NTNU
Norwegian University of
Science and Technology

Top-K Item Recommendations Using Social Media Networks

Using Twitter Profiles as a Source for
Recommending Movies

Trong Huu Nguyen

Master of Science in Computer Science
Submission date: June 2018
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

The advent of internet has served as an offspring for the significant growth of online services and businesses such as e-commerce, entertainment, or social media. A common element among these industries is the process of tailoring the offered services or products towards their users' interests and preferences, also known as *personalization*. Related to this is the *cold start problem*, wherein systems may not have sufficient data on new users or customers in order to provide reasonable, personalized recommendations.

In an attempt to overcome said challenge, this thesis investigates the use of user data available from social media - in this case public Twitter profiles. A *two-step recommender* system is proposed and implemented, using the aforementioned data as input and basis for its predictions. The first step of our approach involves classifying and filtering Tweets based on their expressed sentiment, using Artificial Neural Networks to achieve state-of-the-art classification performance. Following this, we experiment with various combinations of recommender algorithms in order to match a specific user's preferences with the aspects of any movie.

The experiments examine the impact of numerous variables, including preprocessing techniques, feature extraction, similarity measures, word embeddings, entity matching and social circles with regards to the systems predictions. The results produced from the recommender system for any given user is a ranked list of movie titles with corresponding similarity scores.

We evaluate the proposed system on a larger set of Tweets annotated with sentiments, a set of Twitter user profiles, as well as a set of movies with associated data from IMDb. A prediction accuracy of 65% in the most successful case was reached, in which user-based collaborative filtering was utilized.

Overall, the results of our experiments indicate that the use of social media profiles has merit in the task of movie recommendations, and may have applications in other domains as well.

Sammendrag

Internettets frammarsj har åpnet for en betydelig vekst i tjenester og bedrifter med nettet som basis, for eksempel innen e-handel, underholdning eller sosiale medier. Felles for disse industriene er fokuset på å skreddersy tjenester eller produkter til brukernes interesser og preferanser, en prosess som også er kjent som *personalisering*. Et problem som er relatert til dette er *kaldstartproblemet* (cold-start problem) der systemer ikke kan tilby tilpassede anbefalinger til nye brukere grunnet mangel på data om brukerne.

I et forsøk på å løse nevnte utfordring, vil vi i denne oppgaven undersøke om en kan bruke data som er tilgjengelig på sosiale medier - i vårt tilfelle av offentlige brukere på Twitter. Vi presenterer og implementerer et *to-stegs anbefalingssystem* der vi bruker tidligere nevnte data som grunnlag for anbefalinger. Det første steget involverer å klassifisere og filtrere Twitter-meldinger basert på deres uttrykte følelser eller *sentimenter*, der vi bruker kunstige nevrale nettverk til å oppnå ytelse tilsvarende løsninger som er rådende innen feltet. Deretter eksperimenterer vi med ulike kombinasjoner av anbefalingsalgoritmer for å kunne skreddersy spesifikke brukeres preferanser i forhold til en hvilken som helst film.

I eksperimentene undersøker vi kombinasjoner av en rekke variabler og deres innvirkninger på systemets forutsigelser, deriblant preprosessering, ekstrahering av egenskaper, sammenligningsmetoder, matching av enheter og sosiale sirkler. For enhver bruker vil anbefalingssystemet produsere en rangert liste av filmtitler med tilhørende sannsynlighet.

Det foreslåtte systemet evalueres mot et større sett av Twitter-meldinger kategorisert etter uttrykte følelser, en samling av Twitter-brukerprofiler, samt et mindre sett med filmer og tilhørende data fra IMDb. I det beste tilfellet oppnår løsningen en nøyaktighet på 65% der anbefalingene er basert på likhet mellom brukere (*user-based collaborative filtering*).

Alt i alt indikerer resultatene fra eksperimentene våre at bruken av data fra sosiale medier kan ha nytte for seg når det gjelder filmanbefalinger. I tillegg kan den foreslåtte løsningen ha tilsvarende bruksområder innen andre domener.

Preface and Acknowledgements

This thesis project serves as the final fulfillment towards the requirements needed to conclude the Master of Science degree at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The thesis project was supervised by Heri Ramampiaro, Associate Professor at the Department of Computer Science (IDI). I would like to thank Heri for the initial project proposal, as well as valuable feedback and excellent guidance throughout the entire process.

The work presented is a result of an initial specialization project conducted during the fall semester of 2016 which served as a basis for the continued work and refinement in the spring semester of 2017. However, due to the unfulfillment of certain formal requirements needed to officially begin work on the thesis, I was not allowed to deliver the thesis for that semester. Work on the thesis was resumed and finalized during the spring semester of 2018.

Finally, I would like to thank my close friends and family, as well as my fellow students and colleagues for their continued encouragement and support.

Trong Huu Nguyen
Trondheim, June 2018

Table of Contents

Abstract i

Preface v

Table of Contents ix

List of Tables xi

List of Figures xiii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Context . . . 3
  - 1.2.1 Twitter . . . 3
  - 1.2.2 Movie Recommendations . . . 3
  - 1.2.3 Feature Engineering . . . 3
  - 1.2.4 Sentiment Analysis . . . 4
- 1.3 Research Questions . . . 4
- 1.4 Thesis Outline . . . 4

2 Background & Theory 7
- 2.1 Traditional- vs Microblog-Information Retrieval . . . 7
- 2.2 Twitter . . . 8
  - 2.2.1 Obtaining Datasets . . . 8
  - 2.2.2 Challenges . . . 8
- 2.3 Recommender Systems . . . 10
  - 2.3.1 Collaborative Filtering . . . 11
  - 2.3.2 Content-Based Filtering . . . 13
- 2.4 Sentiment Analysis . . . 14
  - 2.4.1 Document level sentiment classification . . . 14
  - 2.4.2 Sentence level sentiment classification . . . 15

2.4.3 Aspect-based opinion mining . . . 15
2.4.4 Sentiment lexicon generation . . . 16
2.4.5 Challenges . . . 17
2.5 Information Extraction . . . 17
2.5.1 Named Entity Recognition . . . 18
2.5.2 Feature Engineering . . . 19
2.6 Classification . . . 21
2.6.1 Traditional Approaches . . . 22
2.6.2 Artificial Neural Networks . . . 22

3 Related Work **25**
3.1 Recommendations in Social Networks . . . 25
3.2 Feature Extraction for Sentiment Analysis . . . 27
3.3 Sentiment Analysis in Twitter . . . 28
3.4 Named Entity Recognition in Twitter . . . 29
3.5 Summary . . . 30

4 The Two-Step Recommender Approach **33**
4.1 Theoretical Approach . . . 33
4.1.1 Feature Extraction for User Profiles . . . 33
4.1.2 Feature Extraction for Movie Profiles . . . 34
4.1.3 Two-Step Recommender . . . 35
4.2 Sentiment Analysis . . . 36
4.2.1 Word Representations . . . 37
4.2.2 Classifiers . . . 39
4.2.3 Evaluation and Results . . . 43
4.3 Final System Overview . . . 45
4.3.1 Datasets . . . 45
4.3.2 Preprocessing . . . 49
4.4 Recommender System . . . 51
4.4.1 Feature Extraction for Profile Construction . . . 51
4.4.2 Similarity Measures . . . 52
4.4.3 Recommender Techniques . . . 52
4.5 Evaluation . . . 53
4.6 Summary of Experimental Setup . . . 54

5 Results and Discussion **55**
5.1 Preprocessing . . . 55
5.1.1 Output . . . 55
5.1.2 Text Processing . . . 55
5.1.3 Part-of-Speech Tagging . . . 57
5.1.4 Named Entity Extraction . . . 57
5.1.5 Sentiment Analysis . . . 57
5.2 Recommender System . . . 58
5.2.1 Results . . . 58
5.2.2 Movie Profile . . . 60

