trained to match any samples that should be unmatched, to the null statement option. This would reduce the weight the threshold has in determining matched and unmatched samples.

I present a method of finetuning the pretrained word embedding model using scenario specific information. This method reuses code that has been used for the initial training of the model on the "MFAQ" dataset. Future work should focus on developing a new method of finetuning, rewriting code from scratch specifically for the format of the scenario specific information.

Furthermore, when evaluating the model after it has been finetuned on various datasets, I conclude that using datasets that contain unmatched samples for finetuning leads to a decrease in noise compared to evaluating the model after it has been finetuned on datasets without unmatched samples. Future work should focus on formatting scenario specific data so that unmatched samples are also contained in the scenario specific dataset. This teaches the model that there is always the option that the input text might not be similar to any of the prewritten statement options.

Materials

This section is dedicated to the link to the git repository where you can find all the code used in this thesis. Besides the python files, I also include the datasets mentioned in this thesis.

Git repository link: `https://github.com/Alias939/master_thesis_code.git`

References

[1] Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. "The Evaluation of Sentence Similarity Measures". In: *Data Warehousing and Knowledge Discovery*. Ed. by Il-Yeol Song, Johann Eder, and Tho Manh Nguyen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 305–316. ISBN: 978-3-540-85836-2.

[2] Eneko Agirre et al. "*SEM 2013 shared task: Semantic Textual Similarity". In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. Atlanta, Georgia, USA: Association for Computational Linguistics, June 2013, pp. 32–43. URL: `https://aclanthology.org/S13-1004`.

[3] Eneko Agirre et al. "SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity". In: **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Montréal, Canada: Association for Computational Linguistics, July 2012, pp. 385–393. URL: `https://aclanthology.org/S12-1051`.

[4] Eneko Agirre et al. "SemEval-2014 Task 10: Multilingual Semantic Textual Similarity". In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 81–91. DOI: `10.3115/v1/S14-2010`. URL: `https://aclanthology.org/S14-2010`.

[5] Eneko Agirre et al. "SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, June 2015, pp. 252–263. DOI: `10.18653/v1/S15-2045`. URL: `https://aclanthology.org/S15-2045`.

[6] Alan Akbik et al. "FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 54–59. DOI: `10.18653/v1/N19-4010`. URL: `https://aclanthology.org/N19-4010`.

[7] Khetam Al Sharou, Zhenhao Li, and Lucia Specia. "Towards a better understanding of noise in natural language processing". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. 2021, pp. 53–62.

[8] Felipe Almeida and Geraldo Xexéo. "Word embeddings: A survey". In: *arXiv preprint arXiv:1901.09069* (2019).

[9] Edgar Altszyler et al. “The interpretation of dream meaning: Resolving ambiguity using Latent Semantic Analysis in a small corpus of text”. In: *Consciousness and Cognition* 56 (Nov. 2017), pp. 178–187. ISSN: 1053-8100. DOI: `10.1016/j.concog.2017.09.004`. URL: `http://dx.doi.org/10.1016/j.concog.2017.09.004`.

[10] Julian Alvarez, Damien Djaouti, et al. “An introduction to Serious game Definitions and concepts”. In: *Serious games & simulation for risks management* 11.1 (2011), pp. 11–15.

[11] Keith Anderson et al. “The TARDIS framework: intelligent virtual agents for social coaching in job interviews”. In: *International Conference on Advances in Computer Entertainment Technology*. Springer. 2013, pp. 476–491.

[12] Lorin W. Anderson and David R. Krathwohl. *A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom’s Taxonomy of Educational Objectives.* 2nd ed. New York: Allyn & Bacon, Dec. 2001. ISBN: 978-0801319037.

[13] Mikel Artetxe and Holger Schwenk. “Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond”. In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 597–610.

[14] Jean-Pierre Briot et al. “A serious game and artificial agents to support intercultural participatory management of protected areas for biodiversity conservation and social inclusion”. In: *2011 Second International Conference on Culture and Computing*. IEEE. 2011, pp. 15–20.

[15] Paweł Budzianowski and Ivan Vulić. “Hello, It’s GPT-2 – How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems”. In: *arXiv preprint arXiv:1907.05774* (2019). arXiv: `1907.05774 [cs.CL]`.

[16] John A Casey. “Counseling Using Technology with At-Risk Youth. ERIC Digest.” In: *ERIC Clearinghouse on Counseling and Personnel Services Ann Arbor MI.* (1992).

[17] Meng-Tzu Cheng et al. “The use of serious games in science education: a review of selected empirical research from 2002 to 2013”. In: *Journal of computers in education* 2.3 (2015), pp. 353–375.

[18] Alexis Conneau et al. “Supervised Learning of Universal Sentence Representations from Natural Language Inference Data”. In: *arXiv preprint arXiv:1705.02364* (2018). arXiv: `1705.02364 [cs.CL]`.

[19] Zihang Dai et al. “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”. In: *arXiv preprint arXiv:1901.02860* (2019). arXiv: `1901.02860 [cs.LG]`.

