北海道大学
HOKKAIDO UNIVERSITY

Language Media Laboratory
Research Group of Information Media Science and Technology
Division of Media and Network Technologies
Graduate School of Information Science and Technology
Hokkaido University

Patrycja Ewa Swieczkowska

A Study on Advisory Online Comments for a Motivational Dialogue System

A doctoral dissertation
supervised by
prof. Kenji Araki

Sapporo 2020

Table of Contents

Thesis abstract in English

The lack of motivation in daily life is a pressing problem in the modern world. More and more people are unmotivated at school, at work, and in their daily efforts such as exercise. Therefore, the ultimate goal of this study is to create a dialogue system that would give motivational advice to users. In my previous studies, I confirmed that this task is not trivial. Specifically, I developed a motivational dialogue system that would support the user in their everyday work. The system was supposed to motivate the user with a casual dialogue and a friendly attitude, but during evaluation, the motivation function averaged just 3.9 points and the dialogue function averaged only 3.1 points on a five-point scale. In other words, a nice conversation and a kind attitude were not enough to motivate the users to do their work. It became clear to me that an effective motivational dialogue system needs to have a module specifically dedicated to combating the user's lack of motivation. The research presented in this thesis was conducted to develop such a module.

To achieve the objective, first it was necessary to analyze the nature of advisory and motivational texts. This was important for several reasons described below. First, a corpus of training data for a motivational dialogue system can be created by collecting only texts that have advisory or motivational content and eliminating any other texts. To collect as much data as possible, it is best to use numerous data sources. Online sources such as "Oshiete!goo" usually have labels (whether the text is advisory or not) or some score (telling others how good the advice is), but there exist many other sources such as blogs or general discussion websites where no labels are available. In order to use such unlabeled data, it is necessary to remove non-advisory texts, that is, noise. Then, having accomplished that, one should analyze the gathered data and extract only the texts containing good advice. These goals can be achieved using the classification algorithm and ranking algorithm proposed in this thesis. Secondly, clarifying what an advisory or motivational text is yields valuable information about the kind of utterances the motivational dialogue system should generate.

This thesis describes a study on the nature of motivational and advisory texts. The main contributions of this study are threefold: the discovery of features characterizing advisory texts, detailed analysis of those features, and two algorithms that can classify and rank advisory data. In the field of natural language processing, there is no previous

research on motivational advice analysis, so the main novelty of this research is to propose algorithms that can handle the processing of such advisory data.

For analysis, I used online comments downloaded from the discussion platform Reddit. I only used the threads in which someone asked for advice, so the comments were bound to contain motivational and advisory content. On Reddit, each comment is scored by other users according to its general quality, so I was able to collect only comments providing good, valuable advice.

After gathering the data, the comments were thoroughly analyzed. As a result of this analysis, I created a set of features characterizing advisory texts. In general, best-rated comments in the dataset contained numerous imperative and advisory expressions, and the advice given in these comments was very specific. Furthermore, the person who gave the advice often used to have the same problem in the past and was able to relate to the troubled user who created the thread. I operationalized these characteristics into a few different features; on top of that, I performed sentiment analysis on the comments. In the end, there were 14 advice features. Additionally, I used distributed word representations created with word2vec. Word2vec is an important tool in natural language processing and allows us to appropriately express the meaning of words in the form of word vectors. By incorporating word2vec representations into the study, I was able to analyze the semantic features of the comments as well.

The effectiveness of the features was confirmed in two tasks using neural networks. The first task was classification with the goal of removing noise from the data. For this task, two shallow neural networks were combined to distinguish texts containing advice from regular texts. The second task was advice ranking with the goal of selecting good quality data. Using a convolutional neural network, motivational and advisory texts were ranked within groups of three. In the case of classification, a feedforward network was appropriate, but a different approach was required for the ranking. In a feedforward network, each feature undergoes slightly different calculations, whereas in a convolutional neural network, groups of features can be processed in the same way together by using a filter. In the case of the ranking task, features of each comment in the group had to be computed in the same way in the first layer, so that each comment had equal chances in the ranking regardless of input order. This was accomplished by using a filter with size equal to the group of features that came from each single comment.

As mentioned above, the analysis of motivational advice has not been previously studied in natural language processing, so there was no existing baseline for my research.

However, I performed numerous experiments and gradually improved the results, so my first experiment can be regarded as a baseline. In the classification task, I initially used a Support Vector Machine and a shallow feedforward network, and the last experiments were performed with my proposed method. In this method, I chained two feedforward neural networks, each of which used one of the feature sets: the 14 advice features or the word2vec features. First, the data was classified by one network, and then the data labeled as not advisory was deleted from the dataset. The remaining data was subsequently classified by the other network. The order in which the networks are applied can be freely decided based on the research objective. F-score was 0.760 in the first classification experiment, but in the final experiment it improved to 0.943–0.971, depending on the network order. Similarly, precision increased from 0.844 to 0.965–0.977. Consequently, it was confirmed that the proposed method can effectively remove noise from the data. In the ranking experiment, I obtained accuracy of 0.971. In this experiment, the 14 advice features and the word2vec features were used together, so for a baseline I created a convolutional network that used only the word2vec features. The accuracy of that network was 0.885, which is considerably lower than the proposed method.

My studies confirmed the effectiveness of the 14 advice features and the two proposed methods. I also created a corpus of advisory and motivational texts that can be used in other research projects. Finally, I obtained a lot of knowledge about the nature of motivational and advisory texts. Therefore, this study has accomplished its goals and completes the first step towards developing a motivational dialogue system.

Thesis abstract in Japanese

動機付け対話システムのためのオンラインコメントにおける アドバイスに関する研究

学位論文内容の要旨

日常生活における動機付けの欠如は，現代の世界における差し迫った問題である．ますます多くの人々が，学校や職場で，また運動などの日々の努力において，やる気を失っている．したがって，この研究の最終的な目的は，ユーザにやる気を起こさせるアドバイスを与えるための動機付け対話システムを構築することである．これまでの研究で私は，このタスクが単純ではないということを確認し，日常的なタスクのための動機付け対話システムの開発を行った．この対話システムは気軽な対話や懇切な態度でユーザにモチベーションを与える予定であったが，評価を行った結果，5 ポイントのスケールで，動機付け機能が平均 3.9，対話機能が平均 3.1 と評価された．すなわち，効果的にやる気を起こさせる対話システムは，対話の気軽さ・親切さだけでは不十分で，ユーザの動機付けの欠如に対して動機付けを起こさせる会話を行うという機能が不可欠だということが明らかとなった．そこで，このような機能を開発するために本研究を行った．

以上の目的を達成するためには，まず，助言および動機付けのテキストが持つ性質を明らかにする必要がある．これは以下に述べるいくつかの点で有効である．まず，助言的または動機付けが存在するテキストのみを収集することにより，このような目的を持つ対話システムの学習データのコーパスを作成できる．できるだけ膨大なデータセットを収集するために，多くのデータソースを利用できることが望ましい．「教えて!goo」のようなデータであればラベル（アドバイスかどうか）や評価（どのぐらい良いアドバイスであるか）が付与されているが，ブログや一般的なディスカッションサイトの場合にはラベル無しデータが存在している．それらのラベル無しのデータを利用するためには，まずアドバイスではないテキスト，つまりノイズを除去する必要がある．その後，アドバイスの中から性質の良いもののみを抽出する必要がある．本論文で提案する分類アルゴリズムとランク付けアルゴリズムを用いて，前述したような目的を達成することができるものと考えられる．第二に，助言的または動機付けのテキストがどのようなものかを明らかにした上で，システムがどのような発話を生成すべきかについて重要な資料となる．本論文では，動機付けと助言的なテキストの性質に関する研究について述べる．以上より，本研究の主な成果は，助言的テキストの特徴の発見，その特徴の詳細な分析，あるいは助言的データを分類・ランク付けできる 2 つのアルゴリズムを提案したという 3 点である．自然言語処理分野において，動機付けをするアドバイスの分析は先行研究がほとんど存在しないため，その分析と助言的データを処理できるアルゴリズムを提案したことが本研究の新規性である．

分析のために使用したのは，ディスカッションプラットフォーム Reddit からダウンロードしたオンラインコメントである．ユーザがアドバイスを求め

た投稿のみを使用したため，コメントは動機付けまたは助言を含んでいる．コメントのレベルは，他のユーザが付与したスコアで知ることができる．したがって，最高の助言と動機付けを行ったコメントのみを収集することとした．

データを収集した後，コメントの内容を分析し，その性質について考察を行った．分析した上で，助言テキストの特徴な素性のセットを作成した．一般的に，最高のスコアが付与されたコメントには，多くの命令型およびアドバイス表現が含まれており，与えられたアドバイスは非常に具体的であることが確認された．さらに，アドバイスを与えた人は，過去に同じ問題を抱えていたためその問題をよく理解することができた場合が多いことが明らかになった．これらの性質をいくつかの素性として作成した上でセンチメント分析も行った．結果的に 14 のアドバイス素性が存在した．さらに，word2vec の分散表現の利用も行った．Word2vec は自然言語処理において重要なツールで，単語の意味を分散表現で適切に表現するものである．この分散表現を素性として利用することで単語の意味的特徴も利用することができる．

素性の有効性は，ニューラルネットワークを用いて 2 つのタスクにより確認を行った．最初のタスクは，データからノイズを除去するための分類である．2 つの浅いニューラルネットワークを組み合わせて，アドバイスを含むテキストと通常のテキストの分類を行った．2 つ目のタスクは，性質の良いデータを選択するためのアドバイスのランク付けである．畳み込みニューラルネットワークを使用して，テキストが 3 つというグループ内で動機付けおよび助言テキストのランク付けを行った．分類タスクでは，順伝播型ネットワークが相応しかったが，ランク付けタスクでは別のアプローチが必要となった．順伝播型ネットワークでは，各素性がわずかに異なる計算を行うのに対し，畳み込みニューラルネットワークでは，フィルターで素性のグループの計算を同様に行うことができる．ランク付けタスクの場合，入力順に関わらず各コメントがランキングで同等の評価を受けるように，コメントグループ内の各コメントの素性が最初のレイヤーで同じ計算を行う必要がある．そのため，各コメントの素性グループに同じサイズのフィルターを使用した．

前述したように，動機付けアドバイスの分析は，自然言語処理分野でこれまで研究されていなかったため，今回の実験でベースラインは存在しない．しかし，私の実験は段階的に実験を行い，徐々に改善を行っている．そのため最初の実験をベースラインと考えることができる．分類実験では，最初に SVM と順伝播型ネットワークを使用し，最後的に提案手法で実験を行った．提案手法としては，14 のアドバイス素性セットと word2vec の素性セットを個別に使用している 2 つの順伝播型ネットワークを連鎖させた．最初にデータを一つのネットワークで分類し，アドバイスでないとラベル付けられたものをデータから削除し，その残りのデータを次のネットワークで分類するという流れである．なお，ネットワークを適用する順序は任意である．最初の分類実験では，F 値が 0.760 であったが，最終実験では，ネットワークの順番により 0.943〜0.971 に向上した．同様に，適合率は 0.844 から 0.965〜0.977 に向上した．すなわち，提案手法がデータからノイズを効果的に除去できることが確認された．また，ランキング実験では 0.971 の精度が得られた．この実験では 14 のアドバイス素性と word2vec の素性を合わせて使用したので，ベースラインとしては word2vec の素性のみを使用する畳み込みネットワークが考えられる．

そのネットワークの精度は 0.885 であった．これは，提案手法よりかなり低いものである．

本研究により，14 のアドバイス素性の有効性及び 2 つの提案手法の有効性が確認された．さらに，モチベーションに関する他の研究プロジェクトで使用できるような助言と動機付けのテキストのコーパスの開発を行った．また，動機付けおよび助言テキストの性質について多くの知見を得ることができた．したがって，本研究では動機付けの対話システムを開発するための最初の段階を遂行したと考えられる．

Chapter 1: Introduction

1.1 Background and motivation

Motivation is an important contemporary issue, mostly researched within the context of professional life (Badubi, 2017; Gerhart and Fang, 2015; Kultalahti and Liisa Viitala, 2014), academic life (Elmelid et al., 2015; Litalien, Guay, and Morin, 2015) or mental health problems such as mood disorders (Fussner, Mancini and Luebbe, 2018; Hershenberg, 2017). There are numerous websites specifically targeted at people who feel the lack of motivation in their lives and search online for help (for example HealthDirect, 2018[1]; Nussbaum, 2017)[2]. The online discussion platform *Reddit: Get Disciplined*[3], which defines its purpose as *A [place] for people who have issues with procrastination, motivation and discipline*[4], has as much as almost half a million subscribers as of November 2019. All this shows the magnitude of the problem of being unmotivated and the need and desire to solve that problem in today's society.

Considering the above, creating a dialogue system capable of motivating the user to do their work or to study would be beneficial for multiple reasons. First, it would provide users with immediate motivational support and would eliminate the need to post to discussion forums like *Reddit: Get Disciplined* and wait for replies, sometimes days at a time. Often submissions that were posted at unfortunate times get overlooked by others and the user does not receive any help at all. On the other hand, a dialogue system would cater to the particular user's needs, and give them advice in a timely manner. Secondly, such a system would contribute to the field of HCI (*human-computer interaction*), because the bond formed between a supportive dialogue system and its user could imitate a bond formed between friends. The user would rely on the system for motivation and in return would be provided with appropriate advice. Finally, building such a system requires finding out which advice is applicable to which problems, therefore contributing to the research in commonsense knowledge. Unfortunately, the concept of motivation has not been much studied in the field of dialogue systems.

[1] https://www.healthdirect.gov.au/motivation-how-to-get-started-and-staying-motivated

[2] https://greatist.com/grow/motivation-tips-that-work

[3] https://www.reddit.com/r/getdisciplined/

[4] Throughout this thesis, italics is used for all direct quotes (such as descriptions of Reddit subpages) or any other text that needs to be distinguished in the overall narrative, such as file or network layer names, data categories or explanations of abbreviations. Bold font is used for emphasis otherwise.

1.2 Research objectives

The ultimate goal of this research is to create a dialogue system that would motivate its user to complete tasks on their schedule, regardless of the type of task or reason for being unmotivated, by giving motivational advice expressed in natural language. The input to the system would be the user's description about their lack of motivation, and the output would be a response meant to give the user some advice pertaining to the problem at hand. In this research, I define motivational advice as **an utterance in natural language pertaining to the user's problem with being unmotivated and containing a possible solution to that problem**. The target users are therefore people struggling with motivation in their everyday life.

Figure 1 illustrates an example of system input and output.

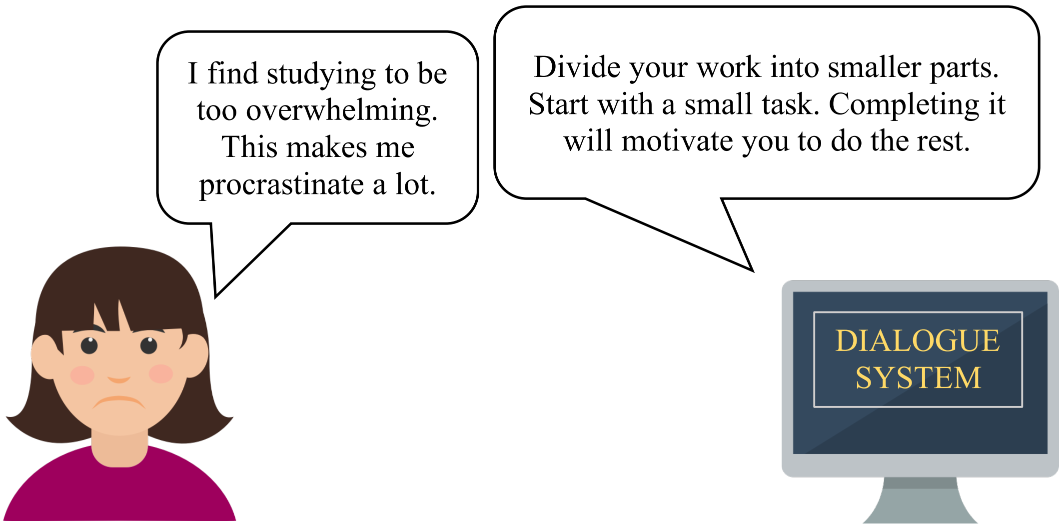

Figure 1: Example input and output to the motivational dialogue system.

Such a dialogue system needs an enormous amount of training data to be able to produce its own utterances. This data is not readily available, because there is no corpus of such texts. Instead, data can be harvested from various different sources, for example from online discussion platforms or motivational blogs. However, in that case it is crucial to eliminate noise, which may constitute a large portion of the obtained data.

Therefore, in this thesis I describe two algorithms that allow for effective sorting of data and extracting only the best quality texts. The first algorithm is a classifier that chooses only those texts that contain advice, particularly motivational advice. The second algorithm is a ranking network that is able to rank texts based on the quality of advice that they contain. This is a first step in the direction of creating the final motivational dialogue system.

1.3 Contributions of the thesis

There are several scientific contributions of this thesis.

First, I provide two algorithms that can be used to sort out noise in textual data that would be used as training data for the motivational dialogue system. Both algorithms perform above 90% in F-score and can be successfully utilized for the purpose of cleaning data from noise. Moreover, both of them are unique in their purpose; up to this date, there are no other algorithms for classification or ranking of advice texts. Finally, both algorithms required original solutions so that they could work best. The classifier combines two feature sets in two separate shallow neural networks. The ranking algorithm utilizes a convolutional network in place of a recurrent network to solve problems with the latter overfitting to the training set.

Second, in my studies I used a corpus of motivational texts that I gathered myself from specific subpages of the discussion platform Reddit[5], where people talk about various topics, including advice. This corpus – or the method of its creation – can be used for further studies concerning advice in general and motivational advice in particular.

Third, the algorithms mentioned above are based on a feature set that I composed myself. Some of those features were adapted from studies on different text classification tasks; others were hand-crafted by myself. The entire set of 14 features has not been used in any other studies before. It has been specifically created for this research only.

Fourth, the error analysis of experiments performed with the aforementioned algorithms and features provided a lot of insights into the nature of advice in general and motivational advice in particular. This includes studies on sentiment analysis, manner of expression and specificity of the given advice.

Finally, to the best of my knowledge, this is the only project currently in development which prepares ground for a dialogue system that motivates people to perform all kinds of tasks using natural language, imitating an actual human-to-human conversation.

[5] https://www.reddit.com

1.4 Structure of the thesis

This thesis consists of six chapters apart from Introduction.

Chapter 2 describes related research in the field of human-computer interaction (HCI), user motivation and advisory systems. This chapter is meant to discuss research related to the ultimate goal of this study, which is creating a motivational dialogue system. Research related to specific sub-goals such as advice classification and ranking are discussed in their respective chapters.

Chapter 3 details my work on an electronic assistant that was meant to help users organize their work. Findings from that research, especially those concerning motivation, have inspired me to pursue the goal of creating a dialogue system that gives motivational advice.

Chapter 4 presents the datasets that served as basis for the studies presented in Chapter 5 and Chapter 6. The datasets came from an online platform Reddit. Chapter 4 gives some insight into the nature of Reddit posts and comments and explains why they were chosen as a data source.

Chapter 5 presents my classification algorithm. This chapter contains the most insight into the nature of advisory and motivational texts. The final algorithm is a result of numerous experiments, all of which are shown in the chapter together with thorough error analysis at each step of the research.

Chapter 6 presents my ranking algorithm. At that stage I was building on the experience already gained during experiments described in Chapter 5. This is why Chapter 6 is shorter, but it still contains valuable analysis of advisory and motivational texts that was not possible with only the classification algorithm.

Chapter 7 summarizes my findings and presents future directions for the study.

Chapter 2: Related research

2.1 Human-computer interaction (HCI)

Emotional intelligence can be defined as, among others, the capability of individuals to recognize emotions of others and use emotional information to guide thinking and behavior (Coleman, 2008). In the context of various agents, from cognitive architectures to dialogue systems, this means being able to respond to various emotional states of the user, which already became an important research topic (Bickmore and Picard, 2005; Callejas and Griol, 2011). The authors of (Bickmore and Picard, 2005) call systems capable of this *relational agents* and define them as *computational artifacts designed to build long-term, social-emotional relationships with their users*. They recognize the importance of implementing human-like emotional intelligence in machines, which in case of dialogue systems is especially crucial, as they are specifically designed to interact with humans.

Likewise, a motivational dialogue system would establish a unique relationship with its user by supporting them emotionally with motivational advice. It is reasonable to assume that the system would be able to influence the user's behavior and habits regarding motivation, thus resulting in long-term lifestyle changes for the user. Agents that already accomplished similar feats include Project RAISE, where participants were encouraged to exercise and protect themselves from sunrays (Sillice et al., 2018). Another study concluded that the field of mental health and psychiatry could benefit from utilizing chatbots in screening, diagnosing and treating mental illness (Vaidyam, Wisniewski, Halamka, Kashavan and Torus, 2019). There is even research into personality interaction between the human and the agent and the effect it has on the conversation (Zhou, Mark, Li and Yang, 2019). All this proves that research into relational agents is a growing field and will develop in the future. This is why it is important to create agents such as motivational dialogue systems.

2.2 User motivation

While there exist papers suggesting various approaches to influencing motivational states in users, usually they do not contain experiments confirming their hypotheses. Callejas and Griol (2016) propose a model for characterizing user motivation by applying psychological research, for example Attribution Theory (Weiner, 1982), which stipulates that the outcomes of an individual's actions influence both what that individual perceives as contributing factors to the actions, and the emotional responses to those actions. Tielman, Neerincx and Brickman (2017) describe a virtual agent that uses various motivational strategies aimed at PTSD patients, based on their therapy progress and trust in success of the treatment. He, Greenberg and Huang (2010) researched types of feedback that would be effective for people at different stages of readiness, willingness and ability to change in motivating them to save electrical energy. Grigore, Pereira and Scassellati (2015) construct a model for user motivation in human-robot cooperation to help the user achieve daily physical activity goals. They propose an adaptive robot companion that models the user's motivation daily and changes its strategies accordingly. Fogg (2009) presents a user behavior model for developing persuasive technologies, defined with three components: motivation, ability and triggers. In his view, target behavior takes place where both motivation and ability to perform the task are high and triggers are present. Oinas-Kukkonen (2013) describes the basics for creating persuasive systems aiming to change the user's behavior. As a core construct, the author proposes an outcome/change matrix for desired actions, which should serve as a guide for determining design goals for behavior change support systems. However, none of these papers present any proof that the theory is applicable and effective in practice.

Empirically verified studies include motivating users to do indoor-cycling every day for a specified period of time with a robot companion by building and maintaining a human-computer relationship (Sussenbach et al., 2014) or encouraging users to perform longer planking exercise by giving them acknowledging feedback from a robot that exercised together with them (Schneider and Kummert, 2016). However, in both cases the dialogue with the agent was scripted and limited to a few topics. Moreover, both studies dealt only with exercise and their very specific results cannot be generalized to other everyday tasks. Kaptein, de Ruyter, Markopulous and Aarts (2012) describe a study where subjects were being persuaded to reduce snacking via personalized short text messages. The messages were tailored to the user based on the user's score on the Susceptibility to Persuasion Scale (Kaptein, Markopulous, de Ruyter and Aarts, 2009). However, once again these messages were all crafted by the researchers and involved no natural language processing.

There are some applications that use gamification as a motivational mechanism, for example Habitica (Habitica, 2019)[6] or Todoist Karma (Doist Team, 2019)[7]. They treat accomplishing tasks as a game, but they do not act as dialogue systems that communicate with the user based on the user's needs. There is little to no dialogue in the interaction.

As is evident from the above literature review, the research presented in this thesis is original and, to the best of my knowledge, has not been attempted before.

2.3 Advisory systems

The work in advisory systems usually concerns giving advice on automatic or semi-automatic operation of various vehicles. Nguyen, Kim, Dang, Moon, and Hong (2016) propose a method of optimizing the speed of autonomous cars with respect to traffic density and traffic lights to reduce travel time and CO_2 emissions. Tonosaki et al. (2016) devise a method of saving energy in train operation that takes place in the dense railway network of Japan. Applications of advisory systems oriented towards average users include an interactive system that advises the user on weather conditions at his or her desired location (United States Patent No. 9,204,252, 2015). There are also advisory systems for legal matters (Greenleaf, Mowbray and Chung, 2018), medical information (United States Patent No. 9,005,119, 2015) or dietary habits (Fallaize, Franco, Hwang and Lovegrove, 2019).

However, to the best of my knowledge, there are no existing advisory systems that would support the user by giving them general life advice concerning being unmotivated. The only research projects with somewhat similar goals have already been discussed in Section 2.2 of this thesis. This reinforces my point about the originality of my research.

Motivational advice is difficult to give without constructing a detailed user model. All the studies mentioned above were able to profile users by gathering information such as their eating habits or location. However, for motivational advice more nuanced knowledge is needed. The system would have to gather information about the user's life situation, their working or studying habits and their mental health. This could be achieved by establishing a relationship with the user, as described in Section 2.1 on relational agents.

[6] https://habitica.com/static/home

[7] https://en.todoist.com/karma

My future research on this topic will utilize studies presented in all three sections of this chapter. In fact, I have already built a dialogue system that was aimed at influencing motivational states in users while also establishing a working relationship with them. This dialogue system and all findings from its evaluation are presented in the next chapter.

Chapter 3: Motivating personal assistant *Asystent*

The main goal of the research described in this chapter was to assist the user in completing their everyday tasks while also making sure the user was sufficiently motivated to do the work. The system was meant as a testing tool to verify my views about users' preferences for motivational agents.

This chapter is organized as follows. Section 3.1 presents related work on similar dialogue systems acting as electronic assistants. Section 3.2 characterizes the *Asystent* dialogue system. Section 3.3 presents the method and results of the evaluation. Section 3.4 discusses the results and includes remarks about further research directions. Section 3.5 explains the impact of *Asystent* on my research into motivational dialogue systems and shows the direction that I took from that point on.