5.2.3 Similarity Measures 60
5.2.4 Social Circles 61
5.2.5 User Profile 61
5.2.6 Word Embeddings 63
5.2.7 Feature Extraction 64
5.2.8 Entity Matching 65
5.2.9 Recommender Techniques 65
5.3 Significance of Results 69
5.3.1 Sentiment Filtering 69
5.3.2 Power of Word Embeddings 69
5.3.3 Twitter Profiles for Recommendation 69
5.3.4 The Social Aspect 70

6 Conclusion and Future Work **71**
6.1 Conclusion 71
6.1.1 Goal Achievements 71
6.2 Future Work 73
6.2.1 Improvements to the Current System 73
6.2.2 Feature Selection 75
6.2.3 Social Circles 75
6.2.4 Machine Learning Classifiers & Recommender Systems 75
6.2.5 Other Domains 76
6.2.6 User-Based Evaluation 76

Bibliography **77**

List of Tables

4.1 Summary of pre-trained word embedding models used in our experiments 38
4.2 Overview of classifiers for sentiment analysis experiments 39
4.3 F_1-scores for combinations of sentiment classifiers and word representations 45
4.4 Comparison of named entity recognition tools, from Jiang *et al.* [78] . . . 50

5.1 Variable combinations for the recommender system experiments 59

List of Figures

2.1 Example of usage of hashtags in a Tweet 10

4.1 The overall general two-step recommender system approach 34
4.2 The multilayer perceptron network 40
4.3 The convolutional neural network 41
4.4 The convolutional neural network with a long short-term memory layer appended 42
4.5 The long short-term memory network 43
4.6 The bidirectional long short-term network 43
4.7 The overall view of the system and its components 46

5.1 Comparison of results for variations of sources for movie profiles 61
5.2 Comparison of results for variations of similarity measures 62
5.3 Comparison of results when including features from the social circle . . . 62
5.4 Comparison of results for variations of sources for user profiles 63
5.5 Comparison of results for variations of pre-trained word embedding models 64
5.6 Comparison of results for variations of feature extraction methods 65
5.7 Comparison of results when including named entity matching 66
5.8 Comparison of results for variations of recommender techniques 66

Chapter 1

Introduction

This thesis aims to explore the possibility of providing users with recommendations based on social media. A collection of data of users and their networks, as well as a collection of products with their corresponding information (e.g. comments or reviews) will serve as a basis for the proposal of a text and data mining method and a machine learning method to produce good recommendations.

1.1 Motivation

Over the past decade or so, social media has become a staple of the modern society's daily routine. Services such as Twitter connect millions of users, most of which share status updates and photos among their own social circles at a daily rate. The sheer scale of data generated is thus enormous. For example, Twitter had over 500 million daily status updates (henceforth referred to as *Tweets*) back in 2013[1]. Though often informal and short (sometimes even incomplete without contextual clues), the textual content can be valuable sources of information for both businesses and people alike. Social media enables the convenience of sharing thoughts and having conversations from anywhere, at any time - as long as you have Internet access.

The emergence of services such as Netflix, Amazon, and Spotify have also made their impact on the digital life. Advertisement and recommendations have moved from being general and impersonal to aggressively utilizing each individual's activity, and perceived tastes and preferences. While such systems will usually elicit a user's preference explicitly and/or implicitly, they usually do not use other sources of data other than their own. A new user in the system would then have little to no personalized recommendations until the system has collected enough data to make accurate predictions. This is called the *cold start problem*.

For example, a user could be looking to find a restaurant while visiting a new city, say New York. Instinctively, he would seek for recommendations (reviews or opinions from

[1] `https://blog.twitter.com/2013/new-tweets-per-second-record-and-how`

other people) of restaurants, whether online or not. He finds several websites containing reviews, but is overloaded with reviews and information. Believing that he could get personalized recommendations, he registers on one of the websites. However, being a new user with no previous ratings, the system fails to provide any useful recommendations. Given this scenario for any average user, having online services take advantage of their existing social media profile(s) would be an interesting and potentially useful addition. The impact of which would be significant if it enabled more and better personalized content while also circumventing the cold start problem if the user is new to the service. As for existing users, their social profiles could also be used in conjunction with their existing system profiles to potentially provide better recommendations. All in all, users would benefit from having a more pleasant and streamlined experience, while businesses would find their customers more satisfied which in turn could improve their reputation and result in increased revenue.

Ethically speaking, there is a trade-off in terms of privacy when providing services access to their personal profiles. Would users be comfortable with sharing their entire profile with third party businesses even if it results in a better user experience? Perhaps if they were able to control exactly what was shared, or were given guarantees that their information would not be abused or used in a way that the user did not intend it to be. Ultimately, it comes down to what each individual person is comfortable with sharing, as well as their trust in a given service/business/website. However, it is not unreasonable to assume that a significant amount of users would be interested in using their profiles for the option of personalized content, especially when considering that many users today leave their social media profiles open to the public. We will not consider the aspects of privacy and entrustment for the rest of the thesis, though we felt they were worth mentioning as they are important aspects to consider when working with user data.

With all of this in mind, is it possible to use a user's social media profile as basis for good recommendations? An active user of Twitter may have a lot of valuable data on their historical timeline which could provide insights that otherwise would be unavailable through traditional means. Explicit and varied opinions, places visited, and social circles are all examples of information a service like Netflix usually does not have access to, but if available have the potential to provide better recommendations. This is essentially the main motivation for this thesis, where we will specifically explore the movie recommender domain using data from Twitter. The development of a method for extracting information from a user's Twitter profile, and a method for using said information in terms of recommending movies given a set of movies will be the main contributions of this thesis, which will be demonstrated with a implementation of a prototype system incorporating these methods in a pipeline. While we do focus on the movie domain in terms of recommendation, we hope to be able to develop methods that can produce recommendations for other domains as well with minor adjustments.

1.2 Context

1.2.1 Twitter

Twitter is one of the largest news and social networking services, serving more than 300 million active users monthly as of early 2017[2]. The service is a so-called *microblog* service, in which posts by the users are short in terms of length. In fact, Twitter enforces a limit of 140 characters per Tweet[3]. Every user has their own timeline, where they essentially maintain their own personal blog. Additionally, revolving around the social networking aspect, every user also has their own dashboard consisting of Tweets published by users they follow or are subscribed to, henceforth referred to as *followees*. A user can analogously be followed by other users, henceforth referred to as *followers*. Any user can also mention or reply to other users in a Tweet, enabling public conversations between two or more users.

With social media being an informal platform for expressing thoughts and sentiments and the availability of Tweets in the public domain, Twitter data is increasingly used in a wide range of research fields[4]. Twitter also provides APIs for accessing their data, further incentivizing both researchers and developers alike to access and use their data.

1.2.2 Movie Recommendations

With the rise of the information and communication technology era, accelerated with the advent of Internet, the studies of recommender systems to help aggregate and summarize information have been prominent and thoroughly explored. In particular, the Netflix Prize competition (2006-2009)[5] sparked significant interest in improving algorithms for recommender systems, looking at the movie domain with users, movies and their related ratings. Since then, a host of sophisticated algorithms and systems have emerged, taking into account more information about users and items. With large datasets such as MovieLens publicly available, movies have typically been one of the commonly explored domains in the field of recommender systems.

The appearance and explosive growth of social media in the recent years has also brought new dimensions and possibilities in terms of personalization of recommendations. Taking individual utterances into account when eliciting preferences is a valuable aspect, especially in terms of personalized recommendations.