[20] Zihang Dai et al. “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”. In: *arXiv preprint arXiv:1901.02860* (2019). arXiv: `1901.02860 [cs.LG]`.

[21] Maxime De Bruyn et al. "MFAQ: a Multilingual FAQ Dataset". In: *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1–13. DOI: 10.18653/v1/2021.mrqa-1.1. URL: https://aclanthology.org/2021.mrqa-1.1.

[22] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (2018). DOI: 10.48550/ARXIV.1810.04805. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[23] Henrik Engström, Jenny Brusk, and Patrik Erlandsson. "Prototyping tools for game writers". In: *The Computer Games Journal* 7.3 (2018), pp. 153–172.

[24] Kawin Ethayarajh. "How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings". In: *arXiv preprint arXiv:1909.00512* (2019). arXiv: 1909.00512 [cs.CL].

[25] George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler. *Computer Methods for Mathematical Computations*. Prentice Hall Professional Technical Reference, 1977. ISBN: 0131653326.

[26] Dion H. Goh, Rebecca P. Ang, and Hui Chern Tan. "Strategies for designing effective psychotherapeutic gaming interventions for children and adolescents". In: *Computers in Human Behavior* 24.5 (2008). Including the Special Issue: Internet Empowerment, pp. 2217–2235. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2007.10.007. URL: https://www.sciencedirect.com/science/article/pii/S0747563207001616.

[27] Juho Hamari, Jonna Koivisto, and Harri Sarsa. "Does gamification work?– a literature review of empirical studies on gamification". In: *2014 47th Hawaii international conference on system sciences*. Ieee. 2014, pp. 3025–3034.

[28] Zellig S. Harris. "Distributional Structure". In: *¡i¿WORD¡/i¿* 10.2-3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520. eprint: https://doi.org/10.1080/00437956.1954.11659520. URL: https://doi.org/10.1080/00437956.1954.11659520.

[29] Mohammed Hoque et al. "Mach: My automated conversation coach". In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 2013, pp. 697–706.

[30] Cathal Horan. "When not to choose the best NLP model". In: (2019).

[31] Johan Jeuring et al. "Communicate! — A Serious Game for Communication Skills —". In: 9307 (Jan. 2015), pp. 513–517. DOI: 10.1007/978-3-319-24258-3_49.

[32] JT Jeuring et al. "Processing open text input in a scripted communication scenario". In: *SEMDIAL 2018: The 22nd workshop on the Semantics and Pragmatics of Dialogue*. 2018, pp. 211–214.

[33] W Lewis Johnson, Stacy Marsella, and Hannes Vilhjalmsson. "The DARWARS tactical language training system". In: *Proceedings of I/ITSEC*. 2004.

[34] Yasmin B Kafai. "Gender differences in children's constructions of video games". In: *Interacting with video* 11 (1996), pp. 39–66.

[35] Fedwa Laamarti, Mohamad Eid, and Abdulmotaleb El Saddik. "An Overview of Serious Games". In: *International Journal of Computer Games Technology* 2014 (Oct. 2014), p. 1. ISSN: 1687-7047. DOI: 10.1155/2014/358152. URL: https://doi.org/10.1155/2014/358152.

[36] Raja Lala. "Scenarios in a serious game for training communication skills". PhD thesis. Utrecht University, 2021.

[37] Raja Lala, Johan Jeuring, and Marcell van Geest. "Scaffolding open text input in a scripted communication skills learning environment". In: *International Conference on Games and Learning Alliance*. Springer. 2019, pp. 169–179.

[38] Raja Lala et al. "Enhancing free-text interactions in a communication skills learning environment". In: *13th International Conference on Computer Supported Collaborative Learning*. International Society of the Learning Sciences. 2019, pp. 877–878.

[39] Raja Lala et al. "Scenarios in virtual learning environments for one-to-one communication skills training". In: *International Journal of Educational Technology in Higher Education* 14 (Apr. 2017), p. 17. DOI: 10.1186/s41239-017-0054-1.

[40] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. "An introduction to latent semantic analysis". In: *Discourse Processes* 25.2-3 (1998), pp. 259–284. DOI: 10.1080/01638539809545028. eprint: https://doi.org/10.1080/01638539809545028. URL: https://doi.org/10.1080/01638539809545028.

[41] H Chad Lane et al. "Coaching intercultural communication in a serious game". In: *Proceedings of the 16th International Conference on Computers in Education*. Citeseer. 2008, pp. 35–42.

[42] Md Tahmid Rahman Laskar, Jimmy Xiangji Huang, and Enamul Hoque. "Contextualized Embeddings based Transformer Encoder for Sentence Similarity Modeling in Answer Selection Task". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 5505–5514. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.676.

[43] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv preprint arXiv:1907.11692* (2019). DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692.

[44] Laura M van der Lubbe et al. “A serious game for training verbal resilience to doorstep scams”. In: *International Conference on Games and Learning Alliance*. Springer. 2018, pp. 110–120.