3.1 Related work

Research in dialogue systems is a vast and quickly developing field. In the last few years there have been proposed many innovative systems meant to interact with the user. Some of them are meant for general use and employ complicated algorithms for information retrieval to stay relevant to the topic of the conversation, for example (Yan, Song, Zhou and Wu, 2016) or (Higashinaka et al., 2014). Other rely on spoken dialogue and the information that can be extracted from the speaker's utterances about their intentions, for example (Liu and Lane, 2016). There are also dialogue systems that are meant to learn specifically through interacting with users online, like the one presented in (Liu, Tur, Hakkani-Tur, Shah and Heck, 2018). Smartphone-compatible dialogue systems, such as Siri (Apple, 2019)[8], Cortana (Microsoft, 2019)[9] or Google Assistant (Google, 2019)[10] are widely used, and chatbots are becoming increasingly popular with businesses (Business Insider, 2016)[11]. In recent years, virtual assistants like Alexa (Amazon, 2019)[12] have also gained popularity.

The dialogue system described in this chapter acted as both an assistant and a calendar. Related work in the field includes systems like SelfPlanner described in

[8] https://www.apple.com/ios/siri/

[9] https://www.microsoft.com/en-us/cortana

[10] https://assistant.google.com/

[11] http://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12/

[12] https://www.alexa.com/

(Refandis and Alexiadis, 2011). This system was a web-based calendar application integrated with Google Calendar and Google Maps that introduced a new method of planning and rearranging tasks, especially with respect to meetings. Another program similar to *Asystent* was Timeful (Bank, Ariely and Shoham, 2012)[13]. It was a smartphone application acting as an electronic organizer, later acquired by Google to cooperate with Google Calendar. The program tailored itself to the user and over time learned their working habits to better plan their tasks. Another notable contribution to the field was the CALO project (DARPA, 2019)[14] aimed at creating an intelligent electronic assistant for the user. Apple's Siri is actually a spin-off of this project. A more recent calendar application is SuperCaly (Yoon et al., 2017), which gathers a variety of information such as personal messages and user location to help the user organize their daily life. There are also applications using gamification as a means to motivate the user to complete tasks (Doist Team, 2019; Habitica, 2019), which were already mentioned in Section 2.2 of this thesis.

Other research in electronic calendars includes, among others, methods of managing time between meetings in different geographic locations (United States Patent No. 8,712,810, 2014), effectively sharing calendars between family members (Eschler et al., 2015) or coworkers (Toxtli, Monroy-Hernandez and Cranshaw, 2018), adding tasks to the calendar via voice commands (United States Patent No. 8,369,493, 2013), improving visual representation of the calendar interface (United States Patent No. D754,692, 2016; United States Patent No. 9,436,934, 2016) or extracting relevant information about the tasks from an ontology rather than directly from the user to improve interaction with the calendar application (Agnatis, Alexiadis and Refandis, 2016).

The survey of the field shows that the main focus is usually to improve planning strategies and techniques in the calendar applications or making the applications more usable. However, even if the user is provided with a work plan and the calendar application is well-designed and accessible, the user may still not complete the tasks. As stated before, there has been little to no research into actually motivating the user to work apart from gamification, which does not include conducting a natural language conversation with the user. The dialogue system described in this chapter, as well as the final system of this research will hopefully be able to fill this gap.

[13] www.support.timeful.com

[14] http://www.ai.sri.com/project/CALO

3.2 Dialogue system *Asystent*

The dialogue system presented in this chapter is a prototype of an electronic assistant. The main goal of the system is to test the users' general preferences about being motivated by an artificial agent and to indicate possible future developments.

This section presents the dialogue system. The following subsections describe in turn the smaller goals of the system (Section 3.2.1), the program's technical specification and implementation (Sections 3.2.2 and 3.2.3 respectively), and the cooperation taking place between the system and the user (Section 3.2.4).

3.2.1 Goals of this research

The system was aimed at people who needed motivation in their everyday work and who would like to have a personal assistant helping them with that work. An example user could be a person who prepared a work plan, but would easily forget about the tasks or just did not feel like completing them. The dialogue system served as a motivating assistant for such users to ensure that they actually did the work. The name *Asystent* is Polish for *assistant*.

The smaller goals of the proposed system were defined as follows:

- Helping the user create the work plan.
- Reminding the user about upcoming tasks.
- Motivating the user to complete the tasks.
- Conducting a natural and friendly conversation with the user.
- Establishing a satisfying cooperation with the user.

3.2.2 General technical information

Asystent was a text-in-text-out dialogue system written for the Polish language and was designed according to the structural approach to dialogue systems. As such, it relied on finite automata and had a regular dialogue structure (Sadek and de Mori, 1998). The initiative in the dialogue was entirely on the system and the conversation was scripted. The dialogue between the program and the user was task-oriented and restricted to the topics crucial for completing the system's goals.

The program was delivered to users via email in a compressed archive package. The package included the system's files – the program itself and the sound files it utilized –

and a text manual. In crucial moments such as first introducing itself or the beginning of the workday, the program offered to open the manual for the user to read through. The user could still open this file manually at any other point, since the system informed him or her of the location of the file.

At times, the system asked the user questions about their work and based on the answers gathered the data necessary for further operation. That data was stored in text files created automatically in the program's main directory on the user's hard drive.

Since the system was meant to serve as part of preliminary research and needed to be evaluated, the package distributed to the users contained an additional text file with the user survey. This survey can be found in Appendix 1.

The program was created in Python 2.7 and used the GUI window as user interface. It was meant for the Windows operating system.

3.2.3 Implementation

The proposed system needed to be run twice: once for the introductory conversation and creation of the work plan, and the second time for the work itself. This section gives overall descriptions of both interactions with the user.

When the user first launched the program, the system introduced itself. This action included asking for the user's name, which was then saved in an external file for later use, and offering to open the manual file. After that, the program asked for the work plan. Only one day of work could be input into the program at a time. All tasks on the plan needed to be input in a way specified in the manual (but not necessarily in a chronological order), including the name of the task, the time when the task should begin and how much time was allocated for it. After all tasks had been input, the program sorted them chronologically and checked against any overlaps. If there were none, the plan was accepted; if some tasks overlapped, the program asked the user to modify the plan. All subsequent modifications were checked again until the plan got accepted. The program then displayed the work plan in its window, showing the starting time and name of each task. This plan was stored in a separate text file.

Following the creation of the work plan, the user was asked to fill in information about rewards. A reward was access to a given website or to a file on the hard drive – for example an e-book – that the user would like to open in their spare time after completing a task. The user was asked to provide a simple name for the website or file and its URL address or file path, which was then stored in a separate text file. This was the end of the first conversation with the program; it then said goodbye to the user and turned off.

The second conversation could take place immediately after the first one or sometime later. The only condition was that the user started the second conversation before the actual work was supposed to begin. After turning on the program, *Asystent* greeted the user by their name and informed them of the current date and time. Then it displayed the work plan again and asked for confirmation. This was the last chance for the user to make changes to the plan if they wished so. The new plan was checked against overlaps in the same fashion as the original plan.

Next, the program asked whether the user would like to turn on the two sound reminder functions. The first function signalized an upcoming task. If the user wanted to use this function, he or she was asked to input how many minutes in advance he or she wanted to be reminded of the task. The second reminder function signalized the time allocated for a task coming to an end. Similarly, the user input the number of minutes when the sound should ring before the task ended. Both functions were adjusted slightly on some occasions where the break between tasks was shorter than the time for the first reminder or a task itself was shorter than the time for the second reminder. The reminder functions were implemented and made optional after oral consultations with potential users. The user could choose to use both, neither or just one of them. To avoid disturbing the user, both reminder functions only used sound, but the program did not display any visual cues.

After asking about the reminder functions, the program displayed the time when the first task was supposed to begin.

The following steps were then executed for all tasks on the work plan list:

- Reminding the user about the upcoming task a few minutes before the task (if the user opted to use this feature).
- At the time when the task was supposed to begin, signalizing the start of the task and informing the user about the amount of time allocated for the task.
- Reminding the user about the task coming to an end a few minutes in advance (if the user opted to use this feature).

- When the task was supposed to be finished, signalizing the task ending and asking whether it was completed.
    - If the task was completed, it was added into the file containing a list of completed tasks. The user could then opt to open a website or a file of their choice as a reward.
    - If the task was not completed, it was added into the file containing a list of failed tasks. There was no reward in that case.
- Notifying the user about the time the next task was supposed to start (this step was not executed if the previous task was the last one on the task list).

After the work was finished, the program summarized the day by displaying both the completed and the failed tasks lists. Both lists were stored as text files in the program's directory for the user to access at any time in the future. Next, the program asked for an opinion about the cooperation, reminded the user to fill out the evaluation survey, thanked the user for working with it and terminated after saying goodbye.

Figure 2 shows the general flowchart of both conversations between the user and the program. The text in italics paraphrases the system's utterances; it conveys the meaning of the actual utterances that were too long to include in the figure. Dotted arrows mark transitions between stages performed for every task on the work plan except for the last one. The file *name.txt* is created during the introductory conversation and contains the name of the user. Therefore, its existence can be used to determine which script the program needs to launch into: the introductory one or the second one for the actual work. The name itself was also used to determine the gender of the user, as Polish female names usually end with -*a* and male do not. This was important to properly gender the language, because Polish uses grammatical gender.

An example conversation between the user and the program translated from its original Polish version can be found in Appendix 2. The number of turns in conversations varied depending on the number of tasks input by the user, but generally stayed within the 45-60 range.

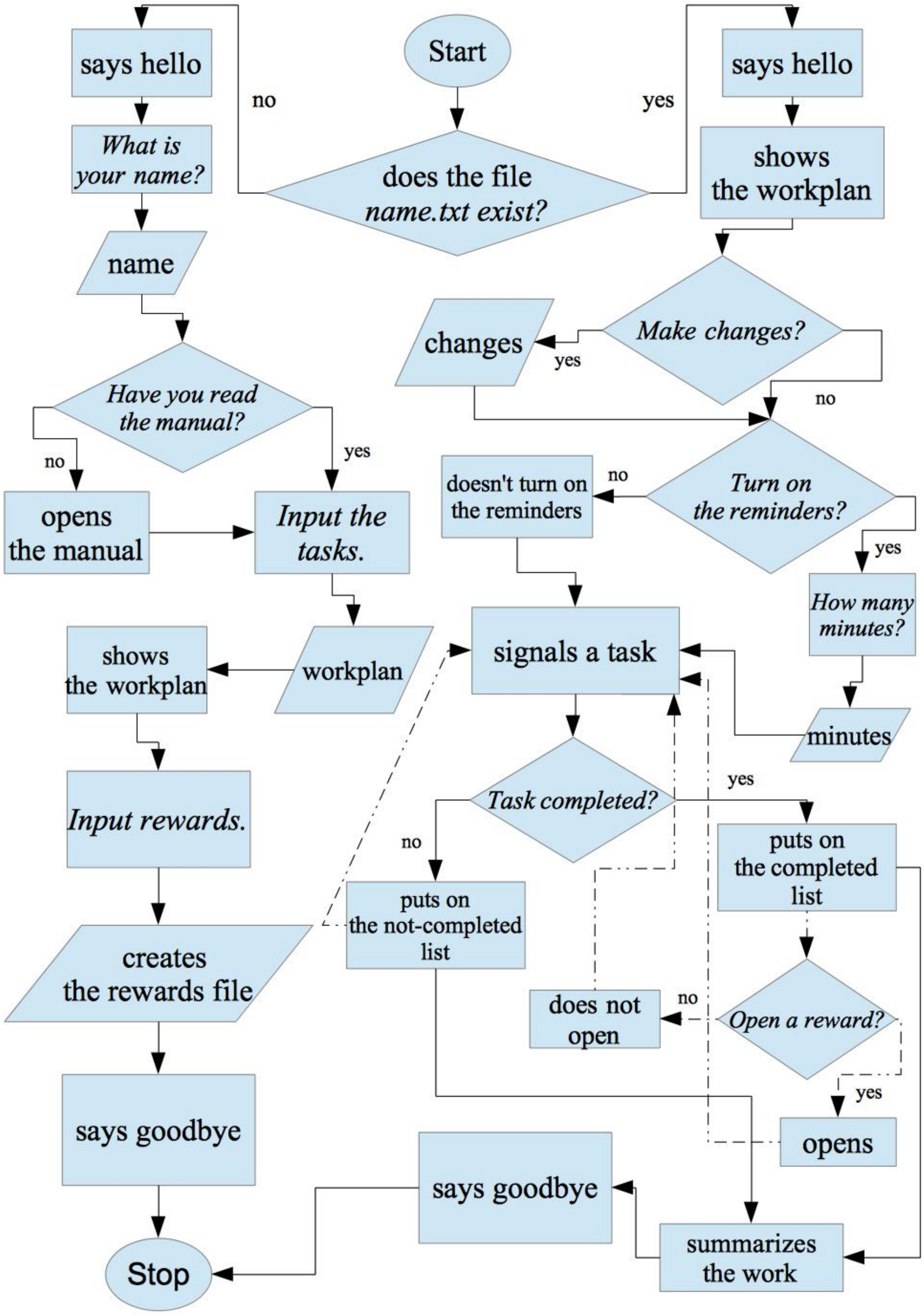

Figure 2: Flowchart illustrating cooperation with Asystent *during both conversations.*

3.2.4 Cooperation with the user

Asystent had various features meant to make conversations with him more friendly, motivational and natural. Some of these features were suggested by potential users during consultations that took place before the creation of the program. As the research was preliminary, the system implemented rather simple techniques. Their purpose was only to test whether particular behaviors would be accepted by users and whether they made the system friendly, motivating and helpful. If they did, the techniques used in this preliminary study could be developed further.

The proposed system saved all the user's tasks, indicated issues with the work plan (by checking against overlaps), stored information about both completed and failed tasks, reminded the user about upcoming tasks and summarized the work done throughout the day. This is reminiscent of the work of a human assistant or secretary. To avoid disturbing the user, the program worked in the background; whenever there was a message from the system, only the corresponding sound was played, but the GUI window was not maximized. The system also kept the conversation to the minimum. This in turn made it similar to an electronic organizer. All the above functions were meant to make *Asystent* as helpful as possible. Moreover, keeping track of the tasks and reminding the user about them were meant to motivate the user to work.

Asystent also implemented several different techniques to make the conversation natural. One of them was using colloquial language in the dialogue. This was done to create the feeling of familiarity with the user and create a friendly atmosphere. Another feature was introducing some variety to the conversation script. Even though all the dialogue was predetermined in the program code, most utterances had a few synonymous versions (usually three). At appropriate points in the conversation, the system chose one of them at random. All the options were roughly the same in terms of semantics and register, but shuffling through them let the program avoid repetition. Finally, the program's utterances came with two seconds' delay after the user's answers. This served to imitate an actual human interlocutor and was inspired by online chatting applications. In a usual conversation taking place online, the other person responds after a few moments, because they need to think about the answer and type it into the GUI. As a computer program, *Asystent* was technically capable of responding immediately, but the delays were incorporated into the code so that the conversation would feel more natural.

To minimize the number of errors occurring due to invalid input of the data and to give the user some freedom in their word choices, *Asystent* usually suggested possible answer options for its questions, but could recognize some other answers with similar meaning as well. For example, even if the program expected the answer *ready*, it could still recognize the answer *ok*, which might seem more natural to the user. All the options (usually around three) were included in the code. Furthermore, to recognize the answer, *Asystent* did not differentiate between upper- and lowercase letters, and only needed the first few characters of the word. In the example of *ready*, it could recognize the word after matching only the first three characters. This was implemented to avoid errors arising from spelling mistakes, which tend to occur with longer words.

3.3 Evaluation method and results

The proposed system was evaluated through a user survey that came with the system's package. The users were allowed a few days to test the program and then were asked to complete the survey. Whenever there were issues with the program's functioning, the users were allowed to contact me for help. However, the testing itself was not supervised; the users installed Python and launched the program on their own personal computers, and interacted with it without my assistance.

There were seven users, five female and two male, aged 21-30, who filled out the evaluation survey. Six of them had previous experience with Python, which may have helped them handle cooperation with *Asystent*. The participants were not paid for their efforts. While I recognize the need of a bigger-scale evaluation, I had difficulty gathering more participants due to small interest.

Asystent was evaluated on a 5-point scale, with 1 being the lowest and 5 being the highest score, on six criteria presented below. All the criteria were explained in detail in the survey to avoid confusion. The list below includes a rather faithful paraphrase of the wording used in the user survey for the explanations of each criterion. For the exact phrasing of the questions, please refer to Appendix 1.

- Naturalness of interaction (being able to imitate a human interlocutor).
- Helpfulness (having useful features that make the user's work easier and not disturbing the user).
- Flexibility of dialogue (the dialogue not giving the impression of being strictly pre-defined).

- Friendliness (being able to create a nice working atmosphere and assisting the user to the best of the program's abilities).
- Ability to motivate the user to work (user's personal impression that they completed more tasks compared to when working alone).
- Quality of the cooperation (the program being a good assistant and the cooperation going smoothly).

The survey also included three other questions to users:

- Have you had any issues when working with *Asystent*? What was the problem?
- Would you like to work with *Asystent* again and why/why not?
- Please write your additional comments and opinions.

Table 1 presents the results of the evaluation including the six criteria and the *Would you use the program again?* question. The three bottom rows give mean, median and mode (the most frequent value) of each numerical result. Names of the six criteria are shortened to *naturalness*, *helpfulness*, *flexibility*, *friendliness*, *motivation* and *cooperation* respectively. The question whether the users would like to try *Asystent* in the future is shortened to *Work again?*

Table 1: Results of Asystent*'s evaluation.*

No	**Age**	**Sex**	**Naturalness**	**Helpfulness**	**Flexibility**	**Friendliness**	**Motivation**	**Cooperation**	**Work again?**
1	27	F	5	5	2	5	5	5	yes
2	29	M	3	3	2	3	3	3	no
3	30	M	4	4	5	3	4	2	no
4	28	F	4	5	4	5	5	5	yes
5	28	F	4	1	3	3	1	1	no
6	21	F	4	5	4	5	4	5	yes
7	27	F	3	4	2	5	5	3	yes
		Mean	3.9	3.9	3.1	4.1	3.9	3.4	
		Median	4	4	3	5	4	3	
		Mode	4	5	2	5	5	5	

3.4 Error analysis and discussion

The results of the survey indicate that the system was rated moderately well. All the criteria received a mean score above 3 points, which was the middle of the scale, and four of six criteria had a mode value of 5, which is the highest score. Especially high ratings were given for helpfulness, friendliness, being motivating and the quality of the cooperation. Naturalness of interaction also scored high. Moreover, most users would like to work with *Asystent* again in the future. According to them, *Asystent* has an easy user interface, provides motivation and is useful. One user complimented the reward system of opening a website of choice, because it imitated this user's usual habit of only switching to relaxing activities after having completed a task.

However, not all users were satisfied with the program. Three users would not use *Asystent* again because the user interface was unattractive graphically and there occurred many errors during the work. The interface can be seen in Figure 3 below. Program's utterances are in blue and user's answers are in black.

Incidentally, the users who gave the lowest ratings were also the ones who encountered most errors. The errors were mainly caused by the incompatibility of the program with some versions of text editors – which *Asystent* used to store the data – and with Mac OS, as it was written for Windows (it used the *winsound* module to play the sounds, which was created specifically for Windows OS). Moreover, any error meant that the program needed to be relaunched, which was rather tiresome. As such, these users' dissatisfaction was mainly caused by technical issues rather than by the poor performance of the program as an assistant. User 5 in particular stood out as the most dissatisfied with the program, and the ratings given by this user do not correspond to ratings given by other users regarding the respective criteria. It is possible that this user's ratings were particularly low because she encountered especially many technical errors.

One user indicated that they found some instructions of the program unclear. Furthermore, on occasion the users' answers were unreadable to the program and generated errors or misunderstandings. For example, the program mistook a positive answer for a negative one or did not recognize the answer at all, so it remained idle until a proper answer was typed in. Yet another user indicated that, despite clear instructions not to do so, they turned off the program during work and left the house, hoping to be able to resume the cooperation after returning. This was not possible for this version of the program, so the user's satisfaction dropped significantly.

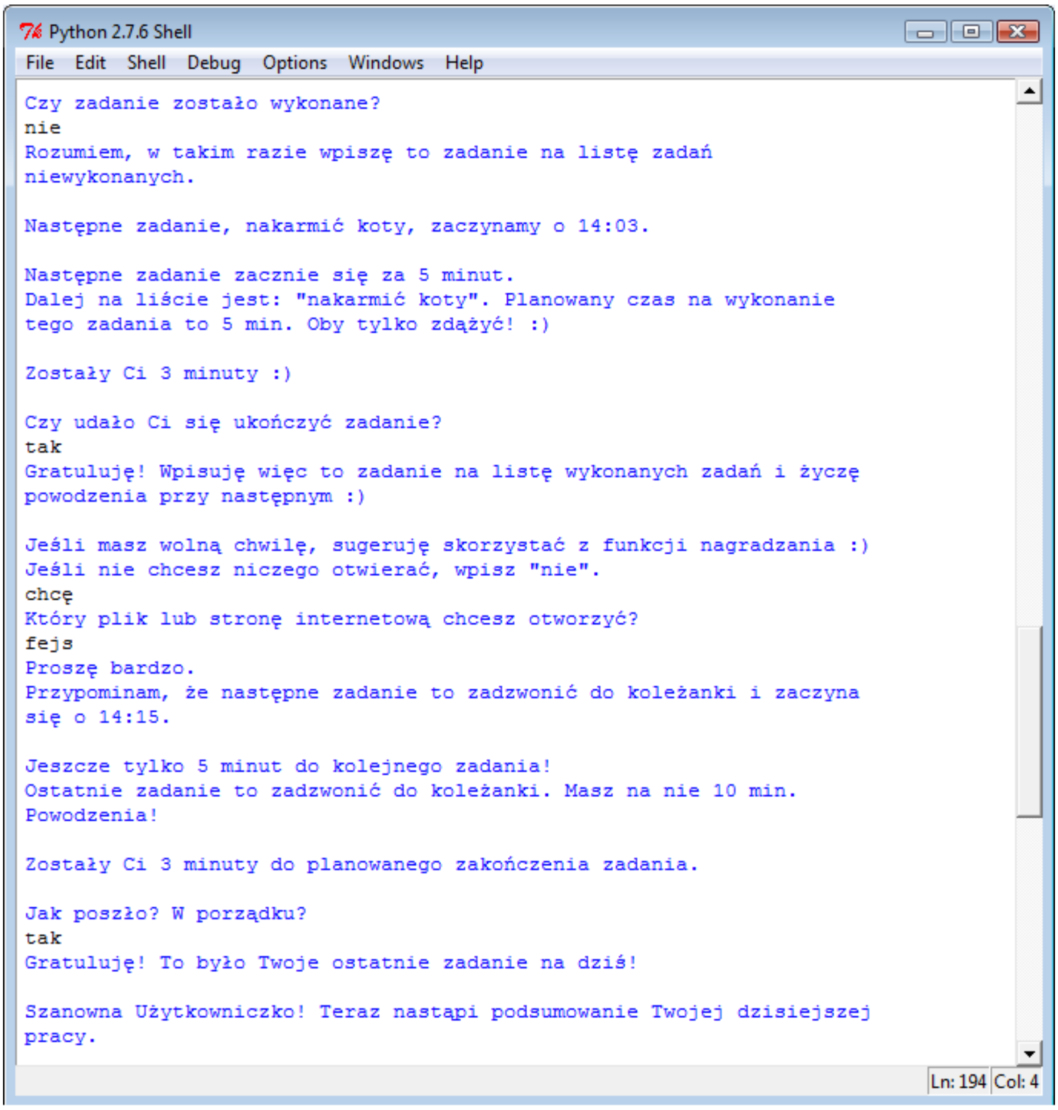

Figure 3: Asystent*'s interface in its original Polish language.*

The lowest scores were given for flexibility of the conversation. Since the system was rather simple, this was a necessary cost. Moreover, there were too many turns in the conversation and the user had little to say, because the initiative was almost entirely on the system. This indicates the need to improve user-program dialogue, which is one of the most important directions for further development of *Asystent*.

3.4.1 Potential future developments

The users' survey comments included suggestions about adapting the program for various operating systems and mobile devices such as smartphones. While the system described in this chapter required working with a computer, it seems reasonable to assume that most users would like a more portable program, perhaps in the form of a smartphone application. This need came especially clear through the comments of the user who terminated the program mid-work to pursue outdoor activities. Putting the program on a smartphone would enable users to report any progress with work on the spot without having to wait to get home and turn on the computer to connect to the program's database.

Other suggestions regarded the graphical form of the user interface and the possibility to plan more than one day ahead, which was not included in this version of the system. One of the users also advised to reduce the number of conversation turns and to include the feature of turning off the delays, because the dialogue became tiresome during consecutive launches of the program.

There are also other potentially useful features that were not suggested by users, but could be implemented in any further versions of the program to improve user experience. Those would be displaying the work plan on command at any point during the work, turning the reminder functions on and off whenever desirable, and providing a graphic representation of the user's progress. Moreover, the system could have a feature similar to Timeful's ability to plan the tasks itself instead of putting most of the planning work on the user.

Motivating the user to do the work could be approached in a few different ways. One of them could be to create a motivating module that adapts to the user's individual needs. Motivation is a complex psychological concept and as such different people are motivated in different ways. For example, some people may be motivated by the friendly approach *Asystent* had, but others may need a stricter assistant. This could be implemented as different registers for the system's utterances and changes in the reward system. Perhaps with the former user group the system could praise them for completing a task while the latter group would prefer a system that rebukes them a lot for being late with their work (these two are not mutually exclusive, but there would be different percentage of "praises" and "rebukes" in the system's utterances for both groups). Another potential discrepancy could arise between users who work slowly but steadily throughout a prolonged period of time and those who are motivated by time pressure and only complete all the work right before the deadline. To differentiate between these two user groups, the system could plan their tasks on the timeline accordingly. Finally, some users might prefer an assistant who quietly records their progress, while others might be more motivated by a program that reminds them about the tasks multiple times a day. For the latter group, the system's reminder functions could be much more active than for the former group. This list is by no means complete, but generally illustrates the technical steps that could be taken to make *Asystent* more motivating.