1.2.3 Feature Engineering

Many recommender systems rely on using features or attributes to represent both users and items, often depending on domain. Determining and extracting these features are computationally and architecturally demanding tasks. Considering the Twitter domain, the textual content itself is likely one of the most important sources for features. However with Tweets often being short and the content quite variable, representing the words using

[2] `https://about.twitter.com/company`

[3] `https://dev.twitter.com/basics/counting-characters`

[4] `https://blog.twitter.com/2015/twitter-data-research`

[5] `http://www.netflixprize.com/`

traditional methods such as vectors or matrices may not be feasible in terms of sparsity and scaling. Additionally, there are associated metadata with each Tweet, providing a host of possible features for analysis, e.g. data pertaining to spatial or temporal dimensions, or social circles.

1.2.4 Sentiment Analysis

Closely related to the notion of recommender systems is sentiment analysis. People and businesses alike have always been invested in the opinions of other people. Finding out what people like and dislike is valuable, e.g. in terms of eliciting user preference, or discovering aspects in which a product or business is lacking. This has become important with the rise of social media platforms and online discussion forums where users can easily express their opinions. Determining the sentiment of an utterance is however a difficult task for machines, due to the ambiguous nature of human languages. This problem is magnified by the often short, informal and noisy language of online text, especially on Twitter with its character limit. Common characteristics of such language include abbreviations, acronyms, slang, as well as erroneous grammar or spelling. Historically, methods of sentiment analysis were developed with the assumption of text being formal and complete, while in the case of online text this can no longer be a reasonable assumption, and adaptations or development of new methods for analysis may be needed. In terms of Twitter, using sentiment analysis to filter Tweets by their sentiment could prove useful, especially when eliciting user preference for recommendation.

1.3 Research Questions

In our exploration of the possibilities of improving recommender systems by utilizing the valuable data contained within Twitter and development of methods to do so, we will attempt to answer two main research questions:

RQ 1 *How can we use Twitter profiles to recommend movies?*

RQ 2 *How viable is it to leverage Twitter profiles for recommendation with regards to the cold start problem?*

Answering the first question involves answering the following subquestions:

RQ 1.1 *How can we identify and extract important features from a user's Twitter profile?*

RQ 1.2 *How does a user's social circle affect these features?*

RQ 1.3 *How can we use these features to recommend movies to users?*

1.4 Thesis Outline

This thesis begins with **Chapter 1**, introducing the motivation and field of research it aims to explore as well as presenting the work's context and research questions. The

background and theory for the topics treated in this thesis are established in **Chapter 2**, further elaborating on research and challenges with regards to Twitter, recommender systems, sentiment analysis, and information retrieval in general. In **Chapter 3**, a survey of literature pertaining to information retrieval in social media is presented. Following this is **Chapter 4**, where we present and explain the project's overall approach to the acquisition of a dataset as well as the implementation of a prototype system. The results outputted from the system are showcased and reviewed in **Chapter 5**. Finally, in **Chapter 6** we conclude our findings and contributions, and present our achievements with regards to the research questions. Propositions for future work are also presented here.

Chapter 2

Background & Theory

This chapter presents the background and theory related to the context and scope of this thesis. Section 2.1 compares and presents the differences between information retrieval in the traditional versus the microblog domain. Following this is an elaboration of the characteristics and challenges of information retrieval in the context of Twitter in section 2.2. Sections 2.3 and 2.4 present established theory within the fields of recommender systems and sentiment analysis, respectively. In section 2.5, we look at the field of information extraction, addressing specific subtasks relevant to our thesis. Lastly, in section 2.6, we look at traditional methods and approaches in classification tasks, with a short introduction to artificial neural networks and deep learning.

2.1 Traditional- vs Microblog-Information Retrieval

While information retrieval (IR) is a thoroughly studied field within Computer Science, the basis for such studies have historically been restricted to longer, formal texts such as news articles, journals, and other literature. Most social media on the other hand can be viewed as microblogs, which are a series of usually short status updates with often informal language and structure. In this context, traditional methods of retrieving information may not always be applicable or efficient, as they might not consider contextual clues or noisy aspects of the written language (e.g. abbreviations, hyperlinks, emoticons, sarcasm, or orthographic errors). Additionally, a lot of the data might just be spam, generated by bots with the purpose of enticing users to click on links, or perhaps attempt to sway public opinion by spreading propaganda. Twitter reported back in 2014 that 23 million of their 271 million active users were in fact bots, amounting to around 8.5% of its active userbase[1]. A more recent study focusing on the detection of Twitter bots estimates that roughly 9% to 15% of active Twitter accounts are bots, amounting to figures between 27 million and 45 million bots as of 2017 [1].

[1] http://www.techtimes.com/articles/12840/20140812/twitter-acknowledges-14-percent-users-bots-5-percent-spam-bots.htm

Another aspect to consider is the notion of relevance. While articles often have clear topics and structure, microblog posts tend to be ambiguous and lack conventional structures due to their informal and short nature. Deciding whether a post is relevant or not with regards to a topic or query is thus a difficult task. As users often have the ability to share or forward other users' posts to their own social circle, one should also consider the relevance and strength of such posts with regards to the user's personal sentiments.

2.2 Twitter

2.2.1 Obtaining Datasets

Very large and detailed datasets from Twitter are usually not available in the public domain, largely due to Twitter's developer policy of respecting a user's control and privacy of their Tweets[2]. Most datasets available are thus often either anonymized or purged of any data that may violate user privacy. The methods and explorations in this project requires certain data not usually provided, such as data related to a user's timeline, their followers as well as the users they follow. In addition to this, there are no publicly available datasets specifically for the movie domain that satisfies our needs at the time of this thesis. Obtaining our own dataset was thus deemed necessary.

Fortunately, Twitter makes a subset of its data available from two separate endpoint APIs, the *Search API*[3] and the *Streaming API*[4]. These endpoints provide recent data from Tweets published in the last 7 days and as a constant incoming stream of Tweets as they are published, respectively. Additional parameters such as querying (e.g. restricting data to match certain keywords and/or hashtags), Tweet language and geolocalization are also available.

Of course, certain restrictions are in place to hinder abuse and stress on Twitter's servers. For our purposes, these limits range between 15 to 900 requests per 15 minutes[5]. Obtaining a substantial amount of data is thus a time-consuming task. Another issue with accessing data through these official APIs is the limit of data available. A user's personal timeline may have tens or hundreds of thousands of Tweets, however the Search API only allows for the user's 3200 most recent Tweets to be returned.

2.2.2 Challenges

As mentioned in section 2.1, there are many challenges to consider when working with data from Twitter, as opposed to more traditional sources.

Sparsity The length of Tweets are limited to 140 characters as mentioned previously, a restriction that leads to interesting effects. As with many social media posts, Tweets are often informal, short and concise [2]. One could argue that the short length does not provide a sufficient basis for extracting information, especially when considering recommending

[2] `https://dev.twitter.com/overview/terms/policy.html#c-respect-users-control-and-privacy`
[3] `https://dev.twitter.com/rest/public/search`
[4] `https://dev.twitter.com/streaming/public`
[5] `https://dev.twitter.com/rest/public/rate-limits`

movies based on Tweets. Detecting features and sentiments may not be feasible given the short length.

In the context of recommending movies, we would be interested in looking for Tweets expressing sentiments about movies. While a traditional movie review would represent a user's composite sentiment on several aspects and features of a movie, a user's Tweet might only express a single sentiment pertaining to a single aspect or feature. On one hand, a Tweet can be a well-defined representation of a single sentiment. A user is thus able to express multiple sentiments on multiple aspects over a series of Tweets. On the other hand, the length restriction could encourage users to express complex sentiments using a single Tweet.

Noise, Spam and Retweets In section 2.1 we saw that around 8.5% of Twitter's user base in 2014 consisted of active bots. The spam generated from these would likely constitute a substantial amount. While it would be interesting and useful to detect and filter out Tweets that are generated by bots, this has not been a priority in this thesis.