[45] H. P. Luhn. “A Statistical Approach to Mechanized Encoding and Searching of Literary Information”. In: *IBM Journal of Research and Development* 1.4 (1957), pp. 309–317. DOI: `10.1147/rd.14.0309`.

[46] Michael Mateas and Andrew Stern. “Natural language understanding in Façade: Surface-text processing”. In: *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*. Springer. 2004, pp. 3–13.

[47] Chris McCollum et al. “Developing an immersive, cultural training system”. In: *The Interservice/Industry Training, Simulation and Education Conference(I/ITSEC)*. 2004, p. 2004.

[48] David R. Michael and Sandra L. Chen. *Serious Games: Games That Educate, Train, and Inform*. Muska & Lipman/Premier-Trade, 2005. ISBN: 1592006221.

[49] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv preprint arXiv:1301.3781* (2013). arXiv: `1301.3781 [cs.CL]`.

[50] Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. “Comparative study of word embedding methods in topic segmentation”. In: *Procedia Computer Science* 112 (2017). Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France, pp. 340–349. ISSN: 1877-0509. DOI: `https://doi.org/10.1016/j.procs.2017.08.009`. URL: `https://www.sciencedirect.com/science/article/pii/S1877050917313480`.

[51] Magalie Ochs et al. “Training doctors’ social skills to break bad news: evaluation of the impact of virtual environment displays on the sense of presence”. In: *Journal on Multimodal User Interfaces* 13.1 (2019), pp. 41–51.

[52] A.A. Patel and A.U. Arasanipalai. *Applied Natural Language Processing in the Enterprise*. O’Reilly Media, 2021. ISBN: 9781492062547. URL: `https://books.google.ro/books?id=5dktEAAAQBAJ`.

[53] Jeffrey Pennington, Richard Socher, and Christopher Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: `10.3115/v1/D14-1162`. URL: `https://aclanthology.org/D14-1162`.

[54] Matthew E. Peters et al. “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: `10.18653/v1/N18-1202`. URL: `https://aclanthology.org/N18-1202`.

[55] M. Prensky. “Computer games and learning: Digital game-based learning”. In: *Handbook of Computer Game Studies* (Jan. 2005), pp. 97–122.

[56] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[57] Olivia Realdon et al. “Learning communication skills through computer-based interactive simulations”. In: *From communication to presence: Cognition, emotions and culture towards the ultimate communicative experience* (2006).

[58] Nils Reimers and Iryna Gurevych. “Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation”. In: *arXiv preprint arXiv:2004.09813* (2020). DOI: `10.48550/ARXIV.2004.09813`. URL: `https://arxiv.org/abs/2004.09813`.

[59] Xin Rong. “word2vec Parameter Learning Explained”. In: *arXiv preprint arXiv:1411.2738* (2014). arXiv: `1411.2738 [cs.CL]`.

[60] Kaveh Taghipour, Shahram Khadivi, and Jia Xu. “Parallel corpus refinement as an outlier detection algorithm”. In: *Proceedings of Machine Translation Summit XIII: Papers*. 2011.

[61] Andrea Tartaro and Justine Cassell. “Playing with virtual peers. Bootstrapping contingent discourse in children with autism”. In: *ICLS 2008* (2008).

[62] Joseph Turian, Lev Ratinov, and Yoshua Bengio. “Word representations: a simple and general method for semi-supervised learning”. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. 2010, pp. 384–394.

[63] Frederik Vaassen and Walter Daelemans. “Emotion classification in a serious game for training communication skills”. In: *LOT Occasional Series* 16 (2010), pp. 155–168.

[64] Ashish Vaswani et al. “Attention Is All You Need”. In: *Advances in neural information processing systems* 30 (2017). arXiv: `1706.03762 [cs.CL]`.

[65] Wim Westera et al. “Artificial intelligence moving serious gaming: Presenting reusable game AI components”. In: *Education and Information Technologies* 25.1 (2020), pp. 351–380.

[66] Claes Wohlin. “Guidelines for snowballing in systematic literature studies and a replication in software engineering”. In: *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. 2014, pp. 1–10.

[67] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal component analysis”. In: *Chemometrics and Intelligent Laboratory Systems* 2.1 (1987). Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists, pp. 37–52. ISSN: 0169-7439. DOI: https://doi.org/10.1016/0169-7439(87)80084-9. URL: https://www.sciencedirect.com/science/article/pii/0169743987800849.

[68] Thomas Wolf et al. “HuggingFace’s Transformers: State-of-the-art Natural Language Processing”. In: *arXiv preprint arXiv:1910.03771* (2019). DOI: 10.48550/ARXIV.1910.03771. URL: https://arxiv.org/abs/1910.03771.

[69] Zhilin Yang et al. “XLNet: Generalized Autoregressive Pretraining for Language Understanding”. In: *Advances in neural information processing systems* 32 (2020). arXiv: 1906.08237 [cs.CL].