3.5 Conclusions from the study

This experiment, while simple and small-scale, helped to identify and understand the users' needs for a motivating electronic assistant. Most of all, it showed that simply being friendly and helpful was not always enough to motivate users to do the work. Although the ability to do this was one of the highest rated aspects of the current version of the system, this feature can and must still be improved, especially if the system is supposed to work for more than one day. Moreover, for the program to be able to imitate a real human assistant, it is crucial to increase flexibility of the conversation.

The system had a lot of technical flaws. However, a lot of the issues indicated with *Asystent*, such as incompatibility with some platforms or an unattractive user interface, have already been successfully solved in other calendar applications or electronic assistants. This means that focusing on mending those issues would be repetitive. On the other hand, *Asystent* had something that most other systems lacked. This was the specific purpose of motivating the user to do the work by conversing with them in natural language. I decided to capitalize on this feature. The findings from this study inspired me to create a motivating dialogue system that could then be potentially used as an addition onto an electronic calendar. Furthermore, research on relational agents and the fact that so many people search for motivational advice online inspired me to create the system so that it could be a supportive companion to the human user.

The ultimate goal of this research is to create a dialogue system that would be able to hold a motivational conversation with the user. For example, if the user has issues with completing a particular task, the system could detect that and ask for explanation. The user would say why they are unmotivated to do the work and the system would give them advice on how to overcome this obstacle. Essentially, this is the type of conversation that was illustrated in Figure 1 in Section 1.2 of this thesis.

There are two things essential for the creation of such a system. The first one is a deep understanding about what makes an utterance motivational. In other words, what features the motivational speeches or texts possess and how they can be distinguished from non-motivational utterances. Perhaps there could even be a way to detect good quality advice. The second thing is to create a large corpus of motivational utterances for the system. This corpus could be used either as training data (if the system implemented machine learning as an underlying mechanism) or as a source to retrieve advice from (if the system opted for retrieval instead of generation). As of now, no such corpus exists. However, it could be automatically created after solving the first two questions about distinguishing motivational advice texts from regular ones and then choosing only the best advice texts to put into the corpus. The next three chapters detail my research on these two topics.

Chapter 4: Datasets for advice classification and ranking

The dialogue system *Asystent* was created for the Polish language. However, I decided to move on to English so that my studies could be helpful to a variety of researchers around the globe and so that there would be more data available for experiments.

As the source of motivational and advisory texts, I chose the online discussion platform Reddit, which allows its data to be downloaded freely for research purposes. Reddit is a place where people discuss various topics, share opinions and ask questions to other users. It is divided thematically into so-called subreddits, where users can exchange opinions and information on specified topics. A thread is created by one user posting to any appropriate subreddit and other users commenting on that post. Users can also vote on both posts and comments, adding to and subtracting from the overall score of each post and comment. Reddit has already been used as a data source in a variety of NLP tasks, such as topic mining (Park, Conway and Chen, 2018) or sentiment analysis for suicide prevention (Allen, Bagroy, Davis and Krishnamurti, 2019).

Some subreddits are explicitly dedicated to seeking advice in general or motivational advice in particular. In my experiments, I have studied data downloaded specifically from those subreddits, because the comments were bound to contain advisory or motivational content. Moreover, user posts in those subreddits are closest to what I imagine as input to the motivational module while other users' comments are closest to the expected output.

Although the main focus of this research is motivational advice, the shortage of data necessitated the use of texts containing other types of advice as well. Some subreddits are not very active, which means there is not enough data to work with. Moreover, Reddit's API (Reddit, 2019)[15] limits the number of downloadable posts to under a thousand most recent ones at a time. While the entirety of Reddit can be downloaded as a monthly-based dump, this would be cumbersome and take a disproportionate amount of resources. Instead, I opted for downloading the contents of each subreddit on a weekly basis and then remove duplicate threads by leaving only the most recent version. The idea was to obtain the most recent comments and their scores.

In general, a Reddit submission always requires a title for the post, but some subreddits allow posts with no text in their body. As a rule, for the motivational and

[15] https://www.reddit.com/dev/api/

advisory subreddits I only downloaded those threads that had text in the body of the post and disposed of those that only had a title. This was done because in my initial studies I worked with both posts and comments. Even though later my main algorithms focused on comments only, I kept the tradition of downloading posts along with the comments for any future reference.

Comments on Reddit can be infinitely nested as responses to other comments. In my studies, I only downloaded the comments from the highest layer. Therefore, any time that a number of comments is indicated in this thesis, it means the number of comments of the highest order without their nested responses.

Figure 4 shows the layout of a typical Reddit post.

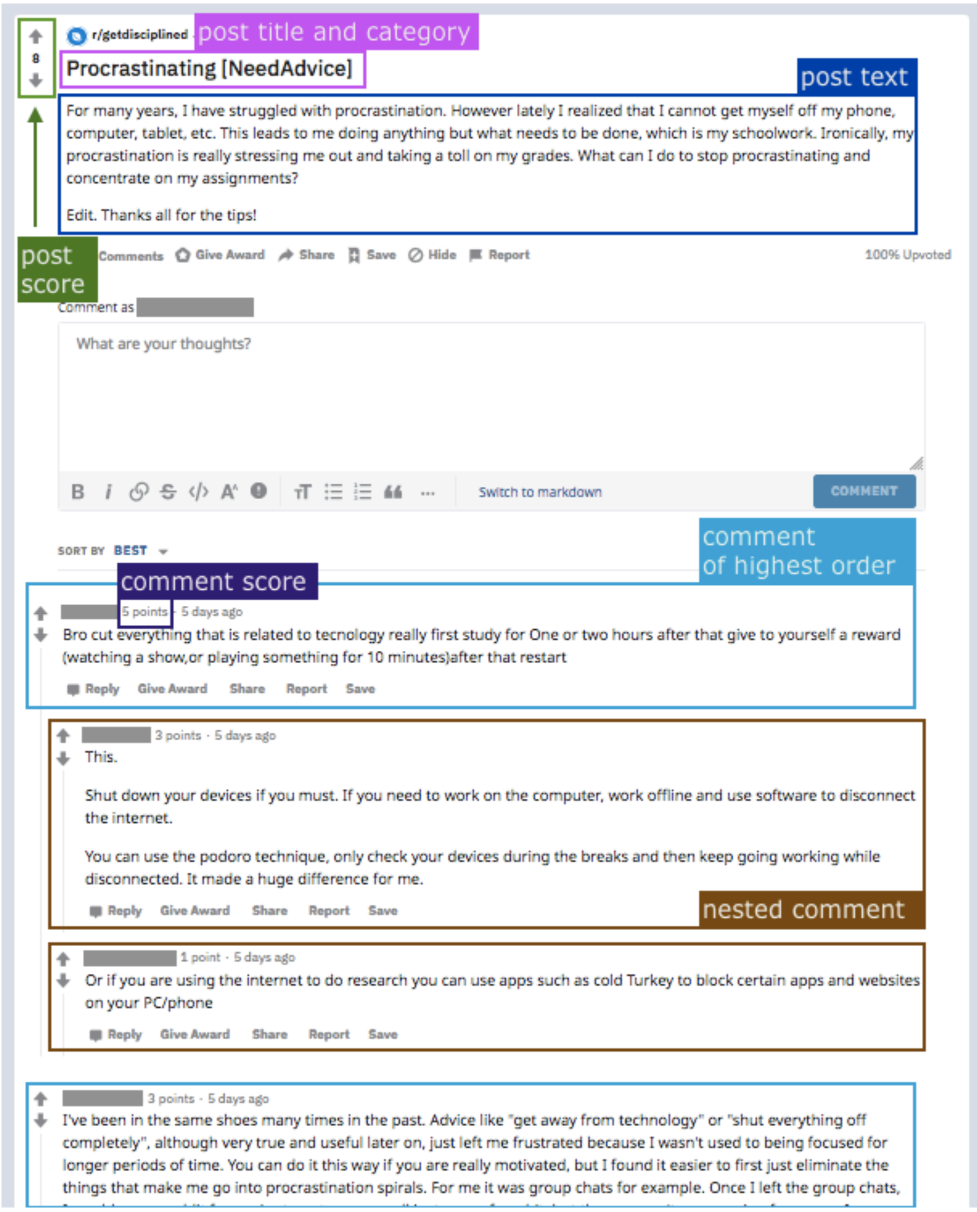

Figure 4: A Reddit post from the r/getdisciplined *subreddit. Usernames have been greyed out.*

The specific ways the data has been downloaded and utilized will be detailed in the next two chapters. This chapter serves to introduce the datasets and remark on their qualities. The sections below give details on each subset of data obtained for the experiments from Reddit. At the beginning of each section, there is an official description of the subreddit that was taken directly from its page. All links to subreddits can be found in Appendix 3.

Finally, Reddit is an online discussion platform and as such is prone to spelling and grammar errors. These have been corrected in data preprocessing in ways detailed in the next two chapters. However, all comments and posts presented in tables are in their original versions.

4.1 Advisory subreddits

4.1.1 The *r/getdisciplined* subreddit

A subreddit for people who have issues with procrastination, motivation, and discipline. It is a great place to gather and meet others with a similar mindset. Meet your goals and improve your life, reddit style!

The subreddit *r/getdisciplined* is a place for unmotivated people to seek motivational advice from others. The posts cover a broad range of topics, from being unmotivated at work or university to procrastination stemming from mental health issues. People who comment are required to help others attain self-discipline by sharing what helps them personally or otherwise offering advice.

Posts in this subreddit are divided into a few categories, such as *Discussion*, *Method* or *NeedAdvice*. I only used the last category for my dataset. Posts from other categories contain discussions, plans and general suggestions, but only *NeedAdvice* guarantees that the original post asks for advice and that the comments give motivational advice regarding that particular problem. As stated before, these particular threads closely resemble the type of input and output that I envision for my dialogue system.

Even though the subreddit is rapidly growing and gathers over 495,000 members as of November 2019, there are too few posts in category *NeedAdvice* for the dataset to be sufficient for deep learning experiments. Therefore, I have downloaded posts from other advisory subreddits as well.

4.1.2 The *r/Advice* subreddit

This is a place where you can ask for advice on any subject. Everybody has issues that they run into, and everyone needs advice every now and again. This is Reddit's very own solution-hub.

As stated in the description above, *r/Advice*[16] answers questions on a variety of different topics, from general life advice to specific problems with studying or family relationships. This subreddit is very active and therefore a good supplementary source of data for my experiments.

4.1.3 The r/relationship_advice subreddit

Need help with your relationship? Whether it's romance, friendship, family, co-workers, or basic human interaction: we're here to help!

At over 2.2 million members as of November 2019, the relationship advice subreddit[17] can provide a lot of advisory data. This data is, however, limited to the topic of relationships of various kinds, such as familial, spousal or workplace relations.

4.1.4 The *r/legaladvice* subreddit

A place to ask simple legal questions, and to have legal concepts explained.

The last advisory subreddit used in my studies was *Legal Advice*[18], a place for people to ask advice on legal questions. These questions vary from harassment lawsuits, custody battles and legal name changes to simpler matters like dealing with burdensome neighbors or unruly customers.

[16] https://www.reddit.com/r/Advice/

[17] https://www.reddit.com/r/relationship_advice/

[18] https://www.reddit.com/r/legaladvice/

4.2 Non-advisory subreddits

4.2.1 The *r/todayilearned* subreddit

You learn something new every day; what did you learn today? Submit interesting and specific facts about something that you just found out here.

As negative examples for the neural network models described in the next two chapters, I have used two subreddits where comments do not contain advice. One of them is *Today I Learned*[19], where people share interesting facts and have discussions about them.

4.2.2 The *r/pics* subreddit

A place for pictures and photographs.

The other non-advisory subreddit was Pics[20]. This is a page that only accepts posts in the form of pictures; no text can be typed in the body of the post. Commenters discuss the content of the pictures, but there is no advisory content; as this is not a photography subreddit, people do not comment on the quality of pictures or advise the photographer on their technique.

In either *r/todayilearned* or *r/pics* it is technically not forbidden to provide advice, but a thorough look through the comments on these subreddits reveals that in general, advice is not present. This is because the nature of the subreddits, as well as the content, does not invite advice. Moreover, it would be useless to try to advise the author of the post anyway, since in most cases they are not the creator of the picture or the interesting fact; they merely relay the information. Therefore, both these subreddits could be successfully used as negative examples in my experiments.

[19] https://www.reddit.com/r/todayilearned/

[20] https://www.reddit.com/r/pics/

Chapter 5: Advice classification task

It was not immediately obvious how the task of advice classification should be approached. Therefore, I started with some preliminary studies to better understand the data and test out a few tools for language processing. However, these studies did not bring any significant results. I proceeded to devise a different method for classification by using a neural network.

This chapter gives details about each step in my research. Section 5.1 presents the preliminary studies on the qualities of motivational/advice texts. Section 5.2 describes features used for the neural network classifier and the way they were chosen. Section 5.3 deals with my initial experiment, including a thorough error analysis that served as a basis for future studies. Section 5.4 details my experiments using deep neural networks. Section 5.5 presents experiments that used word2vec word embeddings in addition to my 14 advice features. Section 5.6 shows my proposed method of classification. Section 5.7 concludes the chapter.

5.1 Preliminary studies

In my preliminary studies, I analyzed 293 threads from *r/getdisciplined* in category *Need Advice* that had at least one comment and post score above 0. This was done as a measure against bad quality data; posts with a score below 0 often devolve into irrelevant discussions. Overall, I obtained 1,925 comments.

I analyzed the comments using popular NLP techniques, such as sentiment analysis, polarity analysis and calculating cosine similarity between the post and its comments with word2vec (Mikolov, Sutskever, Chen, Corrado and Dean, 2013) and doc2vec (Le and Mikolov, 2014). I then compared results from all these tools for each comment with the comment's score as it was given by users of Reddit to find any potential correlations. I hoped to find a textual quality – for example, a particular sentiment score or cosine similarity level – that would correlate with comment score and therefore serve as an indicator of a comment's motivational power or determine whether the comment contained advice in general.

Sentiment was analyzed with Vader, which is a part of the NLTK library (Bird, Klein and Loper, 2019)[21]. It assigned four real number values to each comment text. The values

[21] https://www.nltk.org/

represented *positive*, *negative*, *neutral* and *compound* sentiment, respectively. *Positive*, *negative* and *neutral* sentiment values fell between 0 and 1, and *compound* value, which denotes sentiment intensity, fell between -1 and 1.

Polarity was analyzed with TextBlob (Loria, 2018)[22], which gave real number values between -1 and 1 for each text.

Word2vec and doc2vec models were trained on articles from Wikipedia (latest version downloaded in February 2018 from (Wikimedia Foundation, 2019)[23]). However, both models were trained on different subsets of the data, which was caused by the differences between the algorithms themselves at the time of the study. Word2vec had an option of incorporating new texts into its vocabulary during training, so the data could be supplied sequentially. Therefore, I used the entire dump for this model. Doc2vec did not have this option at the time, so all training data had to be supplied at once. For this model, I used the first 1,320,909 articles of Wikipedia dump, which was the maximum that would not give a memory error (the machine that was used for these calculations had 128GB of working memory). Table 2 shows hyperparameter values for both models. All hyperparameters not specified in the table were left at their default values.

Table 2: Hyperparameter values for word2vec and doc2vec models.

Hyperparameter	Word2vec	Doc2vec
algorithm	skip-gram	distributed memory (PV-DM)
vector size	100	100
minimum word frequency	5	5
initial learning rate	0.025	0.025
maximum distance between current and predicted words	5	8
epochs	10	10

With word2vec, I trained two models: one on words and one on single characters. I used the character model to deal with unknown words, which is generally a problem in word embeddings models. In this study, a vector for the unknown word was calculated as an average of all character vectors of this word.

[22] https://textblob.readthedocs.io/en/dev/index.html

[23] https://dumps.wikimedia.org/backup-index.html

Doc2vec provides a vector for the entire text. Word2vec provides only vectors for individual words. Therefore, for the word2vec method, I first calculated vectors for all sentences in the text using the method described in (Arora, Liang and Ma, 2017). Then I calculated the text vector by averaging all sentence vectors.

The calculations were performed for each comment in the dataset and each corresponding post text.

There was a high average cosine similarity between posts and their comments, which was 0.906 as measured with word2vec. This was to be expected, as the comments were supposed to answer the call for advice from the post, so they stayed on the same topic. Unfortunately, neither this nor any other feature correlated significantly with the plain comment score, or comment score normalized against its post score. All Pearson coefficients calculated for this study were lower than absolute value of 0.1. There were also no strong correlations between sentiment scores or polarity scores within post-comment pairs (Pearson coefficient values around 0.1 for all pairs of sentiment scores and 0.033 coefficient for polarity pairs).

As a result, I concluded that these rather simple features could not be helpful in determining the motivational power of a text, the quality of advice expressed in that text (as measured by comment scores) or whether a text contains motivational advice or not in general. To find out which features could be helpful, I performed a short qualitative analysis on the best ranked comments from *r/getdisciplined* and established a list of semantic features separating them from other comments. These features are described in the next section.

5.2 Advice features

The aforementioned analysis revealed that the best ranked comments from *r/getdisciplined* usually had two things in common: they provided very specific, practical advice for the troubled user who made the post, and included mentions of being able to relate to the struggles of that user, usually because the commenter had to deal with the same problem in the past. I operationalized these characteristics into 13 textual features.

Before calculating the feature values, I pre-processed each comment by detecting sentence boundaries, assigning part-of-speech tags, and, for some feature calculations, removing stopwords (the following subsections will give details on whether stopwords had been removed for each particular feature). From now on, I will use names *wordlist_withstops* for a list of all words in the comment, *wordlist_nostops* for the same list with stopwords removed, and *sent_list* for a list of sentences in the comment. I also removed all non-ASCII characters, since the English language – which is the language of Reddit – does not need other coding. This was done to remove noise such as emoticons or unusual ways of embedding text into the comments.

The features are described one by one in the rest of this Section. Mathematical formulas are provided for features that were not calculated automatically with Python libraries.

5.2.1 Sentics Scores

Sentics Scores of **aptitude**, **attention**, **pleasantness** and **sensitivity** were measured using the Sentic library for Python on *wordlist_nostops*; this was done so that only content words were included in sentiment calculations, as they carry all the important meaning. The library is an API to the SenticNet knowledge base (Cambria, Poria, Hazarika and Kwok, 2018), which contains information on sentiment values of words. Each Sentic is associated with a dichotomy between different emotions: *admiration-loathing* for aptitude, *vigilance-amazement* for attention, *ecstasy-grief* for pleasantness and *rage-terror* for sensitivity. This is illustrated in Figures 5 and 6 (Cambria, Livingstone and Hussain, 2012).

All Sentics Scores fell between -1 and 1. Figure 7 (Cambria, Livingstone and Hussain, 2012) shows the scoring across the pleasantness axis with respect to emotions denoted on the emotional hourglass from Figure 5.

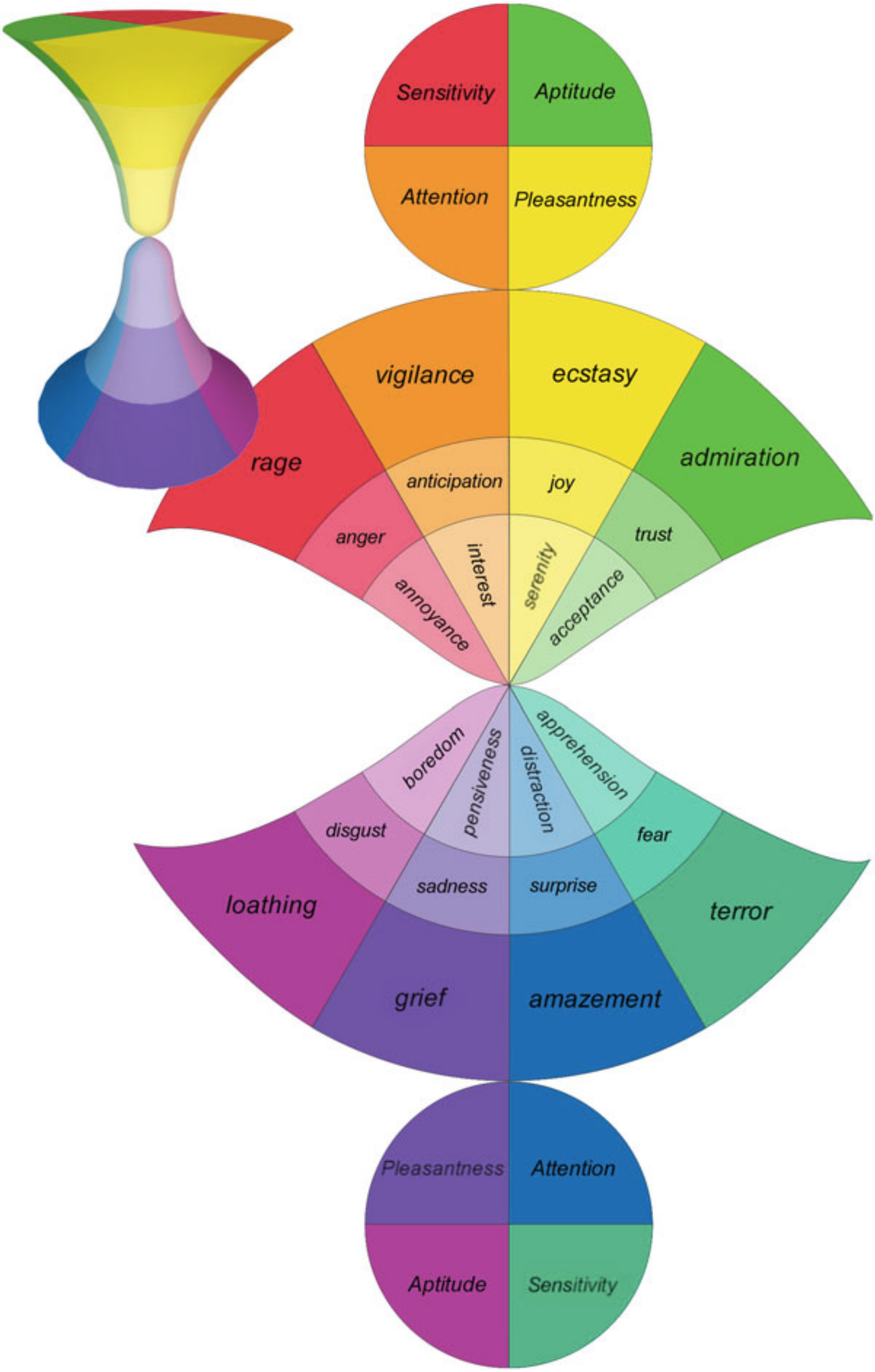

Figure 5: The emotion hourglass for Sentics (Cambria, Livingstone and Hussain, 2012).

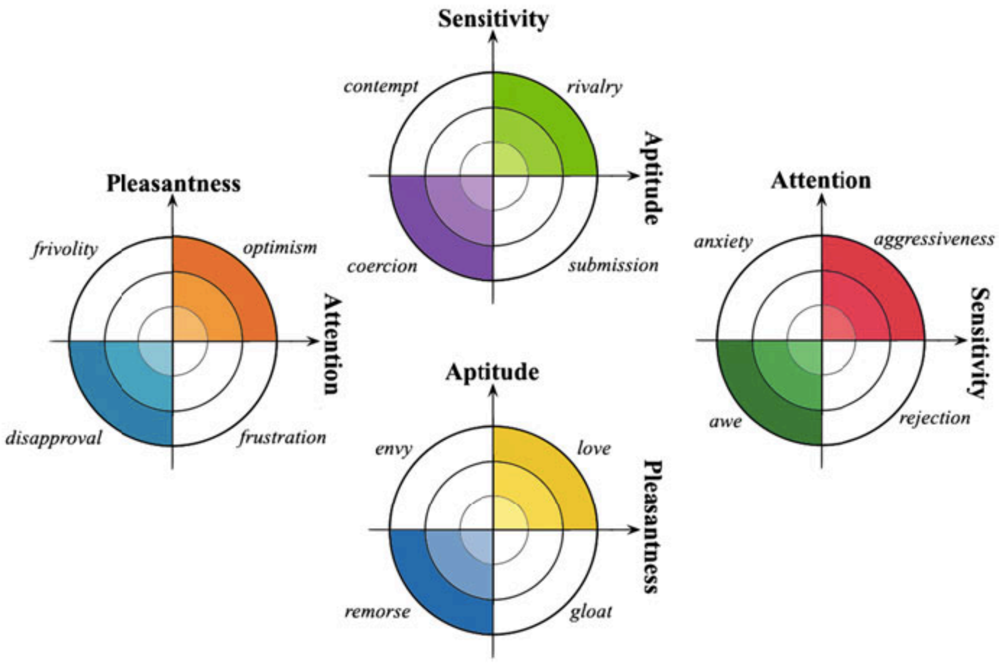

Figure 6: Secondary emotions that can be measured by combining Sentics scores (Cambria, Livingstone and Hussain, 2012).

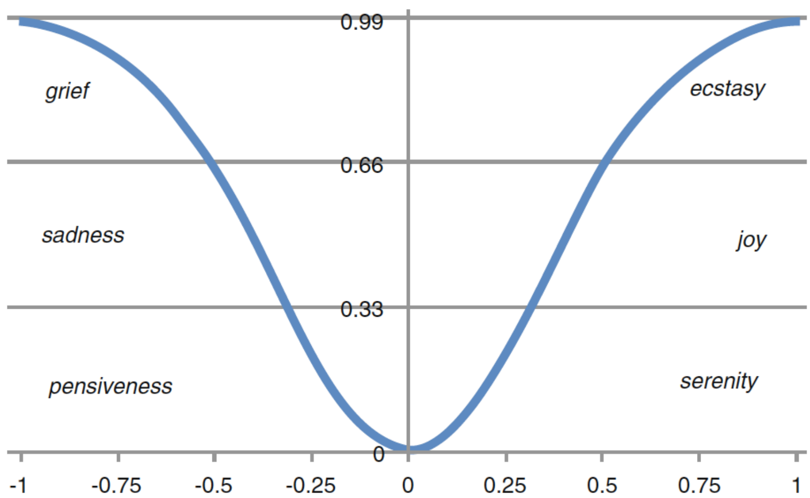

Figure 7: Scores across the pleasantness axis with respect to various emotions (Cambria, Livingstone and Hussain, 2012).