Another challenge with Tweets is that they can revolve around any topic of the user's choice. Finding data pertaining to the movie domain thus involves filtering out uninteresting and mundane Tweets. The usage of *hashtags*[6] to categorize Tweets is helpful here, as well as using named entity recognition methods for additional support. Domain specific mundane Tweets should also be filtered out, e.g. in the movie domain there might be Tweets reporting on premieres, box office results, trailers, external movie reviews, etc.

A unique functionality of Twitter is the ability to *Retweet* a Tweet. In essence, a user re-posts or shares another user's Tweet to their own timeline. While a Retweet could imply a shared sentiment with its original poster, a dataset consisting of a high amount of Retweets is redundant and does not provide much information or unique insights.

Language The language used on social media poses another challenge. As mentioned earlier, Tweets, like most social media posts, are often short and informal in nature. As a consequence grammar and spelling are usually not a priority, often leading to unconventional or incomplete sentences. Using classic natural language processing systems designed and trained on "proper" corpora, i.e. books, journals or articles, could thus prove difficult.

With Tweets being so short in nature it could be reasonable to assume that the Tweets are easier to classify as the restriction encourages terseness in the language. On the other hand, complex and compact forms of expressions may rise. In recent years the usage of emojis to express feelings and sentiments have become ubiquitous. In fact, Oxford Dictionary announced an emoji as the 2015 Word of the Year[7] and over 110 billion emojis were present in Tweets between 2014 and 2016[8].

In terms of the domain of movie recommendations, using emojis to determine sentiments could be an interesting approach. Some works exploring the meanings and usage of emojis exist. Kralj *et al.* [3] looked at determining the emotional content of emojis and

[6] https://support.twitter.com/articles/49309

[7] http://blog.oxforddictionaries.com/press-releases/announcing-the-oxford-dictionaries-word-of-the-year-2015/

[8] https://blog.twitter.com/2016/introducing-emoji-targeting

proposed an emoji sentiment lexicon, mapping 751 emojis to their corresponding sentiment polarities. Such a lexicon could have potential use in automated sentiment analysis of text in social media.

Hashtags As with many social media platforms, a characteristic of Twitter is the widespread usage of hashtags. While mostly used to help categorize Tweets by topic or theme, certain users might replace words in their Tweets with hashtags as seen in figure 2.1.

Figure 2.1: Example of usage of hashtags in a Tweet

The use of compound hashtags (i.e. hashtags consisting of multiple words, illustrated in figure 2.1) also poses as a challenge in terms of automatically identifying and splitting individual words. Splitting these by detecting uppercase letters is a naive solution, but fails when considering hashtags in e.g. all lowercase. Some research on compound splitting exist, with Koehn and Knight [4] and Macherey *et al.* [5] among the most prominent. Research within this area still appears to be in its infancy, with efficient methods for splitting not widely available.

2.3 Recommender Systems

With increased availability of online information, users may find it difficult to search for and recognize items that may be of interest to them. A recommender system aims to help with managing this information overload by tailoring recommendations to individual users by exploiting available data (e.g. user preferences, item characteristics, popularity, ratings, reviews, context).

Typically, the system attempts to provide the user with a serendipitous experience by recommending largely unknown or less popular items (i.e. items from the *long tail*). Depending on the domain and purpose, a user might already know what they are looking for. In that case the system should also be able to provide the user with the correct proposals.

Historically, as described in Zafarani *et al.* [6], Rajaraman and Ullman [7], Jannach *et al.* [8], and Sarwar *et al.* [9], there have been three types of approaches to recommender systems:

Collaborative filtering identifies users with similar rating profiles and recommends items based on this neighborhood. This can also be used as a basis

for predicting a user's rating of an item based on similar users' ratings.

Content-based systems are based on the properties associated with the content and a user's preferences. The content recommended is similar to the ones the user has previously liked or rated highly, or alternatively matching the user profile.

Hybrid approach where multiple different recommendation techniques can be used to combine predictions. Combining different recommenders may help in overcoming their respective shortcomings.

These techniques have been widely used over the past few decades, and through time enhanced with other information (e.g. demographics, tags, social context).

In recent years, **personality-based** systems have been proposed where a user's personality is taken into consideration when eliciting user preferences. In psychology, personality is believed to be a critical influence on preferences. Thus researchers believe it can be used to enhance both prediction quality and user experience. Buettner [10] also proposes a personality-based recommender framework in which a user's personality is predicted using social media data. Product preferences is subsequently derived based on this personality.

2.3.1 Collaborative Filtering

Collaborative filtering (CF) is based on the assumption that users who have previously agreed on ratings will likely agree in the future. Additionally, one can also assume that items that have received similar ratings in the past will receive similar ratings in the future. A user-item matrix is usually the basis for collaborative filtering techniques. This is used to calculate the similarity of either the users or the items and subsequently calculate missing entries and recommend the highest rated predictions.

Similarity Similarity can be calculated using different measures, commonly one of either Jaccard similarity, Cosine similarity, or Pearson correlation. Typically, the predictions are calculated using either historical data directly, or by assuming that an underlying model can be approximated and learned. These approaches are called *memory-based* and *model-based* collaborative filtering, respectively.

Memory-based collaborative filtering A distinction between two forms of memory-based CF is made here, *user-based CF* and *item-based CF*. In the former, similarities and predictions are calculated based on the users. That is, given a user u and an item i which u has not rated, find other users whom have rated i while also having similar previous ratings as u. A rating for item i by user u can thus be inferred using an aggregation of the other similar users' ratings. A ranked set of recommendations could then be produced by finding the N highest predicted ratings for the user.

However, new users often lack ratings. Whenever new ratings are added the system must recompute the similarities, which scales poorly for a business with millions of users and items. Another issue is the diverse and complex nature of humans. Even though two

users may coincidentally like the same two different genres of e.g. music, they may still have diverging preferences otherwise.

An item-based CF makes predictions using the similarity between items instead. Generally, items are simpler which means their similarities are more stable and require less frequent recalculations. With systems generally having more users than items, the items also tend to have more ratings than an individual user, and as such the average rating for an item will seldom change, relatively speaking. The general idea behind the item-based approach is to first construct an item-item matrix, in which the similarity between each pair of items is determined. The approach to calculating similarities vary, from using correlation between rating to calculating the cosine similarity between the items using their set of ratings as vectors. When the matrix is constructed, one can find and recommend other items which are similar to an item i that a user u has rated highly.

Model-based collaborative filtering Rather than using the rating matrix directly at run time, a model can be learned offline and used to make predictions when needed. While building and updating the model can be computationally expensive, using the model for predictions at run time is much cheaper than the alternative memory-based filtering. There are a number of approaches to learning a model, including matrix decomposition (an instance of the more general Singular Value Decomposition), probabilistic methods (Bayesian classifiers), association rules, and clustering.

Challenges A number of challenges with the collaborative filtering approach exist:

- Collaborative filtering approaches depending on nearest neighbor algorithms face scalability issues with a high number of users and items.
- Another issue is data sparsity. There is often a large amount of users, however most will only have rated a fraction of the available items. Again, nearest neighbor algorithms are unable to find similar users and as a result may recommend items with poor accuracy. A common solution strategy for recommender systems is to gather data either explicitly by e.g. asking users to rate items or provide a ranked list of their preferred items, or alternatively inferred implicitly by e.g. observing user behavior (historical purchases, product page views, viewing times) or analyzing their social profiles.
- Related to the sparsity problem is the cold start problem where new users and items have no or few ratings, leading to the system being unable to accurately calculate predictions.
- Being based on neighbors, certain algorithms may be prone to popularity bias. A possible solution might be to give more weight to items with high variance.
- As collaborative filtering does not require any actual knowledge of the items recommended, the approach also lacks explanation of the results presented to the users.