5.2.2 Sentiment Score

Sentiment score was also provided by the Sentic library and was measured on *wordlist_nostops*. The results fall on the 5-point scale of *strong negative / weak negative / neutral / weak positive / strong positive*. I converted them accordingly to integer values between -2 and 2, where -2 represented *strong negative* and 2 represented *strong positive*.

5.2.3 Relatability Score

$$Relatability\ Score = \frac{count\ of\ first\ person\ pronouns}{len(wordlist_{withstops})} \quad (1)$$

This feature was measured by the percentage of first person pronouns (including possessive pronouns) in *wordlist_withstops*. The score range was 0 to 1. Relatability Score was meant to capitalize on the fact that the best ranked comments used numerous first person expressions.

5.2.4 Imperative Score

$$Imperative\ Score = \frac{count\ of\ imperative\ expressions}{\frac{len(wordlist_{withstops})}{2}} \quad (2)$$

This feature was measured by the percentage of imperative expressions in the comment text. Specifically, I looked for clauses beginning with non-infinitive verbs, the word *please* preceding a verb, the phrase *why don't you* and phrases comprised of *you* or *OP* (*Original Poster*, which is a popular way of referring to the author of the post on Reddit) and a modal verb. Since most of these are bigrams, I counted the percentage on the number of all words divided by 2. The score range was -1 to 1; negative values come from deducting points for question marks. This was done to correct the score if the algorithm found a clause beginning with a verb and counted it as imperative even though it was a question. Deducting those points prevented skewing the overall score for the comment.

5.2.5 Specificity Scores

Specificity Scores included six separate features. Specificity Score was first proposed by Deshpande, Palshikar and Athiappan (2010) to help extract suggestions and complaints from employee surveys and online product reviews. According to my analysis of best ranked *r/getdisciplined* comments, best advice was usually the most specific. Therefore, I adapted the studies on text specificity for my experiments.

I calculated the Scores as described in Deshpande et al. (2010) with slight modifications, which are detailed below. The calculations were performed for each sentence in the comment using *sent_list*, and the final Scores for the entire comment were obtained by adding up all the sentence Scores. The features were: **Average Semantic Depth** (ASD), **Average Semantic Height** (ASH), **Total Occurrence Count** (TOC), **Count of Named Entities** (CNE), **Count of Proper Nouns** (CPN) and **Sentence Length** (LEN).

Semantic Depth of a word was the length of the path in the semantic tree (illustrated in Figure 8) that led from the word to the top of the hierarchy, which is the hypernymy-hyponymy hierarchy of the WordNet ontology (Princeton University, 2010)[24]. Semantic Height, on the other hand, was the length of the path from the word to its lowest hyponym. For example, in Figure 8 the Semantic Depth of the word *seafood* is 5, because it takes five steps to reach the top of the hierarchy (the *entity* element). Likewise, the Semantic Height of the word *substance* is 4, because it takes four steps to reach the lowest hyponym under that word. The idea behind these features is that a specific word would have higher score for ASD and lower score for ASH than a general word.

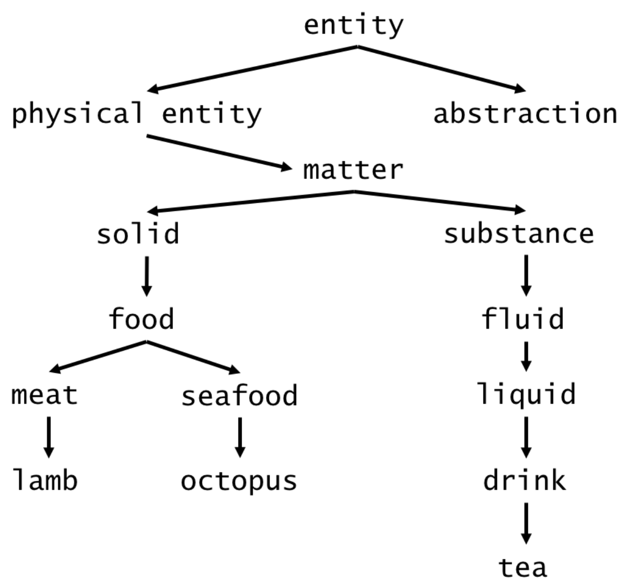

Figure 8: The semantic tree retrieved from WordNet adapted from (Zhu and Iglesias, 2017).

To obtain the Average Semantic Depth and Average Semantic Height scores for each sentence, all the ASDs and ASHs, respectively, were added for all the content words (meaning nouns, verbs, adjectives and adverbs) in that sentence and the sum was divided

[24] https://wordnet.princeton.edu

by the total number of those content words. Each sentence was a sentence from *sent_list* with stopwords removed.

$$ASD = \frac{\sum(longest\ path\ in\ WordNet\ to\ highest\ hypernym)}{count\ of\ content\ words} \tag{3}$$

$$ASH = \frac{\sum(longest\ path\ in\ WordNet\ to\ lowest\ hyponym)}{count\ of\ content\ words} \tag{4}$$

The next feature was Total Occurrence Count. For each word, WordNet stores the number of times that word occurs in the ontology. This number can be easily obtained and serve to calculate TOC. For each sentence in a comment, I obtained occurrence counts from WordNet for each lemmatized content word and added up three lowest scores in a sentence. This feature was meant to detect words that are so specific that they even rarely occur in the ontology.

$$TOC = \sum(bottom\ 3\ occurence\ counts\ in\ sentence) \tag{5}$$

Count of Named Entities meant the number of named entities in each sentence including stopwords. This was determined with the NLTK Named Entities tagger.

$$CNE = count\ of\ named\ entities\ in\ sentence \tag{6}$$

Count of Proper Nouns was measured by the number of proper nouns in the sentence with stopwords removed. Specifically, the feature looked for part-of-speech tags *NNP*, *NNPS* and *CD*. The first two mark singular and plural proper nouns, respectively. The *CD* tag marks cardinal numbers. This is not a proper noun, but was included in CPN calculations by Deshpande et al. (2010), because sentences containing numbers tend to be more specific. I decided to keep my calculations the same.

$$CPN = count\ of\ (proper\ nouns + cardinal\ numbers)\ in\ sentence \tag{7}$$

The last Specificity Score, Sentence Length, was the number of words in the sentence with stopwords removed.

$$LEN = len(sentence) \tag{8}$$

ASD, ASH, TOC and LEN were divided by 100 and CNE and CPN were divided by 10 to put the Scores in the same numerical range as other features. Before division, ASD, ASH and TOC had rather high values that went into dozens. This is because these Scores were calculated on a sentence-to-sentence basis and then added together to obtain the Score for the entire text. For example, if there were three sentences in the text with ASD of 4, 5 and 3.5 respectively, the text Score was 4+5+3.5=12.5. I divided these Scores to put them closer to the 0-1 range. The same goes for LEN, which was the length of the entire text and also tended to have high values before division. On the other hand, CNE and CPN tended to be much lower overall, so it was sufficient to divide them by 10 to scale them down. As long as the features are scaled uniformly across data classes, the difference in scaling individual features has no bearing on the results.

The main modification introduced in this research compared to Deshpande et al. (2010) was that I omitted converting every content word into a noun. The authors of the paper performed this step because at the time of their research the semantic trees in WordNet were relatively complex for nouns but very sparse for verbs and adjectives. Currently there is no need for such actions, as WordNet has grown significantly since 2010. Moreover, I did not combine the Scores into one value, but used them all separately in my algorithm.

Table 3 below provides example scores for a comment.

Table 3: A comment containing motivational advice retrieved from r/getdisciplined *with its feature values.*

Comment						
If your body clock is out of sync and you are having trouble falling asleep, try the 4-7-8 breathing technique. Simply exhale as much as comfortably possible, then take a deep breath to a count of 4, hold it for 7 seconds and then exhale for a count of 8 seconds. Repeat this 3-4 times and you will find a natural calming sensation which will also help to mimic your body's natural breathing pattern when you are sleeping.						
Aptitude	**Attention**	**Pleasantness**	**Sensitivity**	**Sentiment**	**Relatability Score**	
0.096	0.294	0.188	0.021	1.000	0.000	
Imperative Score	**ASD**	**ASH**	**TOC**	**CNE**	**CPN**	**LEN**
0.023	0.630	0.537	6.010	0.200	0.900	0.460

5.3 Initial experiments

5.3.1 Datasets

For the classification task, I utilized comments from a few different subreddits. For the motivational and advice comments, I used *r/getdisciplined* and *r/relationship_advice*. For the regular comments, I used *r/todayilearned* and *r/pics*. My goal was to train a classifier to differentiate between these two classes of comment texts.

Table 4 is an overview of the number of comments in all four datasets. I divided each one into training and test sets by roughly 70:30% of the entire dataset, respectively. These training and test sets were then combined appropriately for experiments.

Table 4: Overview of motivational/advisory and regular data used in the initial experiments.

Class	Subreddit	Number of comments	Training set	Test set
Motivational/advisory	*r/getdisciplined*	1,925	1,352	573
	r/relationship_advice	3,492	2,470	1,022
Regular	*r/pics*	3,399	2,255	1,144
	r/todayilearned	4,830	3,395	1,435

5.3.2 Technical details of initial experiments

Three classification tasks were performed. First, I used the *r/getdisciplined* dataset against the regular dataset of *r/pics*, and then against the regular dataset of *r/todayilearned*. Then, to increase the amount of training data, I combined the *r/getdisciplined* and *r/relationship_advice* datasets into one *all-motivational* class, and *r/pics* and *r/todayilearned* into another *all-non-motivational* class, and performed another experiment.

I used two classifiers: a Support Vector Machine (Ben-Hur, Horn, Siegelmann and Vapnik, 2001) and a custom-made fully connected shallow neural network with two layers. I chose the SVM because it is robust thanks to its large margin optimization technique and performs well in classification problems. The shallow neural network was then implemented to see whether it could improve on the SVM results.

For the SVM computations I used an RBF kernel with the parameter C=20. The hyperparameter values of the neural network are given in Table 5 below. Its weights were initialized randomly. All the hyperparameters for both classifiers were chosen experimentally based on performance.

Table 5: Hyperparameter values for the shallow neural network.

Hyperparameter	Value
input layer units	13
hidden layer units	10
hidden layer activation function	tanh
output layer units	1
output layer activation function	sigmoid
learning rate	0.2
epochs	20,000

5.3.3 Results

The SVM achieved accuracy of 0.860 in the *r/getdisciplined* vs. *r/pics* and 0.856 in the *r/getdisciplined* vs. *r/todayilearned* experiments. The scores rose to 0.864 and 0.875 respectively with the neural network. On the *all-motivational* vs. *all-non-motivational* classification task, the performance of both SVM and the neural network was slightly worse, decreasing to accuracy of 0.822 for the SVM and 0.843 for the network. Table 6 summarizes the results.

Table 6: Results of initial experiments.

Dataset	Training examples	Test examples	SVM	Shallow NN
r/getdisciplined vs. *r/pics*	3,607	1,717	0.860	0.864
r/getdisciplined vs. *r/todayilearned*	4,747	2,008	0.856	**0.875**
All-motivational vs. non-motivational	9,472	4,174	0.822	0.843

Since the results were best for the *r/getdisciplined* vs. *r/todayilearned* datasets with the shallow neural network, I focused on this particular classification task for further analysis. Table 7 shows precision, recall and F-score for this task.

Table 7: Results for the r/getdisciplined *versus* r/todayilearned *classification task.*

Dataset	Accuracy	Precision	Recall	F-score
Training set	0.872	0.829	0.692	0.755
Test set	0.875	0.844	0.691	0.760

5.3.4 Error analysis and discussion

For the *r/getdisciplined* vs. *r/todayilearned* dataset, the neural network had accuracy of 0.875 on the test set. Accuracy for the training set was 0.872, which means that the algorithm generalized well. However, assuming that the Bayes error for this task is around 0.0, these results suggest a high avoidable bias problem, with error rate at 0.128. This problem may be solved by standard techniques used to combat avoidable bias, including adding more features to the algorithm, using a bigger neural network and training for more epochs.

Overall, in the *r/getdisciplined* vs. *r/todayilearned* experiment, there were 250 misclassified comments out of the total of 2,008 comments in the test set. I performed a manual error analysis on all of them. I also calculated mean and median values for all of the 13 features across the 2,008 test set comments. The exception was the Sentiment feature, which is an interval variable and as such should not have its mean calculated. Instead, I calculated median and mode for this feature. The results are presented in Table 8. I divided the scores between *r/getdisciplined* and *r/todayilearned* groups, which are further broken into *correctly classified* and *misclassified* subgroups, giving 4 groups in total. This is to show differences in feature scores between the *motivational/advisory* and *regular* dataset, and the way they may have influenced the classification algorithm.

Table 8: Feature scores for the r/getdisciplined *versus* r/todayilearned *classification task. The differences between correctly classified comments from both categories that especially stand out are bolded.*

Feature		Correct *r/getdisciplined*	Error *r/getdisciplined*	Correct *r/todayilearned*	Error *r/todayilearned*
Aptitude	Mean	**0.153**	0.108	**0.092**	0.117
	Median	**0.148**	0.092	**0.053**	0.106
Attention	Mean	0.074	0.057	0.072	0.061
	Median	0.071	0.045	0.036	0.064
Pleasantness	Mean	**0.129**	0.096	**0.083**	0.118
	Median	**0.128**	0.042	**0.017**	0.127
Sensitivity	Mean	0.057	0.062	0.041	0.028
	Median	0.053	0.005	0.000	0.029
Sentiment	Median	1.000	1.000	1.000	1.000
	Mode	1.000	1.000	1.000	1.000
Relatability Score	Mean	0.038	0.032	0.038	0.034
	Median	0.028	0.000	0.000	0.021
Imperative Score	Mean	**0.056**	0.036	**0.007**	0.059
	Median	**0.043**	0.000	**0.000**	0.000
ASD	Mean	**2.279**	0.369	**0.222**	0.985
	Median	**1.517**	0.316	**0.180**	0.917
ASH	Mean	**2.139**	0.343	**0.202**	0.919
	Median	**1.421**	0.290	**0.160**	0.823
TOC	Mean	**17.040**	3.244	**1.673**	10.404
	Median	**5.440**	0.580	**0.240**	3.150
CNE	Mean	**0.178**	0.097	**0.087**	0.103
	Median	**0.100**	0.000	**0.000**	0.000
CPN	Mean	**0.542**	0.209	**0.165**	0.318
	Median	**0.300**	0.100	**0.100**	0.200
LEN	Mean	**0.877**	0.166	**0.104**	0.431
	Median	**0.575**	0.120	**0.070**	0.320

As evident from Table 8, there are clear differences between *motivational/advisory* and *regular* comments across almost all features except the Sentiment score, which means that the features were chosen well and are indicative of whether a text contains advice or not. However, Sentiment score seems to be irrelevant to this particular research problem.

I used the results from Table 8 to determine which features were to blame for misclassification of the 250 test set comments. Specifically, I estimated typical thresholds for all feature scores for both *r/getdisciplined* and *r/todayilearned* classes and compared them with actual scores of the misclassified comments. If the actual score was below/above the estimated typical threshold, I assumed that the feature was responsible for misclassification. Table 9 shows the results of this analysis. I indicated how many comments out of the 250 misclassified ones had values significantly diverting from the threshold for each feature. The analysis was performed without the Sentiment feature, because its median and mode did not differ between classes. However, it includes a difference between ASD and ASH, which will be discussed in the following paragraphs.

Table 9: Features responsible for misclassification in the r/getdisciplined *vs.* r/todayilearned *experiment. The highest percentage values are bolded.*

Feature	Number of comments	Percentage of misclassified comments
Aptitude	133	53.2
Attention	137	54.8
Pleasantness	145	58.0
Sensitivity	136	54.4
Relatability Score	149	59.6
Imperative Score	131	52.4
ASD	208	**83.2**
ASH	205	**82.0**
ASD-ASH	233	**93.2**
TOC	186	**74.4**
CNE	110	44.0
CPN	159	63.6
LEN	199	**79.6**

A close analysis revealed that, for the majority of misclassified comments, almost all feature scores simultaneously were responsible for the misclassification. However, three features stood out as the most misleading: Average Semantic Depth (ASD) and Average Semantic Height (ASH) scores responsible for 83% and 82% of misclassifications respectively, and Sentence Length (LEN) responsible for 80% of misclassifications.

LEN was included in the Specificity Scores by Deshpande et al. (2010) because, according to them, longer sentences tend to be more specific. Therefore, LEN should be higher for the specific motivational comments from the *r/getdisciplined* dataset. However, it is reasonable to assume that some comments from *r/todayilearned* reached greater lengths because they were a part of a heated discussion about an interesting fact. This is why this feature could be misleading to my algorithm. It is worth noting that this problem was spotted by Deshpande et al. (2010) as well. They included LEN in their study because, while this feature is not informative enough of specificity on its own, it works as a part of the Specificity Scores feature set. Likewise, I decided to keep LEN as it was so that it could work together with other Specificity features that could be improved instead.

As for ASD and ASH, I calculated the expected difference threshold between these features based on data from Table 8 and found out that this difference being too small was responsible for 93% of misclassifications, as seen in Table 9. This suggests that ASD and ASH scores may not be an issue on their own, but the value difference between them needs to be at least around 0.1 for the text to be classified as *motivational/advisory* class. As seen in Table 8, the difference is relatively high in comments correctly classified as *motivational/advisory* (difference between means = 0.140), and relatively low for misclassified comments from that class (difference = 0.026). Analogically, the score is low for comments correctly sorted as *regular* (difference = 0.020), and higher for *regular* comments mistakenly classified by the algorithm into the *motivational/advisory* class (difference = 0.066).

Both ASD and ASH were a part of the Specificity Scores feature set. Ideally, a specific word would have high score for Semantic Depth but low score for Semantic Height, i.e. place deep down in the hypernymy/hyponymy hierarchy. This is not to say that these features are complementary. There is no arbitrary number to which ASD and ASH should always add up. For example, in Figure 8 ASD for the word *food* is 4 and ASH is 2, so they add up to 6. In contrast, for the word *attribute*, ASD is 2 and ASH is 3, so they add up to 5. However, even though there is no complementary relationship between these two features, a small or no difference between ASD and ASH clearly was misleading to the algorithm. This was therefore an important issue in my study.

Further analysis revealed that in some cases, Imperative Score and CPN were not calculated correctly. Table 10 shows an example of such a comment.

Table 10: Miscalculated scores for a comment.

<table>
<tr><th colspan="7">Comment</th></tr>
<tr><td colspan="7">Since a fitness goal is done, maybe a goal in another aspect? Professional or mental maybe. “Be able to beat the hard level when playing chess against a computer”, something like that.</td></tr>
<tr><th>Aptitude</th><th>Attention</th><th>Pleasantness</th><th>Sensitivity</th><th>Sentiment</th><th colspan="2">Relatability Score</th></tr>
<tr><td>0.147</td><td>-0.028</td><td>-0.041</td><td>0.103</td><td>1.000</td><td colspan="2">0.000</td></tr>
<tr><th>Imperative Score</th><th>ASD</th><th>ASH</th><th>TOC</th><th>CNE</th><th>CPN</th><th>LEN</th></tr>
<tr><td>-0.050</td><td>0.396</td><td>0.396</td><td>1.580</td><td>0.000</td><td>0.200</td><td>0.210</td></tr>
</table>

The comment from Table 10 has Imperative Score of -0.050, which indicates that the feature was penalized for including a question mark, but the algorithm did not detect the non-infinitive verb *Be* after a quotation mark. With Count of Proper Nouns (CPN), the score from Table 10 suggests 2 proper nouns in the comment, even though there are none in the text. This is because in some cases capitalized words were counted as proper nouns by the NLTK tagger that was used in the algorithm. Both the Imperative and CPN scores for this comment suggested the need to improve pre-processing.

Moreover, relatively high Total Occurrence Count (TOC) scores across the entire dataset indicated another potential mistake in the calculations. It was expected that the motivational comments, which would be more specific, would have lower TOC. As evident from Table 8, this was not the case. Also, as seen in Table 9, TOC was responsible for as much as 74% of misclassifications.

Out of the 250 misclassified comments, 177 comments (70.8%) were *r/getdisciplined* comments misclassified as *regular*. I looked through some of them and found out that, in fact, a lot of these comments were simply mislabeled. Concretely, I labeled them as *motivational/advisory* because they came from the motivational subreddit, but there was no motivational content in their texts. The comments instead included bare links, short replies tangentially related to the struggles of the original poster, or in some cases negative remarks about the poster. I estimated that of the 177 comments, around half suffered from this problem. This means that the algorithm classified them correctly as non-motivational, but they were marked as misclassified because of their misleading labels. This is the reason that the motivational class had such low recall, which brought down the overall F-score.

A few conclusions can be drawn here. First, since there were so many comments that were mislabeled rather than misclassified, it can be assumed that the actual error rate of the algorithm would be reduced by around 40-50% if the labels were correct on all these comments. Secondly, I therefore understood that there was a need to obtain better data for my study. This conclusion was also supported by lower accuracy scores for the *all-motivational* vs. *all-non-motivational* experiment, which had much bigger datasets. It is reasonable to assume that the lower score was caused by having proportionally more noise in larger datasets. Thirdly, some changes needed to be made to the feature calculations so that the values were more accurate. Finally, I decided to implement bigger neural networks to combat the avoidable bias issue.

5.4 Deep network experiments

5.4.1 Datasets

To ensure quality of data, I decided to change the way that the comments were downloaded from advisory subreddits. Until that point, I downloaded all available comments from all available threads. However, this caused there to be a lot of noise in my data. Therefore, this time I only took the top-rated comments for each post instead of all of them. Concretely, I only searched for posts with more than 5 comments – which to some extent guarantees that a thoughtful discussion was going on – and out of those I only downloaded the top 3 or 5, depending on the subreddit (the details are given below). This ensured that the lowest rated comments got discarded, highly limiting noise in the data.

Since the *r/getdisciplined* subreddit was not very active at the time of this study, many of the posts I used before had less than 5 comments, which would now exclude them from the dataset. Because this significantly limited the amount of available data, I decided to rely on a different training method than before. Specifically, instead of training the algorithm just on comments containing motivational advice, I opted to download various advisory comments, pre-train the algorithm on this data, and then fine-tune it exclusively on motivational data. This is known as transfer learning and has proved to be effective in situations where there is little training data available for the problem at hand, but large amounts of other, similar data can be used as a starter. Since at that point there were no strictly motivational features that I was analyzing – i.e. all the text features could be applied to general advisory texts as well – I hypothesized that transfer learning should work well for this study, which was later confirmed by my results.

For pre-training, I downloaded new data from *r/relationship_advice*, *r/legaladvice* and *r/Advice* for the *motivational/advisory* class and from *r/todayilearned* and *r/pics* for the *non-motivational* class. For each post from the advisory subreddits, I took only the top 5 comments. I randomly shuffled the datasets separately and then divided each by hand into training set, development set and test set, where the development set and test set have about the same amount of data; the slight discrepancy comes from the fact that I did not want to divide comments of the same thread between different sets. I concatenated all training sets into one and then randomly shuffled it again. The same process was applied to the development set and test set. Table 11 shows the amount and division of data.

Table 11: Training, development and test data for pre-training the algorithm.

Class	Subreddit	Number of comments	Training set	Development set	Test set
Advisory	*r/relationship_advice*	4,485	3,588	448	449
	r/legaladvice	3,265	2,612	326	327
	r/Advice	4,245	3,396	424	425
Non-advisory	*r/todayilearned*	4,830	3,864	483	483
	r/pics	3,399	2,719	340	340
	Total	20,224	16,179	2,021	2,024

For fine-tuning, I downloaded a new dataset from *r/getdisciplined*. To ensure that there will be no mislabeling problem – essentially, that all comments downloaded from this subreddit would actually be motivational – I downloaded only top 3 comments from each thread that had more than 5 comments in total. I was more restrictive here than with the datasets for pre-training, because the *r/getdisciplined* dataset is of more importance to my study, while with pre-training I wanted to obtain as much data as possible. I downloaded 889 comments in total. As negative examples, I used the original *r/todayilearned* dataset, but limited in quantity to match the motivational class. Overall, there were 1,778 examples for the fine-tuning experiments. These experiments were performed using the 10-fold cross validation method, where each training set had 1,600 examples and each test set had 178 examples.

For both classes, the features were calculated according to changes made to the algorithm. These changes will be described in Section 5.4.2 below.

Additionally, I performed some experiments on shallow neural networks using the very same dataset. These experiments were meant to provide a comparison with my initial results described in Sections 5.3.3 and 5.3.4 above as well as with each other. Therefore, to reflect the way that the original experiments were done, I used a training set and a test set that were fixed. Table 12 shows the division of this data. The division was done using the same method as described above: by randomly shuffling the separate datasets, dividing them by hand into training and test sets, concatenating both classes into those sets and then randomly shuffling them again within themselves.

Table 12: Data division into training set and test set for shallow neural network experiments.

Class	Subreddit	Number of comments	Training set	Test set
Motivational	*r/getdisciplined*	889	641	248
Non-motivational	*r/todayilearned*	889	641	248
	Total	1,778	1,282	496

5.4.2 Improvements to the algorithm

I introduced several changes to the feature calculations based on the results from the previous experiments, which are detailed below.