2.3.2 Content-Based Filtering

The previously discussed collaborative filtering methods do not exploit the information about the items, even though it might prove useful. Content-based recommenders are based on an item profile representing important features of the item as well as a user profile describing the user's preferences. An item is thus recommended based on the similarity of an item profile and the user profile.

Item profiles An item profile is a set of important features, e.g. information such as genre, actors, directors for a movie domain, usually found through keyword descriptions or through the content itself. There are many approaches to finding the features of an item. A common approach is using *tf-idf* (term frequency-inverse document frequency) to create a vector of weighted terms. Improvements here include using domain or lexical knowledge, or various natural language processing techniques such as removal of stop words, stemming and lemmatization. More advanced techniques utilizing topic modeling to group synonyms also exist.

User profiles Creating a content-based user profile that describes a user's preferences usually involves creating a vector of weighted features that they may prefer, using previously observed interactions with the system. A possible user profile could be a weighted average of previously rated item profiles.

Recommending items There are many approaches to recommending items based on content. From simple algorithms such as k-nearest neighbors using cosine similarity between the user and item profile vectors, to more sophisticated methods utilizing machine learning algorithms for classification of items using the user and corresponding item profiles as training data. Examples of such classifiers include Rocchio, Naive Bayes, decision trees, and Support Vector Machines (SVMs).

Challenges As with collaborative filtering, there are a number of limitations with content-based recommender systems:

- Finding appropriate features for an item is difficult. Keywords may be irrelevant, the content might be lacking in length or depth, or is difficult for machines to extract meaningful information from (e.g. multimedia).
- Constructing a user profile for a new user with few or no ratings is also an issue. Gathering explicit and implicit feedback during the ramp-up phase is a possible remedy. Alternatively looking at other sources of information such as social media is also a possibility.
- Algorithms may be prone to overspecialization, or in other words only recommend items matching the user's content profile.

2.4 Sentiment Analysis

Another aspect of the web is the increasing amount of user-generated content. With a plethora of platforms available for users to express their opinions and experiences about anything, the threshold for sharing is significantly reduced. Businesses and individuals alike may find others' opinions valuable or influential, especially whenever a decision is to be made. Sentiment analysis or opinion mining is concerned with the computational study of the opinions or sentiments about entities expressed in a text. According to Liu [11], an entity is a product, person, event, organization, or topic. The entity itself can be represented as a hierarchy of components (e.g. an iPhone) and parts of said component (e.g. screen, battery), each with their own set of attributes (e.g. call quality, battery life). For simplicity, both components and attributes will be referred to as aspects. An opinion can be expressed on any aspect, and either targets a single entity or compares more than one entity with shared aspects. The concepts of subjectivity and emotion are also closely related to sentiments and their strength.

Sentiment analysis is a difficult problem, touching on many aspects of natural language processing (NLP). Early naive approaches have been dependent on the occurrence of keywords, which do not consider context and domain, and are especially weak to negations. Approaches in the recent times have used more sophisticated techniques including machine learning (convolutional/recurrent neural networks, deep learning, naive Bayes classifiers, support vector machines), statistical methods, lexicons, and knowledge bases.

Historically the analysis has had three levels of granularity: document-level, sentence level and entity/aspect-level, with the research trends moving towards the finer-grained levels of analysis.

2.4.1 Document level sentiment classification

At this level the task is to classify the entire document, i.e. to determine the overall sentiment expressed. This also assumes that the document only addresses a single entity, and is expressed by a single opinion holder. Most techniques here use some form of learning, either supervised (Naive Bayes, Support Vector Machines) or unsupervised. Features and techniques used here include part-of-speech tagging, tf-idf, semantic shifters, syntactic relations, and lexicons.

An unsupervised approach described by Turney [12] identifies phrases matching certain patterns using a part-of-speech tagger, calculates the semantic orientation of the phrases using Pointwise Mutual Information, and finally calculates the average orientation for the entire review.

Another unsupervised approach using a lexicon was proposed by Kreutzer and Witte [13]. The approach involves expanding a semantic lexicon with domain-independent sentiments. Sentiment-carrying words is then extracted from the text, assigned a three-sentiment score (positive, negative, objective), and finally the scores are aggregated for the entire document.

2.4.2 Sentence level sentiment classification

Going deeper into the document, this level classifies each sentence. While offering more detail of the expressed sentiments of the document, the specific entity or aspect the opinion targets still remains unknown. The assumption that a sentence holds a single sentiment from a single opinion holder is also a limitation, ruling out analysis of comparative or complex sentences. Classification is usually either a two-step problem, or alternatively a 3-class problem. The first step is to classify the subjectivity of the sentence (i.e. opinionated or not). This has been done with known supervised learning methods, as well as unsupervised learning using pattern learning with syntactic templates. The second step is of course to classify the sentiment of the sentence, in which methods similar to the ones used in the document-level classification can be used. Davidov *et al.* [14] studied sentiment classification of Twitter posts (or tweets), using features such as hashtags, punctuation, emojis and frequent patterns among them.

2.4.3 Aspect-based opinion mining

The problem with the previous levels of classification is that neither could find exactly what the opinion holder liked or disliked, which for many applications is insufficient. In aspect-based sentiment analysis the opinion target is decomposed to entities and aspects.

Liu [11] defines an opinion as a quintuple:

$$(e_i, a_{ij}, h_k, t_l, s_{ijkl}) \tag{2.1}$$

where e_i is the name of an entity, a_{ij} is an aspect of e_i, h_k is the opinion holder, t_l is the time when the opinion is expressed by h_k, and s_{ijkl} is the sentiment of aspect a_{ij} of e_i by h_k at t_l.

The task is to find all opinion quintuples in the document. For the quintuple definition of opinion, we can break this task down into five separate subtasks:

Entity extraction and categorization — An entity may be referred to using various expressions. All synonymous expressions extractions should be grouped into the same category, resulting in the representation of a unique entity. This is related to a classic problem within information extraction called named entity extraction.

Aspect extraction and categorization — Analogous to entity extraction and categorization. An aspect expression, which may be implicit, e.g. "expensive" which refers to the aspect "price", or explicit e.g. "battery life", indicates the aspect category, while each category represents a unique aspect of an entity.

Opinion holder extraction and categorization	This task is also analogous to the previous tasks. Discover and categorize all opinion holders in the document.
Time extraction and standardization	Analogous to previous tasks, extract the times of opinion statements and standardize them.
Aspect sentiment classification	Determine whether the sentiment of an opinion of an aspect is positive, neutral, or negative. Alternatively assign a numerical value indicating sentiment.

Two of the most important subtasks will be elaborated upon below.

Aspect extraction and categorization Numerous approaches to aspect extraction have been proposed. One approach is based on finding frequent nouns and noun phrases in a data set, assuming that most reviews will refer to a converging set of aspect expressions. Non-frequent nouns is thus assumed to be irrelevant or less important.

Another approach involves exploiting the relations between an opinion and its target. Knowing that sentiment words describe or modify aspect expressions, one can find the aspect by finding the nearest noun or noun phrase. These dependency relations have been exploited by researchers, e.g. in a double propagation approach by Qiu *et al.* [15]. This approach uses known sentiment words to identify aspects, both of which are then used to identify more sentiment words and aspects, and so on. Extraction is performed using dependency relation rules.

Recent approaches for clustering the extracted aspects exploit prior or lexical knowledge. Examples include Expectation-Maximization (EM) [16] and constrained Latent Dirichlet Allocation (LDA) [17].

Aspect sentiment classification Determining the aspect sentiments involves identifying words that convey positive or negative sentiments. Approaches may for example be based on supervised learning (which depends on the training data), or apply sentiment lexicons in an unsupervised manner. A simple approach is to aggregate opinion words of each aspect in a sentence, taking into account sentiment shifting (e.g. negations) and coordinating conjunctions (e.g. *and, or, but*). Other rules can also be exploited to determine the sentiment of an opinion, e.g. production/consumption of resources or waste, decreased/increased positive or negative.