- The Sentics Scores were left the same.
- Sentiment score was removed, because it did not show any relevancy to the study.
- Relatability Score was left the same.
- For the Imperative Score, I removed the penalty for including a question mark, and instead excluded sentences and clauses starting with auxiliary verbs *do* and *have*. This has more or less the same effect, but would remove the issues stemming from deducting points for question marks. The new score range was therefore 0 to 1. Another change was removing the phrase *why don't you* from this feature, which instead was incorporated into the Advice Score described below.
- Average Semantic Depth (ASD) was left the same.
- I modified Average Semantic Height (ASH). Previously the algorithm counted the longest path from a word to its hyponym; now I decided to count the **shortest** path. This is because previous calculations put ASD and ASH at very similar values, which was an issue for the algorithm. Changing the calculation method would result in different values for ASH, which I believed could be more informative for the network.
- For Total Occurrence Count (TOC), I slightly modified the calculations to make them more accurate. Previously, for each word from a sentence, the algorithm was adding up three lowest lemma occurrences in WordNet of that word, and then summing these numbers up for all words in the sentence. After

summing up all sentence scores for the final text score, TOC indeed got quite high. I fixed the code so that now it would calculate the lowest occurrence count for each word (based on the rarest lemma of that word) and then add only three lowest occurrence counts in any given sentence, the way I intended it to.

- Count of Named Entities (CNE) was left the same.
- Count of Proper Nouns (CPN) was left the same. I intended to use a better tagger (specifically the Stanford CoreNLP Tagger (The Stanford Natural Language Processing Group, 2018)[25]) but the computations took up a disproportionate amount of time and were not feasible.
- Sentence Length (LEN) was left the same.

I also introduced two new features to try to combat the avoidable bias problem.

First, I combined Average Semantic Depth and Average Semantic Height into an additional ASHD feature by deducting the overall ASH score from the overall ASD score for each comment. This was done to reflect the difference between the two features. Just like ASD and ASH, the ASHD score was divided by 100 to scale it down.

$$ASHD = ASD - ASH \tag{9}$$

Secondly, I introduced Advice Score, which was defined as the count of advisory expressions in the text. For this purpose, I prepared a list of possible advisory expressions that can be seen in Table 13. This list was compiled after thorough analysis of the best-rated *r/getdisciplined* comments and comprises the most frequent advice expressions that appeared in those comments. This feature also counted website links. Often, to offer advice to the original poster, a commenter mentioned a website link that contained solutions to the poster's problems. Therefore, in the preprocessing stage I replaced all links with the token *[link]*, which was then used by the Advice Score as an advisory expression. The overall comment score was divided by 10.

$$Advice\ Score = count\ of\ advice\ expressions \tag{10}$$

[25] https://nlp.stanford.edu/software/tagger.shtml

Table 13: A complete list of advice expressions used by the Advice Score feature.

you need to	*have you thought about*
op needs to	*have you tried*
you have to	*how about*
op has to	*if I were you*
it might be worth	*recommend*
I would	*suggest*
it would be good to	*advise*
it might be good to	*you could always*
you had better	*have you considered*
you'd better	*why not*
your only option is	*why don't you*
[link]	

All in all, I had 14 features in the modified feature set: the four Sentics Scores, Relatability Score, modified Imperative Score, Advice Score, Average Semantic Depth, modified Average Semantic Height, ASHD being the difference between Average Semantic Depth and Average Semantic Height, modified Total Occurrence Count, Count of Named Entities, Count of Proper Nouns and Sentence Length.

5.4.3 Technical details of deep models and comparative shallow models

To potentially reduce the avoidable bias, I decided to use a deep neural network with a longer training time. To find the best hyperparameter values, I conducted hyperparameter search using the fine-to-coarse search method. At first, I randomized the hyperparameter values between two set numbers and in the following rounds I kept narrowing this interval with respect to values that showed the best performance on the development set. I tuned five hyperparameters in total: the initial learning rate α, the number of epochs, the number of layers, the number of neurons in each layer and the mini-batch size for gradient descent. The β_1, β_2 and epsilon values for Adam (Kingma and Ba, 2014) were set at their default values. Weights for all the deep network models were initialized using the Xavier method (Xavier and Bengio, 2010).

I performed the experiments using the entire dataset, divided into *advisory* and *non-advisory* classes as shown in Table 11. I chose the model that performed best on the development set. The hyperparameter values of this model are summarized in Table 14.

Table 14: Hyperparameter values for the chosen deep neural network model.

Hyperparameter	Value
input layer units	14
number of hidden layers	3
hidden layers units	[7, 8, 15]
hidden layers activation function	tanh
output layer units	1
output layer activation function	sigmoid
initial learning rate	0.0037
epochs	30,000
mini-batch size	the entire batch
optimizing algorithm	Adam
β_1 for Adam	0.900
β_2 for Adam	0.999
epsilon for Adam	10^{-8}

The chosen model was fine-tuned on the new motivational dataset described in Section 5.4.1 by retraining all layers. The model was re-trained for 10,000 epochs with the same hyperparameter values as specified in Table 14. The fine-tuning experiment was done with the k-fold cross-validation method, where k=10. To prevent overfitting, I used dropout (Srivastava, Hinton, Krizhevsky, Sutskever and Salakhutdinov, 2014) with keep probability = {1.0, 0.95, 0.9, 0.8} for each model.

Two additional experiments were performed. First, I fed the new *r/getdisciplined* versus *r/todayilearned* dataset into the shallow neural network from my first experiment (described in Section 5.3.2 of this thesis) to see whether improving data quality helped with the results. For the sake of comparison, I performed a similar experiment on the same model with new data, but using the old feature set, to see how much I gained just from improving the features.

5.4.4 Results of experiments with deep networks and comparative shallow networks

The pre-trained deep neural network achieved 0.829 accuracy and 0.853 F-score on the test set. Table 15 presents detailed results for this model.

Table 15: Results for the best model with 3 hidden layers.

Dataset	Accuracy	Precision	Recall	F-score
Training set	0.838	0.870	0.856	0.863
Development set	0.821	0.849	0.850	0.849
Test set	0.829	0.870	0.837	0.853

After fine-tuning, there was accuracy of 0.843 and F-score of 0.854 on the test set. Detailed results are shown in Table 16. I report here the best mean F-score results that I got on the test set, which were with dropout keep probability = 0.95.

Table 16: Results for fine-tuning the network.

Dataset	Accuracy	Precision	Recall	F-score
Training set	0.890	0.923	0.896	0.909
Test set	0.843	0.874	0.837	0.854

With the shallow neural network trained and tested on the new dataset with new features, I achieved accuracy of 0.871 and F-score of 0.874. Table 17 presents the results.

Table 17: Results for the shallow neural network with new datasets and new features.

Dataset	Accuracy	Precision	Recall	F-score
Training set	0.888	0.908	0.863	0.885
Test set	0.871	0.854	0.895	0.874

Finally, for the shallow neural network model trained on new data but with the old feature set, I observed accuracy of 0.855 and F-score of 0.855. The results are presented in Table 18.

Table 18: Results for the shallow neural network with new datasets and old features.

Dataset	Accuracy	Precision	Recall	F-score
Training set	0.874	0.900	0.841	0.869
Test set	0.855	0.855	0.855	0.855

For the sake of clarity, I gathered all the test set results into Table 19 below. The model labeled as *Baseline* is my original shallow neural network model from the initial experiments, trained and tested on the old *r/getdisciplined* versus *r/todayilearned* dataset. *Deep model* is the deep network trained on general advice data. *Fine-tuned* means the *Deep model* after fine-tuning it with the new motivational dataset. *Shallow new data* is the old shallow network model trained on new motivational data. *Old features* is the same model as *Shallow new data* except it was trained on the old feature set. For each metrics, the best results across all models are in bold.

Table 19: Comparison of results for all the models.

Model	Accuracy	Precision	Recall	F-score
Baseline	**0.875**	0.844	0.691	0.760
Deep model	0.829	0.870	0.837	0.853
Fine-tuned	0.843	**0.874**	0.837	0.854
Shallow new data	0.871	0.854	**0.895**	**0.874**
Old features	0.855	0.855	0.855	0.855

5.4.5 Error analysis and discussion

There was a significant increase in F-score for all models trained on new data. This was related to greatly improved recall scores. The change was especially evident with the shallow network with new data and new features, where recall rose to 0.895 compared to the previous 0.691. This is clear evidence that I was able to fix the mislabeling problem in my datasets by obtaining better data and that this in turn led to improved results.

In fact, data mislabeling seems to have been the most significant problem for the previous algorithm, because the shallow network achieved good results on new data even with the old feature set. F-score was only 0.019 points lower compared to the network with new data and new features. This means that, while recalculating the features and adding more of them did yield some improvement, it was not as significant as I hoped.

With the fine-tuning experiment, the avoidable bias diminished: the error rate with respect to training set accuracy was 0.045 on average across all folds with no dropout (so in the same setting as with the original shallow network), which is low compared to the previous 0.128. However, even with dropout employed, test set accuracy did not improve above 0.875 of the original experiment. Moreover, shallow networks with new data, as shown in Tables 17 and 18, also achieved accuracy at the same level as the old network, even though their F-scores rose significantly.

There are also some additional observations that I made during the experiments.

First, in the hyperparameter search phase for the deep model, I have noticed that the only hyperparameter relevant to the results was the learning rate α. The number of layers, the number of units in each layer, the number of training epochs (after a certain point of 2-3,000) and the mini-batch size did not influence the results in any significant way.

Because of this, I settled for batch gradient descent and did not investigate networks deeper than 6 layers, to cut down the computational cost.

Moreover, compared to the shallow neural networks where learning rate was effective at 0.2, for deep models I had to reduce it to 0.0037 to ensure an acceptable degree of convergence. Therefore, the training was also accordingly slower.

Finally, the cost function for the fine-tuned deep model plateaued at a rather high level of around 0.4, as can be seen in Figure 9.

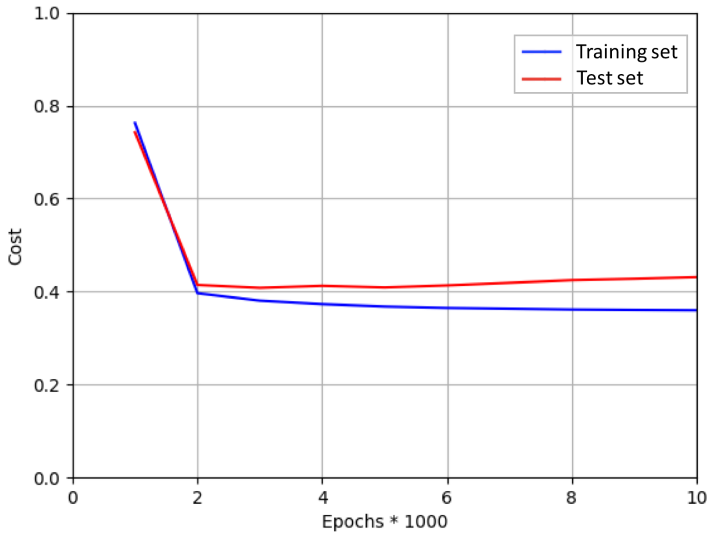

Figure 9: Cost function for the training set and test set for the fine-tuned deep model.

It seems that shallow networks – as well as the SVM from the original experiment – are better suited to solve this particular classification problem. They had better accuracy, recall and F-score than the deep models, as shown in Table 19. However, even shallow networks did not break the 0.875 accuracy barrier that I hoped to overcome. It seemed that no matter how the problem was approached, there was a non-removable cap on the results. I theorized that this was caused by the features. Although the 14 advice features performed rather well for this classification problem, they were not enough to achieve total accuracy. This is why for my next experiments, I significantly increased the number of features by utilizing my word2vec model.

5.5 Experiments with word2vec features

5.5.1 Technical details

Using the same *r/getdisciplined* vs. *r/todayilearned* datasets that were presented in Table 12, I performed experiments using word2vec textual features. This allowed me to not only utilize the 14 advice features of each comment, but also to capitalize on its semantic features expressed with word2vec embeddings. I obtained the vectors for each comment text using the same model and method as described in Section 5.1.

I performed spellchecking and contraction expanding on each comment text using code implementations available online from (MammothB, 2018)[26] and (Nealrs, 2014)[27] respectively. Once more, all non-ASCII characters were deleted.

There were two feature sets: one consisting only of word2vec text vectors and one where those vectors were concatenated with my 14 advice features. The word2vec features were normalized using L2 norm. The 14 advice features were left unchanged.

The experiments were conducted using the shallow network architecture, although for comparison I also used the deep network model described in Section 5.4.3. All experiments were performed using 10-fold cross-validation. The training set had 1,600 comments and the test set had 178 comments in each fold.

5.5.2 Results

Results are presented in Table 20. *Shallow* and *Deep* mean the shallow and deep models, respectively. *100 features* denotes experiments where only 100-dimensional word2vec vectors were used as features. *114 features* means experiments where the word2vec vectors were concatenated with the 14 advice features. *Acc*, *Prec Rec* and *F* denote accuracy, precision, recall and F-score, respectively. The table shows results up to the third decimal point to better display the minute differences between models. The best results in each column are bolded.

[26] https://github.com/mammothb/symspellpy

[27] https://gist.github.com/nealrs/96342d8231b75cf4bb82

Table 20: Results of experiments using word2vec word embeddings.

	Training set				Test set			
	Acc	**Prec**	**Rec**	**F**	**Acc**	**Prec**	**Rec**	**F**
Shallow 100 features	0.948	0.944	0.954	0.949	**0.912**	**0.910**	**0.917**	**0.913**
Shallow 114 features	0.948	0.950	0.947	0.948	0.904	0.906	0.900	0.903
Deep 100 features	**0.999**	**1.000**	**0.999**	**0.999**	0.880	0.878	0.884	0.881
Deep 114 features	**0.999**	**1.000**	**0.999**	**0.999**	0.899	0.893	0.906	0.899

5.5.3 Discussion

The experiment results confirmed that shallow networks worked better for this task. Deep networks overfit to the training set and were worse at generalizing to the test set. The variance was smaller with the shallow models.

The experiments also showed that whether the 14 advice features were incorporated or not did not make much difference. This can be explained mathematically by the way neural networks work. In fully connected networks, each layer's activation values are computed by multiplying the previous layer's activations by the current layer's weights, adding a bias term and putting the results through a non-linear function. Each neuron from the current layer is connected to each neuron from the previous layer. This means that each neuron in the current layer takes as input all values from the layer before it. In this case, the first hidden layer of the network took all 100 or all 114 feature values. It is possible that compressing all these values into each single neuron in the hidden layer erased the significance of the 14 additional features. In other words, as a feature number, 100 was not significantly different from 114.

However, previous experiments already proved that the 14 advice features were very informative to the neural network performing this particular classification task. Word embeddings by themselves also gave good results. Therefore, I decided to employ an original network architecture to leverage the strengths of both feature sets while avoiding overshadowing one by the other. I propose an approach to classification where I use two shallow networks, one trained solely on vector embeddings and one trained solely on 14 advice features. To use both of these feature sets in classification, I use the output of the word embeddings network to decide the input to the advice features network. From this point on, I will call them *Embeddings Network* and *Advice Network*, respectively.

5.6 Proposed method

5.6.1 Multiple networks for data classification

The approach proposed in this section is not the first to use multiple neural networks in classification. For example, Cho and Kim (1995) propose a general method of classification where they combine output from multiple networks trained on the same data to decide on the final result. This is done via fuzzy membership function, in which each network has its own weight in the final decision. Jerebko, Malley, Franaszek and Summers (2003) describe a method of classifying CT scans for presence of colonic polyps in radiology, where they determine 12 features and distribute their random subsets between multiple networks. Each network computes its own results and gets to vote for the final decision, with a weight being given to it based on its performance levels. Tsuda, Shin and Schölkopf (2005) show a method of combining multiple networks for protein classification and determining those that were the most important for obtaining good results. What all these papers have in common is that they see multiple networks as a pool of voters on the final decision. In a way, my method can also be regarded as voting, except that the networks do not vote together. Instead, they are connected in a linear way where the second network only works with the examples that were voted on as class 1 by the first network. However, for the sake of comparison, I performed experiments using my trained models both as a pool of voters and as part of my proposed method to see which gave better results. These experiments are presented in Section 5.6.4 below. Finally, Jeatrakul, Wong and Fung (2010) employ two neural networks called the Truth Network and the Falsity Network to compute the degree of truth/false membership in classification tasks. They seek examples prone to misclassification by both networks and remove them from the training dataset as noise. In their case however, the input to both networks is the same, while in my method the two networks work with two different sets of features. Moreover, I do not look for misclassified examples as compared with the true label, but rather I analyze the classification results of both networks in and of themselves.

More than one network is also used in the technique called Generative Adversarial Networks (Goodfellow et al., 2014), however, in that case both networks have different purposes and compete with each other. This is different from my goal, because I want both networks to work together on solving the exact same task.

Siamese networks (Koch, Zemel and Salakhutdinov, 2015), originally created for face verification, also employ two neural networks. They share the weights and architecture, and give outputs of identical type on two different input images, which is then used for the final verification. However, I do not compare outputs of my networks; doing that would be similar to actual voting. Secondly, as will be discussed in Section 5.6.5, in my method there is no obligation for both networks to be identical.

Gkioxari, Toshev and Navdeep (2016) presented a method of chaining convolutional neural networks for the task of human pose prediction from images. They implemented an architecture similar to sequence-to-sequence models, where the output of each model depended on its input combined with the outputs of previous models. In this method, the outputs are plugged as a part of input to each subsequent network. In my experiments, I use the output of one network to clean the data that is the input to the second network, in a way that will be described in the next section. This is different than actually inputting the class label provided by the first network to the second one.

5.6.2 Outline of the proposed method

The main purpose of this part of my research is to extract advice texts from a dataset downloaded arbitrarily from an online source. Consequently, more than classification itself, the goal is to remove noise, which in this case means regular, non-advisory texts.

Therefore, I propose to perform the classification twice, each time using a different network with a different feature set. By different feature sets, I mean the 100 word2vec features as one set and the 14 advice features as the other set. The *Embeddings Network* uses the word2vec features while the *Advice Network* uses the 14 advice features. Figure 10 illustrates the general flow of data through both networks.

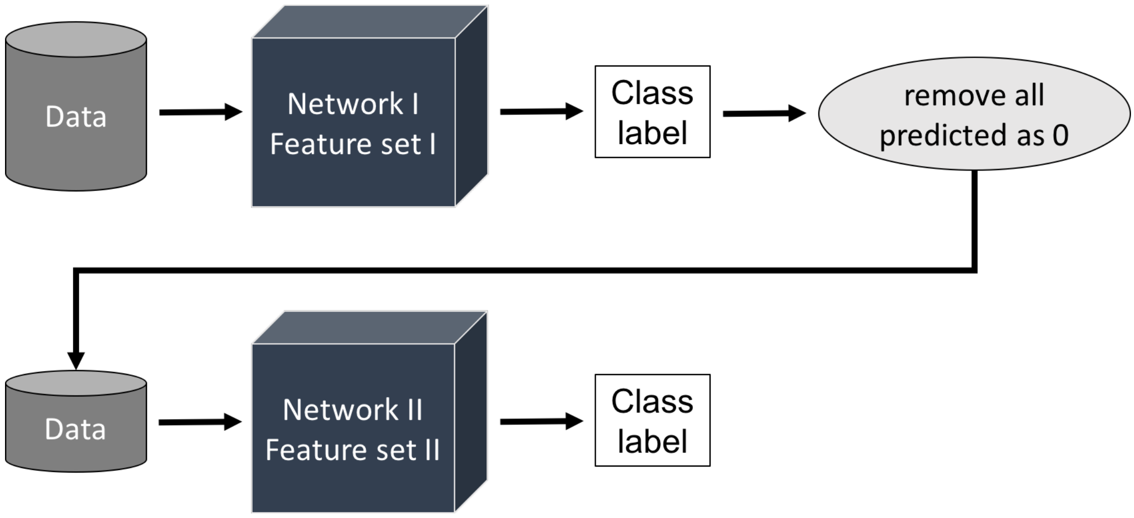

Figure 10: Illustration of the proposed method.

As seen in Figure 10, there are two stages of classification. The first network classifies data according to the results of its training. After this first stage of classification is performed, each data example has a predicted label 0 (meaning *regular*) or 1 (meaning *advisory/motivational*). Then all examples classified by the first network as 0 are removed from the dataset. While this may mean throwing away valuable data that was misclassified, it should also significantly reduce the amount of data irrelevant to the goal of this research. In other words, it should remove regular texts that do not contain any advice. After the removal, this cleaned dataset is then input into the second network, which confirms the predicted label as 1 or classifies the text as 0, contrary to the first network. Only the texts classified twice as 1 are regarded as advisory/motivational at the end.

Figure 10 contains generic names like *Network I* and *Network II*. This is because either network – *Embeddings* or *Advice* – could be the first classifier and the other one could be the second. The reason it can be this way is that both networks performed rather well on the 10-fold cross validation experiments. If one of them had significantly lower scores, then it could not be used as the first network in the chain, because it would not be effective in removing noise or correctly classifying relevant data. However, as will be discussed in Section 5.6.5, different orders have different strengths and weaknesses.

For both networks, I used the models that were already trained. For *Embeddings Network*, this was the 10 models that were trained for experiments described in Section 5.5 of this chapter. For *Advice Network*, the model was similar to the one labeled as

Shallow new data from Section 5.4. That model, however, was meant to provide a comparison with the original experiments, so just like in those experiments, it had one fixed training set and test set. This time I trained the same model in a 10-fold cross validation fashion, where each fold had 1,600 and 178 comments in training and test sets, respectively.

The data used for training of both *Embeddings Network* and *Advice Network* is the same data that was presented in Table 12. For each data example, I obtained a 114-feature vector, where the first 100 features were word embeddings and the last 14 features were the advice features. Each data example was input to the two classifiers with its corresponding feature set. For example, if the first classifier was the *Embeddings Network*, then the input was the 100 word2vec features. Then, if the text was classified as 1, it was input to the *Advice Network*, this time using its 14 advice features.

The data that was used for testing was a new dataset downloaded from *r/getdisciplined* and *r/todayilearned*. This was to ensure that there was no bias in choosing the best model (this process will be explained below). Table 21 presents the data.

Table 21: Datasets for testing the proposed method.

Class	Subreddit	Number of comments
Motivational	*r/getdisciplined*	279
Non-motivational	*r/todayilearned*	300
	Total	579

I performed two experiments. In the first one, I chose the best *Embeddings Network* model as the first classifier. This model was chosen based on its performance on the test set in the 10-fold cross validation training with regards to F-score. Therefore, the previous test sets for the 10 models essentially became my development sets. I input the results from *Embeddings Network* to each of the 10 models of *Advice Network* and obtained mean accuracy, precision, recall and F-score. The second experiment switched the order of the networks. This time the first classifier was the best model of *Advice Network* and the second classifiers were the 10 models of *Embeddings Network*. Figures 11 and 12 illustrate how the experiments are performed with both network orders.

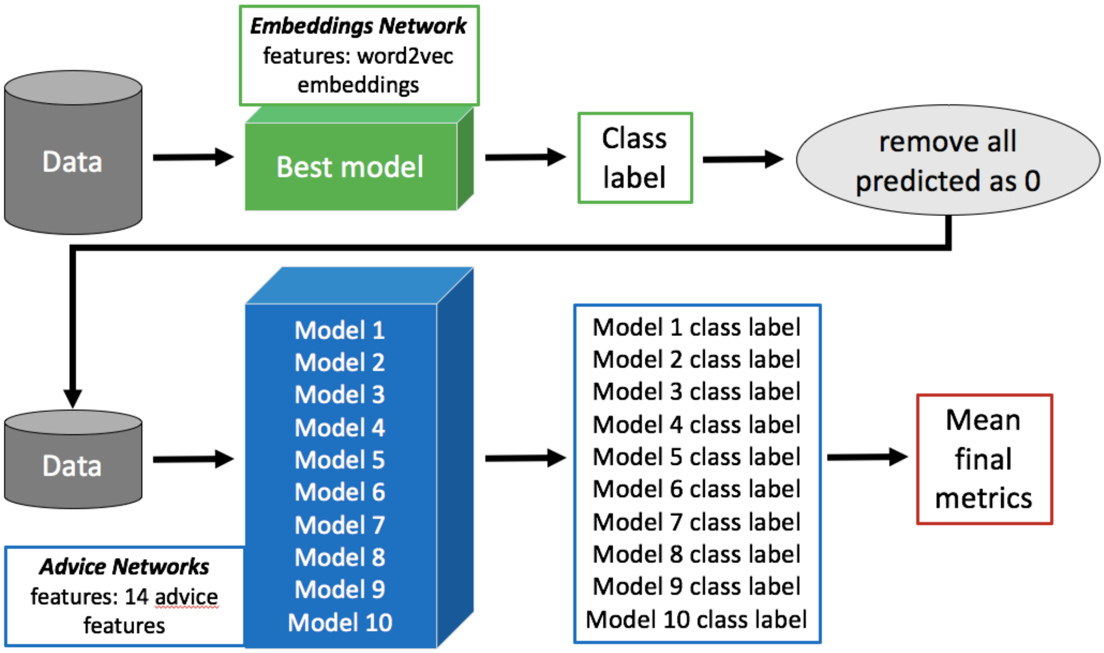

Figure 11: Experiment outline for the Embeddings-Advice *order.*

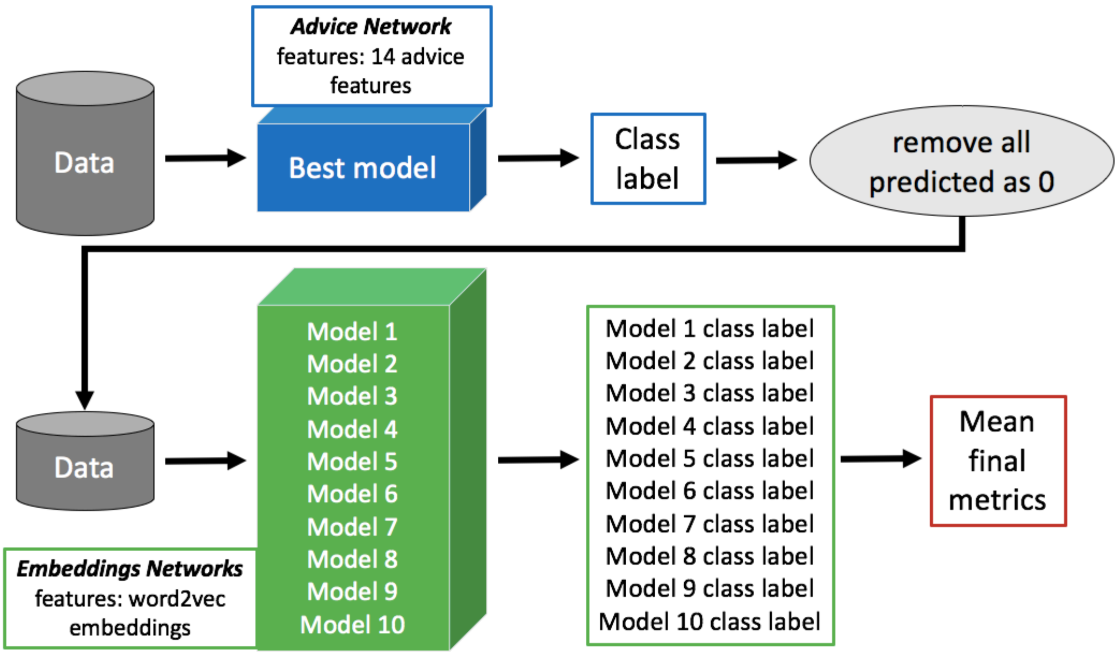

Figure 12: Experiment outline for the Advice-Embeddings *order.*

5.6.3 Results

Results of the experiments are presented in Tables 22 and 23. The first row presents results for the chosen first classifier – the best *Embeddings Network* or *Advice Network* model – on its previous test set. The second row presents its results on the current test set shown in Table 21. Row number three shows results on that test set after removing comments predicted as 0. Obviously, recall for this row is 1.0, since I removed one predicted class, effectively getting rid of both true negatives and false negatives. Accordingly, the F-score rises as well. Therefore, this row is presented only for reference purposes. Row number four shows mean results for the 10 models that served as the second classifier.

With the experiment where *Embeddings Network* was the first classifier and *Advice Network* was the second one, there were 289 examples left in the dataset after removing predicted class 0. There were 22 examples in that removed data that were misclassified as 0 but actually had the true label 1. This means 7.88% loss of motivational/advisory data during the classification. Table 22 presents the results.

Table 22: Results for the order Embeddings Network – Advice Networks. Best results in each column are bolded (with the omission of the *After removal* row which is only present for reference).

Step	Accuracy	Precision	Recall	F-score
Previous test set	**0.944**	0.922	**0.947**	0.934
Current test set	0.888	0.889	0.921	0.905
After removal	0.883	0.889	1.000	0.941
Mean for Advice Networks	0.838	**0.965**	0.922	**0.943**

In the experiments where *Advice Network* was the first classifier followed by 10 models of *Embeddings Network*, there were 267 examples left in the data after the removal. 34 advisory/motivational examples were lost in that removal, which is 12.18% of the class 1 dataset. Table 23 presents the results of classification.

Table 23: Results for the order Advice Network – Embeddings Networks. Best results in each column are bolded (with the omission of the *After removal* row which is only present for reference).

Step	Accuracy	Precision	Recall	F-score
Previous test set	0.904	0.894	0.905	0.899
Current test set	0.827	0.918	0.878	0.897
After removal	0.842	0.918	1.000	0.957
Mean for Embeddings Networks	**0.925**	**0.977**	**0.964**	**0.971**

Additional experiments revealed that in the last step of the classification, the *Advice-Embeddings* order lost on average 8.7 examples while the *Embeddings-Advice* order lost 20.1 examples. This is a significant difference. However, when looking at the amount of data lost in the first removal, these results add up to almost the same number. Table 24 summarizes these findings.

Table 24: The amount of relevant data examples lost in both orders in the proposed methods. The average results of 8.7 and 20.1 have been rounded.

	Embeddings-Advice	***Advice-Embeddings***
First removal	22	34
Second removal	20	9
Total	42	43
Percentage of relevant data	15.1%	15.4%

5.6.4 Comparison with a pool of voters

To confirm the effectiveness of my method, I compared it with a pool of voters.

Each pool was comprised of individual models trained in the 10-fold cross-validation fashion – so 10 models per pool. Just as with my proposed method, all models were trained on the *r/getdisciplined* and *r/todayilearned* data shown in Table 12.

There were 4 conditions:

1. Shallow network models trained with only the 100 word2vec features (10 models).
2. Shallow network models trained with only the 14 advice features (10 models).
3. Models from Condition 1. and Condition 2. pooled together (20 models). This was done to see whether the models trained separately on the two feature sets would be able to act as one pool and overcome the proposed method by voting together.
4. Shallow network models trained with all 114 features (10 models).

In other words, models from Conditions 1 and 2 were the same models used in my proposed method. Conditions 3 and 4 combined the two feature sets in different ways to see how that compares to my combination of *Embeddings* and *Advice Networks*.

For each condition, I ranked the models based on their performance on the previous test sets with respect to F-score. Then, these models were tested on the new dataset that was presented in Table 21.

As mentioned in Section 5.6.1, usually not all voters in the pool have the same weights. Based on performance, some of them get more significant votes than others. To reflect that, I did not use all voters in the pool. Instead, I used only top 9, 5, 3 and 1 for each condition. I chose odd numbers so that there would always be a majority in voting. As the final prediction, I chose the class that most models voted on. I then compared this prediction against the true label and calculated all metrics from that.

For comparison, I also calculated the mean results of all models for each condition except Condition 3, which was just models from Condition 1 and Condition 2 pooled together.

Results are presented in Table 25.

Table 25: Results for the experiments involving a pool of voters. For the Both 100 and 14 features *condition, which included 20 models (10 from* 100 features *and 10 from* 14 features*) all best models came from the* 100 features *condition, which makes their results identical. This is indicated by the* Same as 100 *label. Best results in each row are bolded.*

	Condition	**100 features**	**14 features**	**Both 100 and 14 features**	**114 features**
Best 9	**Accuracy**	**0.921**	0.907	Same as 100	0.919
	Precison	**0.921**	**0.921**	Same as 100	0.920
	Recall	**0.914**	0.882	Same as 100	0.910
	F-score	**0.918**	0.901	Same as 100	0.915
Best 5	**Accuracy**	**0.919**	0.905	Same as 100	0.914
	Precison	0.911	**0.924**	Same as 100	0.916
	Recall	**0.921**	0.875	Same as 100	0.903
	F-score	**0.916**	0.899	Same as 100	0.910
Best 3	**Accuracy**	**0.917**	0.900	Same as 100	0.914
	Precison	**0.914**	**0.914**	Same as 100	0.910
	Recall	**0.914**	0.875	Same as 100	0.910
	F-score	**0.914**	0.894	Same as 100	0.910
Best 1	**Accuracy**	0.907	0.903	Same as 100	**0.919**
	Precison	0.889	0.918	Same as 100	**0.920**
	Recall	**0.921**	0.878	Same as 100	0.910
	F-score	0.905	0.897	Same as 100	**0.915**
All (mean)	**Accuracy**	**0.896**	0.828	0.862	0.895
	Precison	**0.914**	0.911	0.912	**0.914**
	Recall	**0.911**	0.870	0.891	0.909
	F-score	**0.912**	0.890	0.901	0.911

5.6.5 Discussion about proposed method

As evident from Tables 22 and 23, I was able to improve on the results provided by either feature set alone. The F-score rose to 0.943 with the *Embeddings-Advice* order and to 0.971 with the *Advice-Embeddings* order. Moreover, the method beats all combinations of the pool of voters as well as 114 features used together, when compared to results from Table 25. As such, the proposed method proved to be effective for classifying texts into *motivational/advisory* and *regular* categories.

The *Advice-Embeddings* order performed significantly better in the last step. As seen in Table 23, the proposed method beat all results from previous steps across all metrics. That experiment meant the loss of almost 13% of relevant data in the first step. In the other experiment, the initial loss was only around 8%, but F-score was lower by 3% and accuracy by as much as 9% in comparison, as seen in Table 22. Accuracy and recall were also lower when compared to the results from previous steps in Table 22. This means that the *Embeddings-Advice* order performed significantly worse. However, in the end, both network orders lost more or less the same amount of data when both the first step and the last step are considered, as seen in Table 24.

Since my main goal was to extract advisory texts from an unknown dataset, the metrics that matters most is precision. Precision is defined as the ratio of true positives to all positives. In other words, it shows how many comments predicted as motivational/advisory truly belonged to this class. The higher the precision, the less noise there is in the extracted data. Precision was high in both experiments, reaching 97% or 98%. This is the main reason that my method is effective: it reduces the noise to 2-3%. In this light, both orders performed equally well. However, these findings may have significance for other research projects that employ a similar method, where other metrics might be important to the final goal. I postulate that the order of the networks, if computationally feasible, should be determined experimentally and not in an arbitrary manner, to best suit the needs of the research.

The proposed method needs further exploration in other areas as well.

First, it may be effective for other text classification problems as well. Semantic word embeddings have proven to be useful in all kinds of textual data processing. Normally, they are concatenated with any additional features used in experiments, but simple concatenation may obscure the significance of those additional features by processing them together with all the numerous vector values. This was discussed in Section 5.5.3 of this thesis. My method alleviates this problem by giving proper attention to each feature set by using them as input to their own separate neural networks. Perhaps the same method can be used for other text classification tasks where input features consist of both word vectors and original features devised for the task at hand.

Secondly, it is important to note here that both networks were solving the same classification task and that the two feature sets were not entirely independent of each other. Both the 100 word2vec features and the 14 advice features were calculated based on the same text, so while they emphasized different qualities of that text, some information may have been shared between the sets, even if not in a readily visible way. It is possible that connecting the two networks allowed me to approach the same text qualities in two different, mutually complementing ways. It remains to be seen whether this relationship between feature sets is important to the results or not. This would, however, require applying the method to a different task with different feature sets, which is beyond the scope of this thesis.

Thirdly, the proposed method was devised for binary classification. It might be interesting to create a similar method for multiple classes. Further studies on this topic could include exploring the relationship between final results and data loss for each relevant class.

Last but not least, I was able to leverage both methods in this way because both networks worked well enough by themselves. This was briefly mentioned in Section 5.6.2 while outlining the method. If any of the networks had given lesser accuracy, for example around 0.7, this probably would have badly influenced the final results. This means that my method may only be useful in settings where both feature sets do well on their own, or at least one of them makes up for the shortcomings of the other.

There are various benefits to the proposed method, like its simplicity, predictable computational cost (which is the combined cost of all networks involved) and apparent effectiveness. However, there is still room for creative improvement. For example, even though in this case I used the same network architecture for both networks, this is not obligatory. Since both networks are trained separately, their hyperparameter values can be easily re-adjusted on a network basis to fit the purposes of the research best. Moreover, perhaps more than two networks can be chained to gradually increase the accuracy of the last one.

5.7 Conclusions

This chapter presented the details of my research on distinguishing advice texts from regular texts. First, I have successfully determined 14 advice features for recognition of motivational/advisory texts. Secondly, I was able to devise a new way of improving the classification results. In my initial experiments, I achieved test set accuracy of 0.875 and F-score of 0.760 using a shallow neural network. I was able to improve the F-score by obtaining better data. After performing numerous other experiments in attempts to raise the final scores, I discovered a novel method for text classification by linking two shallow networks together in a combined effort to choose only relevant texts. In the end, the F-score improved to 0.943 or 0.971, depending on the order of networks.

There were two feature sets used in the experiments and each one performed reasonably well on its own, as evidenced by the mean results of all models in Table 25. This shows that motivational/advisory texts are different from non-advisory ones not only in visible ways that were represented by the 14 advice features, but also on a basic semantic level expressed in word2vec vector elements. Using both feature sets allowed me to capitalize on these primitive semantic qualities that are not readily visible to the human eye, as well as on the strengths of features designed manually after looking into the data.

The results of the experiments described in this chapter confirmed the importance of high quality data in studies involving deep learning. Just by obtaining better data I was able to significantly improve the classification results. This proves my original point that was stated in Section 1.2 and reinforced in Section 3.5 of this thesis: that the training data for a motivational dialogue system needs to be chosen carefully.

The studies presented in this chapter resulted in an algorithm that is able to distinguish motivational/advisory texts from regular ones. However, just sorting the data into these two classes does not guarantee that the chosen motivational/advisory texts would contain good advice. Therefore, I decided to take this research a step further. Using both the 100 word2vec features and my 14 advice features, I created a ranking algorithm for choosing only the best quality advice. This algorithm will be described in the next chapter.

Chapter 6: Advice ranking task

This chapter details my research into ranking advice texts. I have already explained the importance of choosing only the best advice texts for my final dialogue system elsewhere in this thesis (for example in Section 1.2). This part of my study solves the issue of sorting through the data for good quality advice that can then be used as training data or as a corpus of possible utterances for the dialogue system.

The main research objective was to rank motivational or advisory texts against each other within a group. In this case, I worked with comment groups of three, which I will refer to as **triples** from this point on. Within each triple, there was one best-rated comment, one second best-rated comment and one comment with the lowest score compared to the other two. The goal of my algorithm was to automatically reconstruct this ranking.

Section 6.1 mentions related research in text ranking. Section 6.2 describes the datasets used in the ranking experiments along with the way the data was prepared for the network. Section 6.3 presents the architecture of the network. Section 6.4 details the experiments, together with error analysis of the results and additional observations about the training process. Section 6.5 includes a discussion about the 14 advice features and their role in the ranking, as well as various findings concerning their values and importance. Section 6.6 concludes the chapter.

6.1 Related research in text ranking

Studies concerning text ranking usually involve criteria like relevance to the user’s query and are conducted as part of research in information retrieval. Documents in a given database are ranked according to their usefulness in providing the user with information about a particular topic. Some of the well-known algorithms that accomplish this goal are PageRank (Page, Brin, Motwani and Winograd, 1999), which is used by Google, and HITS (Hyperlink Induced Topic Search) by Kleinberg (1999), which was also meant to be utilized by Internet search engines; both of them analyze links between individual websites to determine which website is most relevant to the query. Most recent developments in the field include improving the tf-idf weighted ranking method with association rules (Jabri, Dahbi, Gadi and Bassir, 2018), incorporating user browsing patterns into sorting query search results (Sethi and Dixit, 2019) and utilizing the Hadoop MapReduce framework to improve search precision (Malhotra, Malhotra and Rishi, 2018). However, these studies are only partially relevant to my research problem. An effective document ranking algorithm, such as the ones mentioned above, would be useful in retrieving appropriate advice for the user based on their query and can be a point of focus in the next steps of this study, when working on creating motivational utterances for the final dialogue system. In contrast, this chapter describes an algorithm for ranking advice quality, which is a different issue and therefore a different approach must be used. The main difference is that the method proposed in this chapter ranks the texts against each other rather than by relevance to some external search term.

There are also numerous papers describing approaches to ranking texts for purposes other than answering the user's query. Fang, Mu, Deng and Wu (2017) present a sentence ranking algorithm for extractive text summarization. Vajjala and Meurers (2016) rank sentences in the text based on their readability while studying text simplification. Myangah and Rezai (2016) rank Persian texts based on their vocabulary richness and use this information to determine the genre of the text. Finally, Wei, Liu and Li (2016) conduct a study resembling the one described in this thesis, where they rank Reddit comments within pairs with regard to the comments' persuasiveness power using a Support Vector Machine. However, to the best of my knowledge, currently there is no algorithm for ranking advice texts based on advice quality other than the one I propose in this chapter.

6.2 Datasets

For the experiments described in this chapter I have used the *r/getdisciplined* and *r/Advice* datasets, so only motivational and advisory data was utilized. The data was downloaded in a similar way as for the classification experiments, which was described in Section 5.4.1 of this thesis. Essentially, I only downloaded threads with more than 5 comments, and out of those 5 comments I put only the top 3 into my dataset. This is the same way that the *r/getdisciplined* data was downloaded for the classification task, and once again, the purpose was to reduce noise to minimum. The 3 comments coming from the same thread formed a triple. This ensured that all comments within the triple contained advice on the same topic and as such could be compared against each other. Table 26 shows the breakdown of data.

Table 26: Datasets for the ranking algorithm.

Subreddit	Number of threads	Number of comments
r/getdisciplined	624	1,872
r/Advice	3,066	9,198
Total	3,690	11,070

As mentioned in Chapter 4 of this thesis, each comment on Reddit can be voted on by other users. Therefore, the quality of advice expressed in a comment can be measured with the number of upvotes and downvotes that the comment received. In other words, the best-rated comments were considered by the community to be the most motivating or to contain the best quality advice. Ideally, I would like to measure this quality by seeing whether the original poster upvoted the comment or not (meaning whether they were interested in that particular advice), but Reddit does not provide such information. Therefore, I relied on the general voting system. This had the additional benefit of crowdsourcing the ratings, which allowed me to exploit intersubjectivity among numerous Reddit users instead of just one person.

I used the scores to determine the ranking order of comments within each triple. It did not matter how big the gap was between each pair of scores; I was only interested in the ranks. Predicting the actual score of the comment would be more difficult, as it would mean labeling each comment with a real value with no upper limit instead of a rank from {1,2,3}. Moreover, it would require a large corpus of varied gaps between scores so that the data could be representative of all possible combinations. At the same time, it would not be more helpful than ranking in this research, since I was only interested in choosing the best comment, which can be indicated with a rank just as well as with a score. Making the task more difficult while gaining nothing in return would significantly detract from this study, which is why I opted for ranking.

I assigned rank 0 to the comment with the best score and ranks 1 and 2 to the following comments accordingly. This may seem unusual, as normally the ranks would be 1, 2 and 3. However, this happened for technical reasons which should become apparent in Section 6.2.1. Table 27 presents an example of a comment triple with the corresponding post and assigned ranks.

It should be mentioned here that the ratings were not representative across the entire dataset; for example, a comment ranked 0 in its own comment triple may have been ranked 2 in a different triple (given that they pertained to the same topic). However, this was not an issue, since my purpose was to select the best comment in any given fixed set of three. At the time of cleaning the dataset for training the dialogue system, this ranking method can be utilized in a tournament-like manner on all the advisory comments on the same topic to extract the best one from the dataset.

Table 27: An example of a triple with the features calculated for each comment.

Post text				Post score: 15		
I am addicted to sleeping. I think the reason for that is because I cannot tolerate my thoughts and the real world. But after spending years like this, I feel awful for sleeping so much. It’s not like I sleep 15 hours a day but this habit of mine leads to being absent for classes twice a month and skipping half of gym sessions. Above all I don’t bother to improve my life style. With this attitude of mine seeing any kind of future for myself is impossible! Can you give me tips and suggestions how to overcome this bad addiction? If you introduce a reading source, also I would appreciate it a lot. Edit: I don’t sleep 15 hours a Day but I am sure I am addicted to sleeping!						
Comment text				**Comment score: 20 (rank 0)**		
If you’re sleeping 15 hours a Day regularly for no apparent reason you need to see a doctor						
Aptitude	**Attention**	**Pleasantness**	**Sensitivity**	**Relatability Score**	**Imperative Score**	**Advice Score**
0.094	-0.119	0.123	-0.035	0.000	0.000	0.100
ASD	**ASH**	**ASHD**	**TOC**	**CNE**	**CPN**	**LEN**
0.157	0.152	0.005	0.000	0.000	0.000	0.090
Comment text				**Comment score: 7 (rank 1)**		
You associate your sleeping to not tolerating your thoughts. To achieve higher capacity in managing your thoughts, have you heard of Mindfulness work? It's a simple technique with effects showing already after a short while.						
Aptitude	**Attention**	**Pleasantness**	**Sensitivity**	**Relatability Score**	**Imperative Score**	**Advice Score**
0.179	0.245	0.136	-0.013	0.000	0.048	0.000
ASD	**ASH**	**ASHD**	**TOC**	**CNE**	**CPN**	**LEN**
0.460	0.364	0.096	0.000	0.000	0.000	0.180
Comment text				**Comment score: 3 (rank 2)**		
I can relate a little bit, as I too love sleep and try to avoid being alone with my own thoughts. I still love sleep, but finding podcasts I really like has helped me with the avoiding my thoughts part. Then I can use them as a bribe to myself..."I can only listen to this on my drive to work/walk to class." "I can only listen to this one at the gym." YMMV, but if you can find an addictive one, or one you find genuinely funny/entertaining, the bribery works. And then if you are one of those "I'm fine as long as I GET there" people for class/gym/work, you can look forward to the getting there part.						
Aptitude	**Attention**	**Pleasantness**	**Sensitivity**	**Relatability Score**	**Imperative Score**	**Advice Score**
-0.041	0.029	-0.084	0.114	0.100	0.042	0.000
ASD	**ASH**	**ASHD**	**TOC**	**CNE**	**CPN**	**LEN**
1.188	1.106	0.082	0.080	0.000	0.400	0.540

6.2.1 Data preparation

As mentioned in Section 5.7, the ranking task used the same features as the classification task. For each comment, there were 100 word2vec features and 14 advice features. Moreover, since a comment triple was one data example, all features from all three comments were input to the network together. Therefore, they were concatenated into one vector of 3*114=342 features.

Before concatenation, each triple was shuffled to obtain all 6 permutations of the rank order. This was done to prevent the network from overfitting to the order while ignoring the actual features and had an additional benefit of data augmentation. Therefore, each triple was present in the dataset 6 times, each time with different order of comments.

The desired output was a vector of length 3, where each position gave a number 0, 1 or 2 depending on the rank and order of the comments. For example, if the first 114 features of the input vector represented a comment of rank 2, the next 114 features represented a comment of rank 0 and the last 114 represented a comment of rank 1, then the expected output was a vector of [2, 0, 1]. This output was obtained with the *argmax* function, so naturally the order started at 0 and not at 1. In other words, this ranking task was approached as a classification task with multiple classes, where each comment was sorted into a class signifying its rank.

Overall, there were 22,140 data examples available for the task, which is the equivalent of 3,690 downloaded triples, where each triple was present in the dataset 6 times.

6.3 Network architecture

To ensure that each comment text went through the same initial calculations, I constructed a convolutional neural network (Fukushima, 1980), which are widely used in various classification tasks involving textual data (for example (Kim, 2014) or (Santos, Xiang and Zhou, 2015)). The first layer had 342 units that matched the input vector. Then, I used filters of length 114 and stride of 114, which essentially meant that each set of 114 comment features went through the same filter. This ensured that no matter the order of the comments, each one received equal treatment and had equal chances of being assigned any of the three ranks. Moreover, the calculations for each comment were not sequentially dependent on each other, the way it would happen with a recurrent neural network (which will be discussed in Section 6.4.3 of this chapter). Instead, the filtered features were passed up to the further layers, where the calculations were appropriately conducted together, but without any dependence on comment order.

On top of the first convolutional layer, there was a second one followed by three fully connected layers. Table 28 gives an overview of the network along with data shape produced by each layer and operation. *Conv* stands for convolutional layers and *Fc* stands for fully connected layers. For convolutional layers, the number of units is the number of filters multiplied by filter size used on the layer. Since the research was conducted using PyTorch, the order of dimensions for convolutional layers follows the PyTorch convention, which is: *depth* (= number of channels), *height*, *width*. The kernel size gives only height and width; depth is exactly the same as the input to the given layer. The label *m* is the standard notation for batch size.

Table 28: Overview of the convolutional network.

Layer	Number of units	Output shape
Input	---	(*m*, 1, 1, 342)
Conv1	114*(1, 114)	(*m*, 114, 1, 3)
reshape	---	(*m*, 3, 1, 114)
Conv2	3*(1, 3)	(*m*, 3, 1, 38)
reshape	---	(*m*, 3, 38)
Fc1	20	(*m*, 3, 20)
Fc2	10	(*m*, 3, 10)
Fc3	3	(*m*, 3, 3)
reshape	---	(3*m*, 3)
argmax	---	(3*m*, 1)

Figure 13 illustrates how one data example passes through the network from the input of shape (1, 1, 342) to the output of shape (3, 1). Once again, the *Conv* output dimensions follow the Pytorch convention of *(depth, height, width)*.

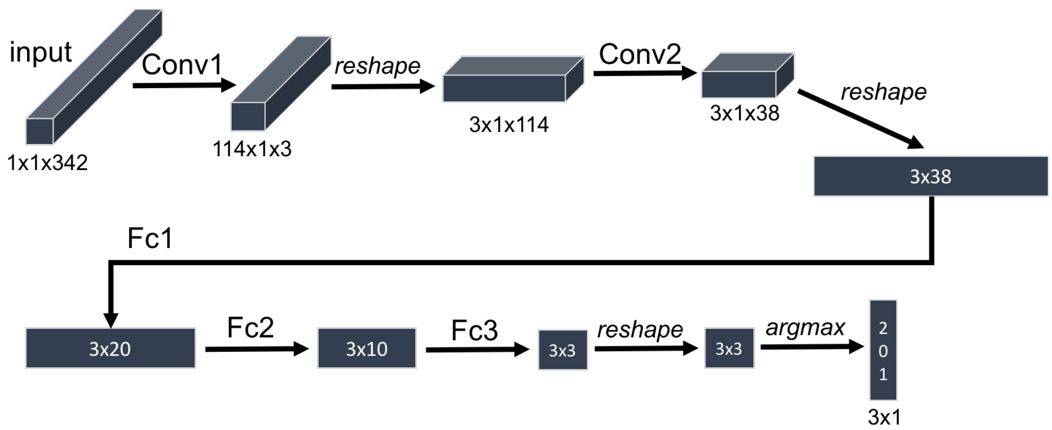

Figure 13: Illustration of the convolutional neural network.

The *reshape* operations had an important purpose. The first *reshape* was used to change the shape of the output of the first convolutional layer before passing it on to the next layer. *Conv1* gave output of shape (114, 1, 3) for each data entry, since the number of filters was also 114. So essentially, this output was a tensor of width 3, where each position represented one comment, and with a depth of 114, because each comment had been convolved by all 114 kernels. I reshaped the output into (3, 1, 114) and fed it as input to *Conv2*. This way, the kernels in the second convolutional layer operated on a tensor of width 114, where each position represented one *Conv1* kernel and had a depth of 3 representing the three comments. Each *Conv2* kernel processed three of the 114 positions (with each position incorporating calculations from all comments), yielding 38 results from the processing. The point of this reshaping operation was to allow the *Conv2* kernels to process subsets of *Conv1* kernel results with all three comments each instead of subsets of comments with all 114 *Conv1* kernel results each. It was important to include information about all three comments for each *Conv2* kernel operation, because the final results are dependent on all feature relationships within the comment triple.