2.4.4 Sentiment lexicon generation

Sentiment words, phrases and idioms are an instrumental part of sentiment analysis. These are collectively called lexicons, which indicate the sentiment, subjectivity, or emotion of words. Compiling such lexicons can be done manually, or automated by using either a dictionary-based or corpus-based approach.

Dictionary-based approach This approach exploits a dictionary's structure, using an initial seed set of words with known sentiments which can be collected manually. The set is then iteratively expanded by an algorithm that searches and adds the synonyms and antonyms of the words. Variations of this approach has been suggested by researchers, ranging from the usage of machine learning to probabilistic methods. Although this approach enables the ease of compiling large sentiment lexicons quickly, the main disadvantage of this approach is that the words found are general. Context- or domain-dependent words are thus difficult to find.

Corpus-based approach While the dictionary-based approach does not handle domain- or context-dependent words, a corpus-based approach can use a seed set of general-purpose sentiment words to find more sentiment words from a domain corpus using e.g. syntactic or co-occurrence patterns. An early key approach by Hatzivassiloglou and McKeown [18] uses connectives (i.e. *and, but, either-or, neither-or*) to identify more sentiment words as well as their orientations. A learning step produces a graph with links between adjectives indicating same- or different-orientations, the result of which is used in a clustering step to produce sets of positive and negative words.

2.4.5 Challenges

Although much research has been done there are still many issues and challenges that remain.

- Dealing with complex sentences, e.g. sentences with conceptual rules, factual sentences implying opinions, or sarcasm.
- A significant amount of the data available on social media is noisy, i.e. the data may include irrelevant content, interactions with other users, and contain grammatical errors. Preprocessing the data for analysis is thus a very important step.
- There are also difficulties in handling domains with little training data, as well as the diverse nature of each domain in their unique, context-dependent ways to express positive or negative sentiments.
- Cross-domain and -language analysis is also a problem. Learning a classifier for a particular set of training data results in poor performance when applied to data from another context or domain. The same applies to language as syntaxes and semantics usually differ. Additionally, there may be a lack of labeling and resources such as lexicons in non-English languages.

2.5 Information Extraction

The field of information extraction involves automatically extracting knowledge of unstructured nature from a large data collection. The case of natural language processing (NLP) is an example of this, concerning the automated processing and management of human language texts. While earlier research focused on extracting text from documents

with a well-defined structure, the emergence of social media and informal domains has sparked interest in researching and solving the problems of processing such informal and noisy text. For example, Freitag [19] studied the use of machine learning for information extraction in informal domains, proposing a combination of multiple approaches and learners as a strategy yielding performance on par or better than the state-of-the-art at the time.

Typically, the goal of information extraction is to simplify and structure the information found in a semi-structured or unstructured text in order to enable machines to read and process the text. There are many subtasks in information extraction, depending on the domain and context, though we will only address the most relevant ones in the following subsections.

2.5.1 Named Entity Recognition

The problem of named entity recognition was originally defined in Grishman and Sundheim [20], a task which involves identifying names and types of entities, e.g. people, organizations, locations. They also included the annotation of numerical expressions (currency and percentages) in their task specification. Hua *et al.* [21] categorizes most of the previous work into two approaches: rule-based, and statistical approaches.

Rule-based Approach Rule-based approaches identify named entities within given domains by using heuristic rules. In early approaches these were handcrafted and built by domain experts, while in later years the use of machine learning for automatic induction of rules. The premise of applying such machine learning techniques is of course the usage of features describing the characteristics or attributes of words. Nadeau and Sekine [22] present some typical features. These range from word-level features (e.g. case, punctuation, digits, character, morphology among others), to list lookups (using e.g. lexicons, gazetteers, dictionaries or other sources of domain knowledge), and document/corpus features (looking at occurrences, syntax, frequencies or meta-data).

An advantage with these types of approaches is the efficiency and low cost of computation. The manual addition or modification of rules to optimize and adapt to the domain is also an advantage. It is however difficult to generate all rules so that they cover all possible cases.

Statistical approach The statistical approach involves decomposing the unstructured texts into tokens or chunks and then labeling these decomposed parts. Machine learning is also used here to train models for automatic classification, and may use the features described previously in the rule-based approach. A severe disadvantage however is the requirement of manually annotated data for training, an often time-consuming and laborious task. These datasets also affect the result and performance of the trained models, and producing the

Approaches have ranged from using Support Vector Machines with Hidden Markov Models, as proposed by [23], [24], to more advanced models in later years such as Conditional Random Fields (CRFs) [25], [26]. Current state-of-the-art approaches use these CRFs in combination with recurrent and convolutional neural networks; examples include Chiu and Nichols [27], Ma and Hovy [28], and Lample *et al.* [29]. These approaches

typically use word embeddings (mappings of vocabulary to vectors of real numbers) and effectively eliminate the need for traditional feature engineering.

2.5.2 Feature Engineering

An important aspect of the application of machine learning is the notion of feature engineering. Typical machine learning tasks, such as classification, requires input that are represented in a convenient manner. However, data gathered from real-world scenarios are usually convoluted and variable. Feature engineering is thus the task of extracting useful features or representations from raw data in order to be utilized in machine learning tasks. In the context of text, this usually involves somehow mapping or embedding the vocabulary to numerical representations, e.g. in vectors or matrices.

Bag-of-words A common baseline approach of extracting of these features in terms of text is the bag-of-words (or n-grams) approach, representing documents as a vector that describes occurrences of the words. From a set of text documents we extract the set of unique words or terms into a dictionary with an associated mapping of each word to an index. In its simplest form, the model then converts each document into a list of terms, represented as a numerical vector. The *i*'th entry in the vector will thus correspond to the term and index pair in the dictionary representation of the vocabulary extracted earlier, with the entry value representing the frequency or weight of the term. As a consequence, the bag-of-words model ignores the order in which the words appear in the documents. To address this shortcoming is the usage of the *n-gram model*, in which text units of length *n* are stored in the set rather than just single words. This approach is advantageous in the sense that it captures the spatial context in which words are used, e.g. to discover words that typically precede or follow another word. In this sense, we can also observe that the bag-of-words model is a special case of the n-gram model, with $n = 1$.

tf-idf While the bag-of-words approach represent textual units in documents by their frequency, this is not necessarily the best approach in determining the importance of a word. Common words that carry little lexical meaning are likely to appear with high frequencies, known within IR as *stop words*, e.g. *the*, *is*, *and* [7]. The rarer terms are likely to be better indicators of the topics addressed in the documents, especially when domain specific words are used, e.g. "inning" which would refer to a concept within baseball or similar sports. A remedy is to add weighting to the terms using the inverse of the document frequency, effectively diminishing the highly frequent words while also increasing the weights of words that appear more rarely. This statistic is known as *tf-idf*, or term frequency-inverse document frequency, which reflects the combination of the two statistics as a product.

The calculations for the two respective statistics can be determined in multiple ways. Term frequency can be the raw count of occurrences as presented in the bag-of-words approach earlier, or use a more sophisticated scheme where the frequency is normalized with regards to the document length, as defined in Rajaraman and Ullman [7]:

$$tf(t, d) = \frac{f_{t,d}}{max_u(f_{u,d})}$$

which is the frequency of term *t* in document *d* divided by the frequency of the most frequent term *u* in document *d*.

The inverse document frequency measure also has its own variations, though the common denominator is the appearance of a logarithmically scaled inverse fraction. The simplest weighthing scheme involves dividing the total number of documents *N* by the number of documents the term *t* appears in, n_t, and then taking the logarithm of that:

$$idf(t) = log(\frac{N}{n_t})$$

One can see from this formula that as a term appears in more documents, the ratio inside the logarithmic function will slowly approximate 1 and the total idf, and thus tf-idf as a whole, will reach 0.

Using these two metrics, tf-idf is calculated by taking the product of them. A term with a high tf-idf weight will therefore be a reasonably good indicator of the topic addressed in the document.

An issue with this approach is the high dimensionality and sparsity of the vectors when working with a large and varied corpus. This is especially apparent when working with Twitter with its issues in terms of language (as discussed back in section 2.2.2).

Feature hashing As an alternative to the bag-of-words (and analogously, tf-idf) approach of vectorizing term features, the feature hashing approach formulated by Weinberger *et al.* [30] attempts to alleviate the dimensionality and sparsity issue by skipping the construction and maintenance of a dictionary. Instead, the feature vectors are built and updated by passing the feature through a hash function, the result of which is directly used as the index in the vector. This results in increased throughput and reduced memory usage, with the trade-off of not being able to transform the vectors back to the input features due to hash functions generally being one-way functions.

Topic Modeling Another form of statistical modeling for analyzing and finding important terms within a set of texts is topic modeling. Generally building on the vectors/matrices outputted from the bag-of-words or tf-idf approaches, a topic model attempts to extract and cluster together words that commonly co-occur in the texts and may be semantically similar. These clusters of similar words then form a *topic*. The model itself opens for documents in a corpus to be inspected and the distribution of topics (and thus important terms) to be discovered. Historically, the most commonly used models include Latent Dirichlet Allocation [31], Non-negative matrix factorization [32], and Latent semantic analysis/indexing [33]. We will not examine these models in greater detail, and rather resort to using these models in a black-box-manner. The interested reader may refer to their respective publications.

Word Embeddings In recent years, the use of artificial neural networks has become an increasingly popular approach. Instead of using one word per dimension as with previously described methods, the concept of word embeddings involves embedding these word into a vector space of significantly reduced dimension [34]. Each unique word is then assigned a vector within this space, effectively forming an embedding space. An advantage

with word embeddings is the language agnostic aspect, in which the model can capture both semantic and syntactic relationships and regularities without the need for external annotation. Similar words will thus be close to each other in the vector space. These vectors can for instance be used as feature inputs for machine learning classifiers, which can then be applied to tasks such as sentiment analysis and named entity recognition.

Mikolov *et al.* [35] present two different unsupervised approaches to compute vector representations of words, which resulted in Google's open-source `word2vec` tool. These two methods both share the architecture of being fully connected neural networks, using only a single hidden layer consisting of linear neurons, and using backpropagation in combination with stochastic gradient descent (SGD) during training. What differs between the two is the underlying model, namely *skip-gram* and *continuous bag-of-words* (CBOW), both with their respective advantages and disadvantages. Skip-gram takes in a single word w_i and attempts to predict its context, i.e. the preceding and following words $(...w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}...)$. Conversely, CBOW uses a given context as input to predict the target word. We refer the interested reader to [36] for a more thorough presentation of these concepts. In terms of performance, Mikolov notes that the skip-gram model works well with smaller amounts of training data, and performs well in representing rare words or phrases. CBOW is faster to train, and performs slightly better on frequent words. This makes sense because creating large amounts of contexts for training using limited data is possible in the case of skip-gram, while CBOW usually needs more data due to depending on contexts for input.

An interesting consequence of the vectors generated is the preservation of linguistic relationships, allowing for algebraic operations on vectors to produce meaningful results [37]. For instance, `vector('Paris') - vector('France') + vector('Italy')` results in a vector representation very close to `vector('Rome')`. Other relationships such as verb tense, male-female, capital-country can also be observed.

In a similar fashion to `word2vec`, Pennington *et al.* [38] demonstrate `GloVe` (Global Vectors for Word Representation) as an alternative approach to word embedding. Both of these models produce embedded word vectors, however `word2vec` is a predictive model while `GloVe` is a count-based model attempting to exploit statistical properties. Essentially, a co-occurrence matrix is built before factorization using SGD to reduce dimensionality, effectively yielding the word embedding vectors.

2.6 Classification

In the context of both sentiment analysis and recommendation, classification is the task of identifying and assigning a new observation or instance (in the case of text, a new document) to a specific category or class. Classification is usually considered as a form of supervised learning for pattern recognition, i.e. the basis for predictions involve using a prior set of training data of observations with their respective classes already known and assigned. In the case of Twitter, a classifier could be used to automatically predict the sentiment of Tweets [39], or to differentiate between spammers and legitimate users [40].

2.6.1 Traditional Approaches

Traditionally, probabilistic algorithms such as Naive Bayes have been used as baseline approaches in classification due to their simplicity in implementation, efficient scaling, as well as reasonable quality of results [41]. The usage of linear models such as Support Vector Machines (SVMs) have also been popular baseline approaches for classification tasks. While SVMs generally have been considered state-of-the-art, some studies have shown Naive Bayes to have comparable performance with appropriate feature engineering, preprocessing and fine-tuning [42], and in some cases even outperforming SVMs [43].

2.6.2 Artificial Neural Networks

Artificial neural networks (ANNs) have been around since the the 1950's with the conception of the perceptron algorithm [44], but never gained much traction until several decades later with the emergence of *deep learning*, though the term today usually just refers to an ANN with more than a single layer.

Basic Concepts Conceptually, an artificial neural network is a computational model within machine learning, consisting of a number of simply, highly connected units called *artificial neurons* - which themselves are simple, mathematical functions mimicking the functions of their biological counterparts. This is inspired by the structure and properties of biological neural systems. Each network is typically organized in layers, with the first and the final layer accepting input and producing outputs, respectively. Any layers between these are referred to as hidden layers, and contain a number of neurons. The neurons in each layer are typically interconnected, depending on the design of the network. Each neuron also has an *activation function* which is a non-linear function defining the output(s) to the following layer given input(s) from the previous layer, effectively adding bias/weights and non-linearity into the neural network itself. Without an activation function the output would just be a linear transformation of the input, effectively turning a "complex" multi-layer neural network into a form of simple linear regression.

Network Types In the simplest form of neural networks the output from a layer is used as an input to the next layer, which is a type of neural networks known as *feed-forward* neural networks. The most well-known example of such an ANN model is the *multilayer perceptron*, popularized by Rumelhart *et al.* [45] with the application of the backpropagation algorithm.

Other notable classes include *convoluational neural networks* (CNNs or ConvNets) and *recurrent neural networks* (RNNs), the latter of which introduce feedback loops within neurons and layers. An early example was the Hopfield Network [46]. A shortcoming of RNNs is the significant amount of time needed to learn long-term information due to decaying error back flow, referred to as the *vanishing gradient problem*. This was later addressed with the introduction of the *Long short-term memory* (LSTM) architecture by Hochreiter and Schmidhuber [47], which introduced units which excels at memorizing values for longer as well as shorter durations of time, which in the context of text is useful for learning sequence dependencies. A variant called bidirectional LSTM increases the input information to the network by splitting neurons into two directions of time, providing

access to both past and future states [48]. *Gated recurrent units* (GRUs) are a fairly recent type of recurrent units introduced by Chung *et al.* [49], also addressing the vanishing gradient problem akin to LSTMs. Studies have found that the performance was comparable to LSTM, however more computationally efficient in terms of the required CPU time for training and convergence depending on the task [50]. GRU was favored in terms of smaller datasets, but otherwise fairly equivalent.

CNNs are variations of multilayer perceptrons, making use of convolution[9] operations to filter the input data for useful information (feature extraction). Largely inspired by the visual cortex in animals CNNs are usually used in tasks where the recognition of patterns across space is learned, such as in image recognition. Combinations of CNNs and RNNs exist, e.g. by applying a CNN followed by an RNN [51].