The second *reshape* was applied to the *Conv2* results to reduce the number of dimensions, so that the tensor could be passed to a fully connected layer *Fc1*. The third *reshape* was for the output. *Fc3* produced a tensor of shape (m, 3, 3) which meant m examples with 3 comments each, where each comment had 3 possible ranks. This layer used a softmax activation function. In other words, for each training example, there were three comments to rank and each of these comments received its own softmax vector of length 3, where the rank was indicated by the position of 1. For example, if the softmax produced output of [0, 1, 0] for a comment, that comment got rank 1, and if the output was [0, 0, 1] then the comment got rank 2. The *reshape* operation reduced the number of output dimensions to two, where each comment in the entire batch had its own softmax row. Then an *argmax* operation was performed on the second dimension, resulting in a tensor of shape ($3m$, 1), meaning one rank for each comment in the batch. This was then easily compared to the tensor with true labels of the same shape.

6.4 Experiments

6.4.1 Technical details

The network was trained for 1,000 epochs using the Adadelta (Zeiler, 2012) optimization algorithm with no changes to its default hyperparameter values. This means that the learning rate was not set manually. I also divided the training data into minibatches of 512. All layers used the tanh activation function except for *Fc3*, which was a softmax layer. These hyperparameters were decided based on performance, which will be further discussed in Section 6.4.3.

The experiments were performed using a 10-fold cross validation method, where there were 19,926 training examples and 2,214 test examples in each fold. Before training and testing, the features were normalized using L2 normalization. This time I normalized both feature sets, but this operation was done separately. In other words, the first 100 features and the last 14 features of the feature matrix were normalized independently of each other. This was because the feature sets have very different feature values and I did not want to obscure the values of one set by normalizing it together with the other one.

6.4.2 Results

The performance of the network was measured with accuracy. Table 29 presents the results broken down by fold.

Table 29: Accuracy and loss for all folds on the training and test sets.

Fold	Training loss	Training accuracy	Test loss	Test accuracy
1	0.016	0.995	0.037	0.991
2	0.038	0.989	0.056	0.984
3	0.276	0.903	0.423	0.889
4	0.031	0.992	0.049	0.989
5	0.072	0.980	0.081	0.975
6	0.060	0.990	0.059	0.987
7	0.084	0.947	0.166	0.931
8	0.032	0.994	0.043	0.992
9	0.020	0.993	0.052	0.989
10	0.036	0.989	0.066	0.986
Average	0.067	0.977	0.103	0.971

I also analyzed the 14 advice features to see whether they correlated with the ranks. Although no single feature showed a significant linear correlation with any rank (as measured by Pearson coefficient), there were small differences in their mean and median values between ranks. Table 30 shows the comparison of raw feature values across the three ranks. I included more decimal points in the table to better reflect the differences. Table 31 presents statistical significance scores of the differences between feature values across rank pairs.

Table 30: Feature values across different ranks. The highest mean value for each feature is bolded.

Feature		Rank 0	Rank 1	Rank 2
Aptitude	Mean	0.119163	0.130136	**0.134223**
	Median	0.128197	0.141360	0.139275
Attention	Mean	**0.087951**	0.087693	0.082651
	Median	0.082036	0.084107	0.082451
Pleasantness	Mean	0.087890	0.099144	**0.100406**
	Median	0.087306	0.107313	0.095032
Sensitivity	Mean	0.051194	**0.051375**	0.050223
	Median	0.039533	0.037422	0.037500
Relatability Score	Mean	0.025531	0.027642	**0.028502**
	Median	0.009709	0.012500	0.012085
Imperative Score	Mean	**0.069907**	0.066100	0.066240
	Median	0.053333	0.048780	0.048780
Advice Score	Mean	0.025881	**0.027425**	0.026667
	Median	0.000000	0.000000	0.000000
ASD	Mean	1.000910	**1.047904**	1.004613
	Median	0.622520	0.657738	0.640000
ASH	Mean	0.943619	**0.988323**	0.947719
	Median	0.588571	0.620119	0.602750
ASHD	Mean	0.057292	**0.059582**	0.056894
	Median	0.030000	0.032348	0.031667
TOC	Mean	0.141257	0.140745	**0.159900**
	Median	0.000000	0.000000	0.000000
CNE	Mean	**0.000108**	**0.000108**	0.000054
	Median	0.000000	0.000000	0.000000
CPN	Mean	0.049593	**0.052602**	0.047940
	Median	0.000000	0.000000	0.000000
LEN	Mean	0.354938	**0.378932**	0.359827
	Median	0.200000	0.220000	0.220000

Table 31: Statistical significance of differences between feature values across rank pairs. p values of 0.05 or lower are in bold.

Feature	Ranks 0-1	Ranks 1-2	Ranks 0-2
Aptitude	**0.035**	0.422	**0.003**
Attention	0.948	0.209	0.174
Pleasantness	**0.028**	0.803	**0.014**
Sensitivity	0.956	0.720	0.761
Relatability score	**0.017**	0.354	**0.001**
Imperative score	**0.042**	0.940	**0.044**
Advice score	0.234	0.560	0.542
ASD	0.113	0.157	0.901
ASH	0.112	0.162	0.884
ASHD	0.228	0.163	0.837
TOC	0.970	0.178	0.193
CNE	1.000	0.564	0.414
CPN	0.378	0.147	0.579
LEN	**0.054**	0.111	0.682

I conducted additional experiments using solely word2vec features to see how much impact the 14 advice features had on the algorithm. To do this, I slightly adjusted the network architecture to accommodate the new input shape, which was (m, 1, 1, 300) instead of (m, 1, 1, 342). As a result, the *Conv1* layer had 114 filters of shape (1, 100) instead of (1, 114). After that point, the input/output dimensions and further layers remained the same as in the network model for all features. The hyperparameters also stayed the same. Results are presented in Table 32.

Table 32: Accuracy and loss for all folds on the training and test sets for models trained only on word2vec features.

Fold	Training loss	Training accuracy	Test loss	Test accuracy
1	0.263	0.888	0.403	0.866
2	0.347	0.866	0.375	0.840
3	0.150	0.945	0.181	0.929
4	0.353	0.849	0.439	0.828
5	0.184	0.919	0.250	0.898
6	0.190	0.943	0.211	0.932
7	0.212	0.923	0.282	0.901
8	0.207	0.920	0.245	0.903
9	0.376	0.880	0.390	0.849
10	0.248	0.923	0.287	0.899
Average	0.253	0.906	0.306	0.885

6.4.3 Observations during training

For a research problem of ranking comments within a thematically consistent group, it was important that each of the three comment texts went through the same initial calculations. This could have been achieved by using a recurrent neural network (Rumelhart, Smolensky, McClelland and Hinton, 1986). In RNNs, each timestep – in this case, comment text – is processed by the same unit (for example GRU (Cho et al., 2014) or LSTM (Hochreiter and Schmidthuber, 1997)) that has its parameters adjusted during the training process. However, my attempts at using an RNN were unsuccessful. Although the algorithm trained well (training set accuracy was usually above 0.95), these results did not generalize to the test set. Test accuracy was always around 0.33, which in this setting is random chance level. One reason for this may be that RNNs are particularly sensitive to the order of the timesteps and even shuffling the comments did not help in alleviating this issue. The network kept overfitting to the training set over the course of many epochs, but then was not able to make correct predictions on the previously unseen data of the test set. With the way RNNs work, the prediction for each comment relied heavily on calculations made for the previous one(s) instead of the network looking at the triple as a group and not as a sequence. Using a convolutional network solved this problem and allowed the model to achieve satisfactory results on the test set as well.

However, even the convolutional network displayed some interesting behavior.

First, the network worked only with very specific settings, namely with the Adadelta optimization algorithm and the tanh activation function. Searching for the best optimization method and activation function is routinely performed in deep learning to yield the best results for the given task (for example, see Section 5.4.3 of this thesis). Therefore, I also performed hyperparameter search, but the differences in accuracy between various options were unusually large in this case. Despite repeated training, the algorithm did not converge with any other optimization algorithm than Adadelta. This was probably related to the issue of local optima. Other optimization algorithms such as Adam or RMSProp, which have all been tried, usually gave good performance very soon into the training, but after around 500 epochs tended to get stuck indefinitely. Adadelta, with its dynamic adaptation of the learning rate, would also get stuck, but only temporarily, and was able to overcome this problem. Additionally, the ReLU activation function, which I initially tried instead of tanh, caused the network to get stuck in a local minimum at a high error level.

Secondly, around epoch 700-800 the error rate would briefly rise and then fall down again to an even lower level. This tendency can be observed in Figure 14.

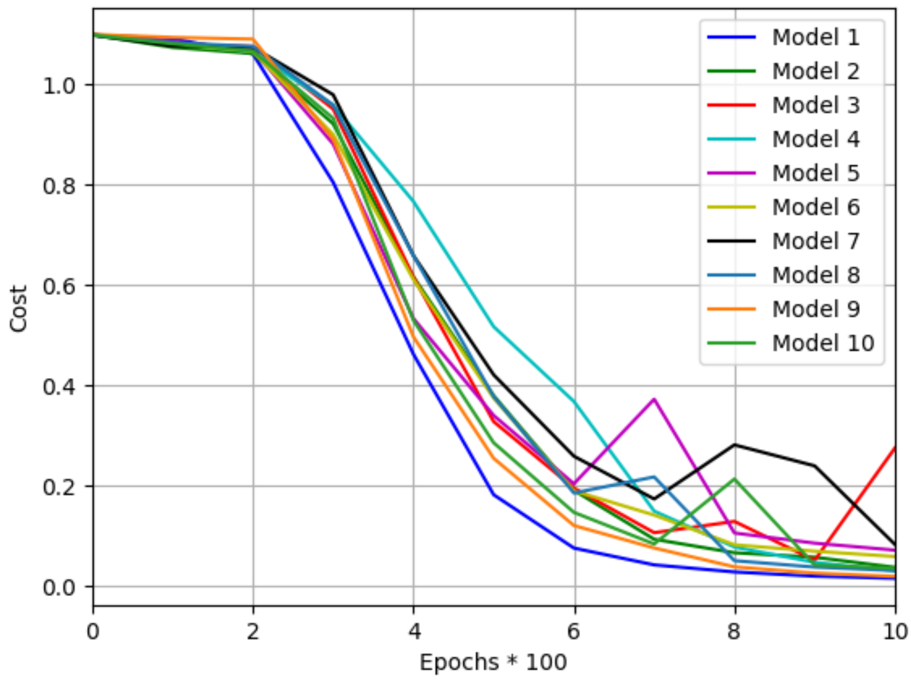

Figure 14: Overview of cost progression across 1000 training epochs for all folds. Each fold is marked with a different color.

Such a temporary drop in performance is usually caused by a learning rate that is too large for the given stage of training. It can be assumed that after Adadelta adjusted the learning rate, the network performance was able to rise up again in the last epochs. Perhaps this is the reason why this particular optimization algorithm worked best in this study. Other algorithms like Adam (Kingma and Ba, 2014) or SGD (Robbins and Monro, 1951) kept getting stuck and were unable to overcome this issue.

6.4.4 Error analysis

As is evident from results presented in Table 29, the network was able to achieve high accuracy on the ranking task. Perhaps this was caused by the relatively large amount of data; at over 22,000 examples, the model had more than enough data to learn how to rank the comment texts.

I performed error analysis on all the folds. First, I prepared confusion matrices to see which ranks were most commonly confused with each other. I found no clear tendency that would be present across all folds. However, I also calculated mean numbers of misranked comments for all confusion matrices, which is presented in Table 33. On average, there were 189.8 comments misranked in each fold.

Table 33: Mean values for all 10 confusion matrices. Values that stand out are bolded.

	Predicted label 0	Predicted label 1	Predicted label 2
True label 0	2153.7	21.1	**39.1**
True label 1	22.0	2143.9	**48.1**
True label 2	31.7	27.8	2154.5

As evident from the table, most differences in numbers of misranked comments do not seem to be large. However, the numbers are higher for comments that were misranked as 2 despite their true label being 0 or 1. In other words, for comments with true label 0 there were more comments misranked as 2 than those misranked as 1, and likewise, for comments with true label 1 there were more misranked as 2 than as 0. Moreover, the numbers were also higher for misranked comments with true label 2 than those of other true labels. While these results are not conclusive in any way, it shows an interesting quality about the algorithm: that it may have a slight tendency to rank comments lower than it should. It is also possible that specifically comments of rank 2 were most misleading. This could be caused by the fact that on Reddit usually only the top comment stands out and there might not have been that much difference between comments of rank 1 and rank 2. This is supported by the findings from Table 31, where there was no statistical significance between feature values with these two ranks, while rank 0 stands out.

It must be noted here that the algorithm performed the final ranking by taking *argmax* of a 3x3 matrix with one-hot vector rows, so the rankings were not interchangeable, but independent at this point. In other words, a misranked comment in the triple did not translate to another comment being misranked by exchanging their mutual rankings. This means that even if a triple contained a misranked comment, other comments in that triple could be (and usually were) ranked correctly.

Some errors might have been caused by the fact that the comments were ranked by humans. Natural language expressions have many nuances and while my algorithm was able to find tendencies and capitalize on them, it cannot be expected to achieve 100% accuracy all the time. Sometimes human judgement is difficult to explain in terms of numerical feature values, which is why the algorithm was misled by the true labels. However, these cases are outliers; as seen in Table 33, the overwhelming majority of comments were ranked correctly.

To see whether the above hypothesis was true, I performed a quality analysis of the content of misranked comments. In this setup, each triple of comments was present in the dataset six times, so there was a possibility of each comment to be present multiple times in the test set. It turned out that comments that were misranked once tended to be misranked again if they reappeared in the test set. This suggests that some single comments may be a bit troublesome for the algorithm, although the percentage of such comments in the overall dataset is negligible – usually 4-5 comments would repeat as misranked in the test set in each fold. Moreover, I found no clear characteristics of such misranked comments compared to those that were ranked correctly. There was no apparent difference in the manner of expression or the content itself. Such findings suggest that the rankings of these comments did not follow any particular pattern – a pattern which the algorithm was searching for. However, as mentioned above, this occurred very rarely and is not surprising when one considers the implications of working with natural language data.

Other misranked comments were of the *[deleted]* kind. Comments with this kind of content were deleted by the user themselves after some reflection. Such comments are usually inappropriate or rude and would not receive a lot of points, which means they would usually rank the lowest in any given triple in the dataset. However, it is possible that some such comments had contained content that was upvoted by people agreeing with the rude or inappropriate message before the removal. Then, at the time of my downloading the data, these comments had a lingering high score despite being already deleted or removed. It is also possible that some user shared their advice in the comment and that advice was good enough to get a lot of upvotes, but then was deleted from discussion by the user themselves because they decided it revealed too much about them after all. This is an occasional occurrence not only in the advice subreddits, but also in subreddits concerning other personal issues, for example mental health. Whatever the cause, the *[deleted]* comments were misleading to the algorithm, as the features were calculated not from the original content of the comment (which was no longer available), but from the single word *deleted*. As a result, the features were not informative enough to perform the ranking correctly. However, the *[deleted]* comments were only a small fraction of all misranked comments from the test set. They most likely did not hinder the training process either, as neural networks are rather robust against occasional noise in data.

As can be seen in Tables 30 and 31, the differences in feature values between ranks were relatively small. For many features, like Advice Score or Specificity Scores, those differences were not statistically significant. This suggests that perhaps the network would not be able to rank advice texts based on these features alone and that it benefitted from the word2vec features as well. This is why I conducted additional experiments using only word2vec word representations. However, as seen in Table 32, the average test accuracy was only 0.885 as compared to 0.971 from Table 29. This proves that even though word2vec features were important in this study, the 14 advice features also played a significant role in achieving good accuracy in the experiments.

6.5 Discussion about features

The experiments allowed me to identify the most important features in my study: aptitude, pleasantness, Relatability Score and Imperative Score. As seen in Table 31, the differences in their values between ranks were statistically significant.

The Sentics are associated with dichotomies such as ecstasy-grief for pleasantness and admiration-loathing for aptitude (pictured in Figure 5 in Section 5.2.1 of the thesis). My findings suggest that these emotions may be more important in advice texts than others like vigilance-amazement associated with attention or rage-terror associated with sensitivity. A lot of posts in the dataset – mainly those that came from *r/getdisciplined* – described the author's dissatisfaction with their current life and desire to change and be happier. Because of this, emotions such as fear, anger or anticipation may have been less present in the advice comments, while talk about sadness, trust or joy seemed to be more prominent, especially when it came to motivational advice.

The statistical significance of Imperative Score and Relatability Score is noteworthy as well. I designed these features to reflect the fact that best-rated advice comments tended to contain a lot of imperative expressions and were authored by people who related to the given problem. On the other hand, lack of significant differences between ranks in Advice Score is not surprising; all comments contained some form of advice, so obviously advice expressions were present in all of them. It seems that, rather than the presence of advice expressions, the manner of giving advice was more significant; specifically, how many first-person pronouns and imperative phrases there were included in the text. As can be seen in Table 30, best ranked comments had the highest Imperative Score and lowest rated comments had the highest Relatability Score. This suggests that while it is important to use imperative expressions when giving advice and to relate to the given problem, too much self-talk deducts from that advice's quality.

Finally, the data from Table 30 sheds new light on previous findings concerning the features. Error analysis of experiments conducted in the previous chapter revealed differences in feature values between texts containing advice and regular texts (Section 5.3.4, Table 8). Considering both the mean and median values, all advice features had perceptibly higher values for advice texts than for regular texts. I assumed that similarly, the values for advice features would be higher for good quality advice compared to lower quality advice. However, the difference in feature values between rank 0 and rank 2 is rather small (albeit statistically significant for some features). Interestingly, some features are highest for the middle rank 1, for example sensitivity, Advice Score or ASHD. All these findings suggest that the relationship between my features and advice quality is complicated and not readily visible, which is in line with my findings about the lack of linear correlations between any given feature and advice rank.

6.6 Conclusions

This chapter presented a convolutional neural network able to rank Reddit comments containing advice, based on advice quality represented by its score assigned by other online users. I was also able to identify specific measurable qualities of a good advisory text, such as scoring high on aptitude, pleasantness and Imperative Score while maintaining Relatability Score on a lower level.

It must be noted here that the task was performed on text triples, which means that the results are valid only in a very specific setting. Ideally, I would like to have a network able to rank the quality of advice contained in any given single text, without the two reference texts, which may not always be available. This network, however, was not trained to recognize objective advice quality, but rather to select the best advice text from a given triple regardless of the overall quality level in that triple. Therefore, right now it cannot be used to judge how good a piece of advice is without any comparison. Constructing a network capable of accomplishing this task based on my current findings is a topic for future studies.

Likewise, I assumed that all advice comments were on topic, because they were downloaded from their respective threads as responses to another user's post, but this may not always be the case with raw data obtained in a different manner. Therefore, further experiments will involve determining whether the advice is thematically appropriate for the given problem before performing the ranking.

On the other hand, while this method cannot be used to determine objective quality of a piece of advice, it is helpful for selecting the best advice in a given dataset. This can be useful in creating output utterances for the motivational dialogue system, for example by choosing the best advice from three candidate outputs and presenting that advice to the user as the final answer.

This chapter concludes the portion of research done during my PhD studies at Hokkaido University. The next chapter is a summary of my findings and presents potential future directions for this research project.

Chapter 7: Conclusions of the thesis

This chapter summarizes my research described in the thesis (Section 7.1) and discusses future work on the project (Section 7.2).

7.1 Summary of my work

In this thesis, I have described my efforts towards creating a motivational dialogue system. After stating the final goal of the research, I then described an electronic assistant that can be regarded as the first attempt to fulfil that goal. Analysis of user evaluation results allowed me to shift my focus to the basics which are crucial to a motivational dialogue system, namely creating a corpus of motivational advice. I then proposed algorithms for classification and ranking of advice texts. These algorithms will play a key role in gathering data for further stages of this research and using it effectively.

Both the classifier and the ranking network achieved high results. Precision in the classification experiments was 0.965-0.977 (as seen in Tables 22 and 23) and accuracy for the ranking experiment was 0.971 (as seen in Table 29). This shows that both these algorithms are effective at accomplishing the tasks that they were created for. Even though there is room for improvement on both (how to take care of the data loss in the classification task? How to modify the ranking algorithm so that it does not need reference texts?), they can already be useful for creating the corpus of motivational advice.

My research also sheds light on the previously untackled topic of qualities of motivational and advisory texts. During the course of my studies, I have identified 14 advice features that characterize such texts. I have shown how they are relevant to both the classification and the ranking task. With classification, all features were visibly different in values for the advisory/motivational texts and the regular texts. With ranking, there were a few features that showed the most importance, namely Sentic Scores of aptitude and pleasantness, Relatability Score and Imperative Score. These findings can be used in other studies on advisory content as well.

However, not only the 14 advice features were used in my research. Word embeddings created with the word2vec algorithm also proved to be very useful. Therefore, in the future stages of this research, both these feature sets need to be taken into consideration.

Finally, the datasets that I created for the purposes of my studies can be utilized in other research projects on advisory and motivational texts.

7.2 Future work

The obvious follow-up to the studies presented in this thesis will be the creation of a motivational advice corpus by using both the classification and the ranking algorithms. Perhaps this can be done in the form of creating a database. Specifically, for each user problem, there could be a few advice pieces present in the corpus. Then those pieces can be ranked using the ranking algorithm or ordered in any other manner, and then served to the user as a solution to their problem. This task may seem daunting, as advice can be given on a wide variety of different topics. However, specifically motivational advice is usually topically limited and has only a few components besides actual advice, such as encouraging statements or expressions of compassion. Therefore, creating a database of motivational advice seems rather feasible.

It should be mentioned here that different users will be motivated in different ways. This has become clear in studies such as those described in Section 2.2, as well as in my own studies described in Chapter 3 of this thesis. Ideally, a motivating dialogue system would keep track of the user's level of motivation as well as various motivational techniques that are effective for that particular user. This means that such a system needs a sophisticated user model. Creating this model would be challenging, but would certainly improve user experience and the effectiveness of motivational advice given by the system.

There is also the issue of generating motivational advice. Potentially, this generation could be done using a standard language model trained on the motivational corpus mentioned above. It would be useful to capitalize on the advice features during generation, since it is desirable that the generated text resembles human-crafted advice texts as closely as possible. Perhaps it would be good to somehow incorporate the 14 advice features into the Beam Search algorithm that runs during the generation process. Alternately, the 14 features could be utilized in a reinforcement learning manner for advice generation. However, should generation prove to be inadequate – as is the case in many language generation systems – then appropriate advice could be retrieved from the motivational corpus, and the dialogue system's task would be to choose that advice and then slightly alter its content or manner of expression, so that the advice suits the user model as closely as possible.

As mentioned in Section 7.1, the algorithms that I proposed in this thesis could be improved. With the classification algorithm, there is the issue of data loss at around a

15% level. Since with this algorithm data can be downloaded and sorted from various sources, it is possible to make up for the loss by simply obtaining more data. However, improving both networks' performance so that data loss is minimal is an important future step for this research. Likewise, the ranking algorithm could be developed further so that it rates the motivational or advisory power of each piece of advice individually, without having to resort to reference texts. On top of that, both algorithms could be explored in experiments in different fields to gain more insight into their strengths and weaknesses and the ways they could be improved.

Finally, as discussed in the previous section, word embeddings played a crucial role in my experiments. I only used 100-dimensional word2vec vectors. However, future studies should include exploring the possibility of raising accuracy and F-scores of my algorithms by switching word2vec embeddings with other, more powerful distributed representations of words, such as those obtained from BERT (Devlin, Chang, Lee and Toutanova, 2018) or ELMo (Peters et al., 2018).

Appendix 1: User survey for evaluation of *Asystent*

This appendix contains the user survey that was distributed to the users for evaluation purposes. It has been translated from its original Polish version.

EVALUATION OF *ASYSTENT*

Dear User! Thank you for using the dialogue system *Asystent.*

Please fill out a short survey about your cooperation with the program.

Age:
Gender: F/M

Have you ever used the Python programming language?
(it is the language that *Asystent* was written in)

………………………………………

On a scale from 1 (lowest) to 5 (highest) evaluate *Asystent* on the following criteria:

1. Naturalness of interaction.

Did the conversation seem like you were talking to a real human instead of a machine?
Did you find the program's utterances to sound natural?
Did the program react to your utterances like a human would?

Naturalness of interaction: 1 2 3 4 5

2. Helpfulness.

Do you think that the program helped you in doing your tasks?
Did the program have functions that made your work easier?
Did the program avoid disturbing you during work?

Helpfulness: 1 2 3 4 5

3. Flexibility of dialogue.

Did the conversation with the program seem to not be artificial?
Did you have the impression that you did not have to answer questions in a strictly prescribed manner?
Did the program respond appropriately to your utterances?

Flexibility of dialogue: 1 2 3 4 5

4. Friendliness.

Was the cooperation with the program pleasant to you?
Do you think that the program tried to assist you as best as it could?
Do you think that the conversation had a nice atmosphere?

Friendliness: 1 2 3 4 5

5. Ability to motivate you to work.

Did the program sufficiently motivate you to complete your tasks?
Do you think that you completed more task while working with the program compared to working alone?
Was the motivation that you got from the program effective?

Ability to motivate you to work: 1 2 3 4 5

6. Quality of the cooperation.

Did you enjoy working with *Asystent*?
Was the cooperation smooth?
Do you think that the program is a good assistant?

Quality of the cooperation: 1 2 3 4 5

Have you had any issues when working with *Asystent*? What was the problem?

……………………………………

Would you like to work with *Asystent* again? Y/N
Why/why not?

……………………………………

Please write your additional comments and opinions:

……………………………………

Appendix 2: Example conversation with *Asystent*

This appendix contains a transcript of an example conversation with *Asystent*. It was translated from its original Polish language. The transcript includes italicized comments in square brackets that indicate the actions of the program such as opening files. To save space, some new line breaks were omitted.

First conversation

Program: Hello, dear User! What is your name?
User: Patrycja
Program: Nice to meet you, Patrycja! My name is Asystent and I'm here to help you plan your work. I hope we'll make a good team! Have you read the manual yet?
User: No
Program: In that case, please read it before we begin. *[the manual opens for the user to read through]* Type "ok" when you're finished reading.
User: ok
Program: Now you know how to work with me and we can begin. I'll ask you now to fill out the task plan. Each task needs to be input in one line as follows:

task_name starting_time time_for_task_in_minutes
task_name2 starting_time time_for_task2_in_minutes

and so on.

Example:

writing my thesis 12:00 120
shopping 14:15 25

Please remember that the manual is always available to you in my directory. When you're finished, please save the text file, close it and type "ready". *[a text file opens for the user to fill out]*
User: ready
Program: Perfect! Let's move on. By the way – I'm sorry that you need to fill out some files for me, but cooperation is cooperation, right? ;) Please wait a minute for me to think the plan through. *[the program checks the work plan]*
Move the task of "go to the post office" to a later time, because otherwise you won't be able to finish the task of "washing the dishes". Type "ok" when you're finished. *[the file opens again for modifications]*
User: ok
Program: *[the program checks the work plan]* I'm happy to say that everything is ok now! This is your work plan:

12:00 writing the BA thesis
12:55 washing the dishes
13:10 go to the post office

14:00 feed the cats
14:10 call a friend

To check your work plan at any moment, please open the file *work_plan.txt* in my directory. If you want to make any changes now, please type in “change” and I’ll wait for you to get finished with filling out the task list.
User: change
Program: I understand. Please make the changes and tell me when you’re done. Don’t forget to save the file! *[the file opens for changes]*
Are you done yet?
User: done
Program: Please give me a minute to check against any overlaps. *[the program checks the work plan]* Modifications were successful :)

This is your work plan:

12:00 writing the BA thesis
12:55 washing the dishes
13:12 go to the post office
14:03 feed the cats
14:15 call a friend

Now please tell me how I can reward you for completing your tasks. If you finish a task, as a reward I will open for you a website or a file of your choice so that you can relax before the next task. Please fill out the file according to the template below:

short_name URL_address
or
short_name path_to_file

For example:

fejs www.facebook.pl
mythesis c:/Documents/myfiles/thesis.doc

[a text file opens for the user to fill out] Shall I open the manual for you, so you know how to do this?
User: yes
Program: Here you go! *[the manual opens for the user to read through]* Type “ready” when you’re finished.
User: ok
Program: Thank you for filling out the files! :) Since everything is clear now, you can turn the shell off. See you! Goodbye!
[the program exits]

Second conversation

Program: Hello again, Patrycja! I hope you slept well and are ready to work!
Today is Tuesday, August 25th, 12:09:34. Let's begin!
As you know, I will assist you in your work and monitor whether you complete your tasks. I will write down which tasks were completed and which ones were not. Both task lists – saved in files *completed.txt* and *not_completed.txt*, respectively – are always available to you in my directory. At the end of the day I will make a short summary of your work. I hope you'll complete 100% of your tasks! :)
Once again, I'll show you your work plan:

12:00 writing the BA thesis
12:55 washing the dishes
13:12 go to the post office
14:03 feed the cats
14:15 call a friend

This is your last chance to make changes. Do you want to do it?
User: yes
Program: No problem :) Please make the changes and get back to me when you're done. Remember to save the file! *[the file opens for changes]*
Are you done?
User: done
Program: I'll check whether everything is ok. *[the program checks the work plan]* Modifications were successful! This is your finished plan:

12:20 writing the BA thesis
12:55 washing the dishes
13:12 go to the post office
14:03 feed the cats
14:15 call a friend

Do you want me to remind you about upcoming tasks?
User: yes
Program: In that case, please write how many minutes before a task you want to be reminded.
User: 5
Program: Thank you, I'll remember that :) I can also remind you about a task coming to an end. Do you want to turn this function on?
User: yes
Program: Great, in that case, please write how many minutes before the end of the task I should speak up :)
User: 3
Program: Thank you, I wrote that down :) A reminder sounds like this. *[the sound is played]*
Your first task starts at 12:20. I'll remind you about it when the time comes.
You have 5 minutes until your first task!
Your first task is writing the BA thesis. You have 30 minutes for this task. Ready, steady, go!
You have 3 minutes left :)

How did it go? Did you finish?
User: yes
Program: That's great! I'm putting this task on the list of completed ones :)
If you have a moment, you can get a reward :) If you don't want to open anything, type "no".
User: I want
Program: Tell me what to open for you :)
User: fejs
Program: Here you go! *[Facebook opens in a browser]*
The next task, washing the dishes, starts at 12:55.
You start your next task in 5 minutes.
Your next task is washing the dishes. You have 15 minutes for this task. Start!
Only 3 minutes left!
Did you finish the task?
User: yes
Program: Congratulations! :) Do you want to open a reward? If you don't want to open anything, type "no".
User: no
Program: That's fine, good luck with your other tasks :)
I'd like to remind you that your next task is go to the post office and it starts at 13:12.
You're starting the next task in 2 minutes.
Your next task is go to the post office. You have 35 minutes for this task. Start!
Only 3 minutes left!
Was the task completed?
User: no
Program: I understand. I'll put it on the list of unfinished tasks.
The next task, feed the cats, starts as 14:03.
The next task starts in 5 minutes.
Next on your list is "feed the cats". You planned 5 minutes for this task. Make it in time! :)
You have 3 minutes left :)
Did you finish the task?
User: yes
Program: Congratulations! I'm putting it on the list of finished tasks and I wish you all the best with the next one :)
If you have a moment, you can get a reward :) If you don't want to open anything, type "no".
User: I want
Program: Which file or website should I open?
User: fejs
Program: Here you go! *[Facebook opens in a browser]*
I'd like to remind you that your next task is call a friend and it starts at 14:15.
You have 5 minutes until your next task!
Your last task is call a friend. You have 10 minutes to do it. Good luck!
You have 3 minutes left until the task should be finished.
How did it go? Did you finish?
User: yes
Program: Congratulations! This was your last task for today!
Dear User! I will now summarize your work.
Here are the tasks that you were able to complete:

writing the BA thesis
washing the dishes
feed the cats
call a friend

Congratulations!
Here are the tasks that were not completed:

go to the post office

But don’t worry, I’m sure you’ll do better next time :)
If you need this information, please refer to my directory where I saved both task lists. It will help you plan better in the future!
This is the end of our cooperation. Did you enjoy working with me?
User: yes
Program: That’s so great to hear! I also had a lot of fun :)
Anyway, thank you for using me. I hope we can meet again in the future :)
It was nice to meet you, Patrycja! Goodbye!
[the program exits]

Appendix 3: Data sources

This appendix contains information about data sources that were used in the research presented in this thesis. The table gives the name of each source along with the relevant link. The top row is the link to Reddit itself. After that, all the subreddit names are given in alphabetical order. The final row contains the link to Wikipedia dump that was used for training the word2vec and doc2vec models.

Table 34: List of datasets.

Name	**Link**
Reddit	https://www.reddit.com
r/Advice	https://www.reddit.com/r/Advice/
r/getdisciplined	https://www.reddit.com/r/getdisciplined/
r/legaladvice	https://www.reddit.com/r/legaladvice/
r/pics	https://www.reddit.com/r/pics/
r/relationship_advice	https://www.reddit.com/r/relationship_advice/
r/todayilearned	https://www.reddit.com/r/todayilearned/
Wikipedia dump	https://dumps.wikimedia.org/backup-index.html

List of Tables

List of Figures

References

Agnatis, K., Alexiadis, A., and Refandis, I. (2016). Intelligent calendar applications: A holistic framework based on ontologies. *14*(2), 1-22.

Allen, K., Bagroy, S., Davis, A., and Krishnamurti, T. (2019). ConvSent at CLPsych 2019 Task A: Using post-level sentiment features for suicide risk prediction on Reddit. *Proceedings of the Sixth Workshop on Computational Linguistics and Clinical Psychology* (pp. 182-187). Minneapolis, Minnesota: Association for Computational Linguistics.

Amazon. (2019). *Alexa*. Retrieved September 2019, from https://www.alexa.com/

Apple. (2019). *Siri*. Retrieved September 2019, from https://www.apple.com/ios/siri/

Arora, S., Liang, Y., and Ma, T. (2017, April). *A simple but tough-to-beat baseline for sentence embeddings.* Retrieved from https://openreview.net/pdf?id=SyK00v5xx

Badubi, R. M. (2017). Theories of motivation and their application in organizations: A risk analysis. *International Journal of Innovation and Economic Development, 3*(3), 45-30.

Bank, J., Ariely, D., and Shoham, Y. (2012). *Timeful*. Retrieved September 2015, from www.support.timeful.com

Ben-Hur, A., Horn, D., Siegelmann, H., and Vapnik, V. N. (2001). Support vector clustering. *Journal of Machine Learning Research, 2*, 125-137.

Bickmore, T. W., and Picard, R. W. (2005). Establishing and maintaining long-term human-computer relationships. *ACM Transactions on Comuter-Human Interaction (TOCHI), 12*(2), 239-327.

Bird, S., Klein, E., and Loper, E. (2019). *Natural Language Toolkit [Software]*. Retrieved from https://www.nltk.org/

Bisht, J. (2016, September 6). *United States Patent No. 9,436,934.*

Business Insider. (2016). *80% of businesses want chatbots by 2020*. Retrieved May 2017, from http://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12/

Callejas, Z., and Griol, D. (2016). An affective utility model of user motivation for counselling dialogue systems. *International Workshop on Future and Emerging Trends in Language Technology* (pp. 86-97). Cham: Springer.

Callejas, Z., Griol, D., and Lopez-Cozar, R. (2011). Predicting user mental states in spoken dialogue systems. *EURASIP Journal on Advances in Signal Processing, 2011*(1), 6.

Cambria, E., Livingstone, A., and Hussain, A. (2012). The hourglass of emotions. *Cognitive behavioural systems. Lecture notes in Computer Science. 7403*, pp. 144-157. Berlin, Heidelberg: Springer.

Cambria, E., Poria, S., Hazarika, D., and Kwok, K. (2018). SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. *Proceedings of 32nd AAAI Conference on Artificial Intelligence*, (pp. 1975-1802).

Cho, S.-B., and Kim, J. H. (1995). Combining multiple neural networks by fuzzy integral for robust classification. *IEEE Transactions on Systems, Man, and Cybernetics, 25*(2), 380-384.

Coleman, A. M. (2008). *A dictionary of psychology* (3rd ed.). Oxford University Press.

DARPA. (2019). *CALO. Cognitive assistant that learns and organizes*. Retrieved September 2019, from http://www.ai.sri.com/project/CALO

Deshpande, S., Palshikar, G. K., and Athiappan, G. (2010). An unsupervised approach to sentence classification. *International Conference on Management of Data COMAD*, (p. 88).

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint:1810.04805*.

Doist Team. (2019). *Todoist Karma: A powerful new way to track productivity*. Retrieved September 2019, from https://en.todoist.com/karma

Elmelid, A., Stickley, A., Lindblad, F., Schwab-Stone, M., Henrich, C. C., and Ruchkin, V. (2015). Depressive symptoms, anxiety and academic motivation in youth: Do schools and families make a difference? *Journal of Adolescence, 45*, 174-182.

Eschler, J., Kendall, L., O'Leary, K., Vizer, L. M., Lozano, P., McClure, J. B., Pratt, W., and Ralston, J. D. (2015). Shared calendars for home health management. *Proceedings of Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing* (pp. 1277-1288). ACM.

Fallaize, R., Franco, R. Z., Hwang, F., and Lovegrove, J. A. (2019). Evaluation of the eNutri automated personalised nutrition advice by users and nutrition professionals in the UK. *PloS one, 14*(4), e0214931.

Fang, C., Mu, D., Deng, Z., and Wu, Z. (2017). Word-sentence co-ranking for automatic extractive text summarization. *Expert Systems with Applications, 72*, 189-195.

Fogg, B. J. (2009). A behavior model for persuasive design. *Proceedings of the 4th International Conference on Persuasive Technology* (p. 40). ACM.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics, 36*(4), 193-202.

Fussner, L. M., Mancini, K. J., and Luebbe, A. M. (2018). Depression and approach motivation: Differential relations to monetary, social, and food reward. *Journal of Psychopathology and Behavioral Assesment, 40*(1), 117-129.

Gerhart, B., and Fang, M. (2015). Pay, intrinsic motivation, extrinsic motivation, performance, and creativity in the workplace: Revisiting long-held beliefs. *Annual Review of Organizational Psychology and Organizational Behavior, 2*, 489-521.

Gkioxari, G., Toshev, A., and Jaitly, N. (2016). Chained predictions using convolutional neural networks. *European Conference on Computer Vision* (pp. 728-743). Cham: Springer.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2672-2680.

Google. (2019). *Google Assistant*. Retrieved October 2019, from https://assistant.google.com/

Greenleaf, G., Mowbray, A., and Chung, P. (2018). Building sustainable free legal advisory systems: Experiences from the history of AI and law. *Computer Law and Security Review, 34*(2), 314-326.

Grigore, E. C., Pereira, A., and Scassellati, B. (2015). Modeling motivational states in adaptive robot companions. *AAAI Fall Symposium Series* (pp. 77-80). AAAI.

Gustafson, P. K. (2013, February 5). *United States Patent No. 8,369,493.*

Habitica Team. (2019). *Habitica: Gamify your life*. Retrieved September 2019, from https://habitica.com/static/home

He, H. A., Greenberg, S., and Huang, E. M. (2010). One size does not fit all: Applying the transtheoretical model to energy feedback technology design. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 927-936). ACM.

HealthDirect. (2018). *Motivation: How to get started and staying motivated*. Retrieved October 23, 2018, from https://www.healthdirect.gov.au/motivation-how-to-get-started-and-staying-motivated

Hershenberg, R. (2017). *Activating happiness: A jump-start guide to overcoming low motivation, depression, or just feeling stuck.* New Harbinger Publications.

Higashinaka, R., Imamura, K., Meguro, T., Miyazaki, C., Kobayashi, N., Sugiyama, H., Hirano, T., Makino, T., and Matsuo, Y. (2014). Towards an open-domain conversational system fully based on natural language processing. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Techincal Papers* (pp. 928-939). Dublin: Association for Computational Linguistics.

Hochreiter, S., and Schmidthuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

Hurst, E. H., and Morris, A. A. (2016, April 26). *United States Patent No. D754,692.*

Iliff, E. C. (2015). *United States Patent No. 9,005,119.*

Jabri, S., Dahbi, A., Gadi, T., and Bassir, A. (2018). Ranking of text documents using tf-idf weighting and association rules mining. *Proceedings of 4th International Conference on Optimization and Applications (ICOA)* (pp. 1-6). IEEE.

Jeatrakul, P., Wong, K. W., and Fung, C. C. (2010). Data cleaning for classification using misclassification analysis. *Journal of Advanced Computational Intelligence and Intelligent Informatics, 14*(3), 297-302.

Jerebko, A. K., Malley, J. D., Franaszek, M., and Summers, R. M. (2003). Multiple neural network classification scheme for detection of colonic polyps in CT colonography data sets. *Academic Radiology, 10*(2), 156-160.

Kaptein, M., de Ruyter, B., Markopoulos, P., and Aarts, E. (2012). Adaptive persuasive systems: A study of tailored persuasive text messages to reduce snacking. *ACM Transactions on Interactive Intelligent Systems (TiiS), 2*(2), 10.

Kaptein, M., Markopoulos, P., de Ruyter, B., and Aarts, E. (2009). Can you be persuaded? Individual differences in susceptibility to persuasion. *IFIP Conference on Human-Computer Interaction* (pp. 115-118). Berlin, Heidelberg: Springer.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751). Doha, Qatar: Association for Computational Linguistics.

Kingma, D. P., and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint: 1412.6980*.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM), 46*(5), 604-632.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. *ICML deep learning workshop*, *2*.

Kultalahti, S., and Liisa Viitala, R. (2014). Sufficient challenges and a weekend ahead–Generation Y describing motivation at work. *Journal of Organizational Change Management, 27*(4), 569-582.

Le, Q., and Mikolov, T. (2014). Distributed representations of sentences and documents. *International Conference on Machine Learning*, (pp. 1188-1196).

Litalien, D., Guay, F., and Morin, A. J. (2015). Motivation for PhD studies: Scale development and validation. *Learning and Individual Differences, 41*, 1-13.

Liu, B., and Lane, I. (2016). Joint online spoken language understanding and language modelling with recurrent neural networks. *arXiv preprint: 1609.01462*.

Liu, B., Tur, G., Hakkani-Tur, D., Shah, P., and Heck, L. (2018). Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. *arXiv preprint: 1804.06512*.

Loria, S. (2018). *TextBlob: Simplified text processing [Software]*. Retrieved from https://textblob.readthedocs.io/en/dev/index.html

Malhotra, D., Malhotra, M., and Rishi, O. P. (2018). An innovative approach of web page ranking using Hadoop and MapReduce-based cloud framework. *Proceedings of Big Data Analytics* (pp. 421-427). Springer.

MammothB. (2018). *Symspellpy [Software]*. Retrieved from https://github.com/mammothb/symspellpy

Meisels, A. C., Coughlin, S. J., and Renner, W. K. (2014, April 29). *United States Patent No. 8,712,810.*

Microsoft. (2019). *Cortana - Meet your personal assistant*. Retrieved September 2019, from https://www.microsoft.com/en-us/cortana

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 3111-3119.

Myangah, T. M., and Rezai, M. J. (2016). Persian text ranking using lexical richness indicators. *Glottometrics, 35*, 6-15.

Nealrs. (2014). *Contractions.py [Software]*. Retrieved from https://gist.github.com/nealrs/96342d8231b75cf4bb82

Nguyen, V., Kim, O. T., Dang, T. N., Moon, S. I., and Hong, C. S. (2016). An efficient and reliable Green Light Optimal Speed Advisory system for autonomous cars. *Proceesings of 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)* (pp. 1-4). IEEE.

Nussbaum, R. (2017). *15 ways to motivate yourself that actually work*. Retrieved October 23, 2018, from Greatist: https://greatist.com/grow/motivation-tips-that-work

Oinas-Kukkonen, H. (2013). A foundation for the study of behavior change support systems. *Personal and Ubiquitous Computing, 17*(6), 1223-1235.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web.* Stanford InfoLab.

Park, A., Conway, M., and Chen, A. T. (2018). Examining thematic similarity, difference, and membership in three online mental health communities from Reddit: A text mining and visualization approach. *Computers in Human Behavior, 78*, 98-112.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 1*, pp. 2227–2237. Association for Computational Linguistics.

Princeton University. (2010). *About Wordnet*. Retrieved from https://wordnet.princeton.edu

Reddit. (2019). *API Documentation [Software]*. Retrieved from https://www.reddit.com/dev/api/

Refandis, I., and Alexiadis, A. (2011). Deployment and evalyation of SelfPlanner, an automated individual task management system. *Computational Intelligence, 27*(1), 41-59.

Robbins, H., and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics, 1951*, 400-407.

Root, S. A., and Root, M. R. (2015). *United States Patent No. 9,204,252.*

Rumelhart, D., Smolensky, P., McClelland, J., and Hinton, G. (1986). Schemata and sequential thought processes in PDP models. In D. Rumelhart, J. McClelland, and C. P. Group, *Parallel distributed processing: Explorations in the microstructure* *of cognition, vol. 2: Psychological and biological models* (Vol. 2, pp. 7-57). Cambridge: MIT Press.

Sadek, D., and de Mori, R. (1998). Dalogue systems. In R. de Mori (Ed.), *Spoken dialogues with computers* (pp. 523-561). London: Academic Press.

Santos, C. N., Xiang, B., and Zhou, B. (2015). Classifying relations by ranking with convolutional neural networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguisticsand the 7th International Joint Conference on Natural Language Processing* (pp. 626-634). Beijing, China: Association for Computational Linguistics.

Schneider, S., and Kummert, F. (2016). Motivational effects of acknowledging feedback from a socially assistive robot. *International Conference on Social Robotics* (pp. 870-879). Cham: Springer.

Sethi, S., and Dixit, A. (2019). A novel page ranking mechanism based on user browsing patterns. *Proceedings of Software Engineering* (pp. 37-49). Springer.

Sillice, M. A., Morokoff, P. J., Ferszt, G., Bickmore, T., Bock, B. C., Lantini, R., and Velicer, W. F. (2018). Using relational agents to promote exercise and sun protection: Assessment of participants' experiences with two interventions. *Journal of Medical Internet Research, 20*(2), e48.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929-1958.

Sussenbach, L., Riether, N., Schneider, S., Berger, I., Kummert, F., Lutkebohle, I., and Pitsch, K. (2014). A robot as fitness companion: Towards an interactive action-based motivation model. *The 23rd IEEE International Symposium on Robot and Human Interactive Communication* (pp. 286-293). IEEE.

The Stanford Natural Language Processing Group. (2018). *Stanford Log-linear Part-Of-Speech Tagger*. Retrieved from https://nlp.stanford.edu/software/tagger.shtml

Tielman, M., Neerincx, M., and Brinkman, M. (2017). Generating situation-based motivational feedback in a PTSD E-health system. *International Conference in Intelligent Virtual Agents* (pp. 437-440). Cham: Springer.

Tonosaki, Y., Miyoshi, M., Koizumi, Y., Takeba, T., Tajima, M., and Miyatake, M. (2016). Punctual train operation with energy-saving driving advisory system in dense traffic railway. *Proceedings of 2016 IEEE International Conference on Intelligent Rail Transportation (ICIRT)* (pp. 482-486). IEEE.

Toxtli, C., Monroy-Hernandez, A., and Cranshaw, J. (2018). Understanding chatbot-mediated task management. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (p. 58). ACM.

Tsuda, K., Shin, H., and Scholkopf, B. (2005). Fast protein classification with multiple networks. *Bioinformatics, 21*(suppl_2), uu59-ii65.

Vaidyam, A. N., Wisniewski, H., Halamka, J. D., Kashavan, M. S., and Torous, J. B. (2019). Chatbots and conversational agents in mental health: A review of the psychiatric landscape. *The Canadian Journal of Psychiatry, 64*(7), 456-464.

Vajjala, S., and Meurers, D. (2016). Readability-based sentence ranking for evaluating text simplification. *arXiv preprint: 1603.06009*.

Wei, Z., Liu, Y., and Li, Y. (2016). Is this post persuasive? Ranking argumentative comments in the online forum. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (pp. 195-200). Berlin: ACL.

Weiner, B. (1982). An attribution theory of motivation and emotion. *Psychological Review, 92*(4), 548.

Yan, R., Song, Y., Zhou, X., and Wu, H. (2016). Shall I be your chat companion? Towards an online human-computer conversation system. *Proceedings of the 25th ACM International Conference on Information and Knowledge Management* (pp. 649-658). Indianapolis: ACM.

Yoon, I., Ng, G., Dong, X., Duan, H., Aggarwal, A., Shah, R., and Ku, G. (2017). SuperCaly: Smart life assistance by integrating a calendar app with location tracking, media organization and social connection. *Proceedings of IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 1-7). IEEE.

Zeiler, M. (2012). ADADELTA: An adaptive learning rate method. *arXiv preprint: 1212.5701*.

Zhou, M. X., Mark, G., Li, J., and Yang, H. (2019). Trusting virtual agents: The effect of personality. *ACM Transactions on Interactive Intelligent Systems (TiiS) - Special Issue on Highlights of ACM IUI 2017, 9*(2-3), 10:1-10:36.

Zhu, G., and Iglesias, C. A. (2017). Computing semantic similarity of concepts in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering, 29*(1), 72-85.

Research Achievements

Journals

1. **Swieczkowska, P.**, Rzepka, R., and Araki K. (2020). Stepwise Noise Elimination for Better Motivational and Advisory Texts Classification. *Journal of Advanced Computational Intelligence and Intelligent Informatics, 24*(1), 156-168.

International conferences

1. **Swieczkowska, P**., Bachan, J., Rzepka, R., and Araki, K. (2017). Asystent: A Prototype of a Motivating Electronic Assistant. *Proceedings of the Linguistic and Cognitive Approaches To Dialog Agents (LaCATODA 2017), CEUR Workshop Proceedings 1926*, (pp. 11-19).
2. **Swieczkowska, P.**, Rzepka, R., and Araki, K. (2018). Analyzing Motivating Texts for Modelling Human-Like Motivation Techniques in Emotionally Intelligent Dialogue Systems. *Advances in Intelligent Systems and Computing, 848* (pp. 355-360).
3. **Swieczkowska, P.**, Rzepka, R., and Araki, K. (2019). A Convolutional Neural Network for Ranking Advice Quality in Texts for a Motivational Dialogue System. *Proceedings of the Linguistic and Cognitive Approaches To Dialog Agents (LaCATODA 2019), CEUR Workshop Proceedings 2452*, (pp. 35-44).

Domestic conferences and symposia

1. **Swieczkowska, P.**, Bachan, J., Rzepka, R., and Araki, K. (2017) An electronic assistant that provides motivation. 工学系イノベーションフォラム２０１７.

Acknowledgments

I would like to express my gratitude to the following people:

Prof. Kenji Araki
for his unending encouragement and support throughout all my years in the Language Media Laboratory;

Assistant Prof. Rafał Rzepka
for providing valuable advice on all my research projects and endeavors;

Members of the review committee:
Prof. Yuji Sakamoto, Prof. Miki Haseyama and Prof. Toshihiko Ito
for their insight and help in refining this thesis;

Associate Prof. Hiroyuki Iizuka
for his crucial input on both my neural network projects;

Colleagues from the Language Media Laboratory
for all the interesting discussions and mutual support;

My boyfriend Ireneusz Bugański and **my good friend Marta Marecka**
for their invaluable support and guidance in my times of doubt;

All my friends, both in real life and online
for their encouragement and unswerving belief in my abilities;

and

Ms. Harue Ishii
for once again making all this possible.