Properties, Usage and Prominence While the performance of traditional approaches heavily rely on feature engineering, the usage of ANNs attempts to obviate the need for manual feature engineering by learning the features as well as learning how to solve a task using these features - ultimately allowing for a wide range of applicable tasks e.g. recommender systems, natural language processing, or computer vision.

Even though the theory and research of deep learning have been around for a while, the popularity growth of deep learning in recent years are largely owed to multiple factors: the increased availability of data for training, advances in hardware (exploiting the power of graphical processing units, parallelization), as well as better, scalable algorithms to name a few.

[9] https://en.wikipedia.org/wiki/Convolution

Chapter 3

Related Work

This chapter presents, summarizes and compares a selection of closely related works to the topics discussed in this thesis. We will mainly examine those pertaining to movie recommendations in social media in section 3.1. Following this in section 3.2 is a survey of works related to feature extraction for sentiment analysis, which is an important step in eliciting user preferences and opinions for recommendations. Section 3.3 examines the work done with regards to sentiment analysis in the Twitter domain. Finally, in section 3.4, selected works exploring named entity recognition in Twitter are presented.

3.1 Recommendations in Social Networks

Yang *et al.* [52] present a study where they attempt to improve accuracy for the top-k recommendation task by using social networks. They propose two approaches; a matrix factorization model enhanced with social trust features, and a nearest neighbor method that also uses the trust network to enhance the users' neighborhoods. The authors find that the use of social networks can have significant improvements to the recommendation accuracy, especially in the case of cold start users. However, an important difference between the paper and this thesis is the chosen datasets and platforms as premise for recommendation. The paper bases itself on using datasets where users may assign ratings to items. Furthermore, the users may assign trust values to other users' reviews and/or ratings. Thus, the approach assumes that one already has access to explicit ratings and social trust relationships - also forming the basis for further recommendation. In our case however, we aim to utilize the user's Twitter data in its entirety, which will capture the individual user in a more general manner rather than pertaining to captured ratings or sentiments towards specific items or entities. In other words, we do not have any guarantees of the existence of explicit ratings for any given user. Thus, we cannot infer that the usage of the same recommendation methods presented in the paper will be applicable for our case.

Another approach to incorporating social information is explored by Silva *et al.* [53], where they introduce Poisson Matrix Factorization with Content and Social trust information (PoissonMF-CS). Three sources of information are combined in the model, and used

as basis for its recommendations: item textual content, user social network, and user–item interactions, effectively adding social and content features to existing Poisson factorization models. The authors evaluate the model on a dataset containing both content and social side information and find:

> [...] that joint modeling of social and content features using Poisson models improves the recommendations, can have scalable inference and generates more compact latent features.

Again, similarly to the work presented by Yang *et al.* [52], the problem tackled by Silva *et al.* [53] is evaluated using a dataset where user-item interactions or ratings are readily available and quantifiable, as well as being restricted to a specific domain in terms of content and social features. In our case however, the problem of inferring such preferences from Twitter profiles must also be taken into consideration - especially as Twitter is not a specialized platform with explicit, well-defined relationships between users and items. Instead, the data available from a Twitter user will aid in representing the user's preferences in a broader, general range covering many domains. As such, one might observe a stronger effect of homophily between users, i.e. users with similar characteristics or preferences are more likely to have established social connections. Nevertheless, the proposed model is very much relevant to the task of recommendation, and may serve as inspiration for future work although the approaches explored are outside of the scope of this thesis.

Armentano *et al.* [54] present an approach that leverages Twitter as a source for information in terms of recommending movies. They evaluate and use a classifier that determines the opinion and polarity expressed in a Tweet before finally extracting the sentiment of Twitter users with regards to a given movie. This is then used as basis for the recommendation of said movie. The paper explores and evaluates different tokenization, preprocessing techniques and algorithms for building a classification model. For tokenization, they experiment with different combinations of n-grams, and concluded that the use of both unigrams and bigrams led to the best results when combined with SVMs. In terms of preprocessing, they also examine different strategies, e.g. stemming, removal of stopwords, hashtags, URLs, conversion of slang language, feature selection with information gain. They found that there were no significant differences in terms of accuracy when applying the different preprocessing strategies. In terms of classifiers, they explored the use of SVMs, Bayesian Networks and Decision Trees, and conclude that an approach based on SVMs and employing both uni- and bigrams combined with feature selection using information gain provide the best results, with the choices of tokenization and classifier having the largest impact. While the paper only considers the movie domain in terms of recommendation, the authors emphasize the possibility of their findings to be extended or adapted to other domains.

Similarly, Gupta *et al.* [55] also explore the use of social networks as a source for information to produce automatic movie recommendations. They evaluate three different classifying algorithms, but focus on Facebook profile information targeting a specific individual unlike Armentano *et al.* [54] who focused on general recommendations based on public opinion. In particular, Gupta et al. analyze and experiment with the use of personal/social features and characteristics as features for improved recommendations, including gender, location, age, and tastes in book and music. They implement and train

three different algorithms (SVM, K-Means Clustering, and Ranking SVM), attempting to produce scores for each movie given a user. The results from their tests showed that clustering was effective and that stereotypes to a certain extent could be used to predict the attractiveness of a movie. As for using user profile features, a high correlation between a user's tastes in books and music and taste in movies was found. The use of gender and location however did not appear to have any significant impact, other than reducing performance. The authors also note that improvements are possible, for example by incorporating movie specific features from `IMDb`[1], as well as user activity and interests from a user's Facebook profile.

While not pertaining to the movie domain nor Twitter specifically, Zhang and Lei [56] explore the modeling of user interests (both long- and short-term) to recommend content in microblogs. In particular, they explore the social friendship aspect of microblogs, hypothesizing the notion of social tie-strength as a factor in user interests. That is, users who follow the same people will like have similar or common interests. The thesis defines tie-strength between two users as the similarity of their respective interests. Results from their study shows that the usage of tie-strength in combination with other users' interests can improve content recommendations.

In a similar fashion, Chen *et al.* [57] experiment on recommending URLs for Twitter users, looking at three separate dimensions: content sources, topic modeling and social voting. They implement and evaluate 12 different proposed algorithms on real Twitter users. Topic modeling was performed by creating a bag-of-words profile for each user, modeling the interests of a user given the content they have previously created or interacted with. Social voting exploits a user's social circle by giving URLs scores depending on their mentions in the user neighborhood. In essence, a URL mentioned often by followees are more likely to be appealing to a user. Results from the study showed that the use of topic modeling and social voting were especially helpful in providing recommendations and noted that while their approach was general, better performance could be achieved if adapted with domain-specific features.

3.2 Feature Extraction for Sentiment Analysis

Related to recommender systems is the field of sentiment analysis. One is usually interested in recommending items or content similar to the ones a user likes, especially in a social media contexts that enables users to express opinions and sentiments. Finding and extracting features and determining sentiments are thus very relevant tasks.

Hamdan *et al.* [58] presents the implementation of a logistic regression classifier, experimenting with different groups of features: sentiment lexicons, semantic role labeling, topic modeling and Z-score ("a standardization of the term frequency using multi-nomial distribution") among the most noteworthy. Their experiments show that addition of sentiment lexicon features had the largest impact in terms of accuracy, and suggested future work to focus on the automatic construction of such lexicons using Z-scores as a metric for the association between a term and its sentiment label. A possible issue with such an approach is the amount of data needed for the construction, especially so with the highly

[1] `http://www.imdb.com/`

variable and noisy language of Twitter. While unique misspellings or the use of symbols are semantically understandable for humans, the decipherment of such language is a demanding task for machines.

Zhao *et al.* [59] study comparisons of two similar products and propose a system to mine opinions from Twitter in order to extract the product features referred to in said Tweets. Following this, sentiment analysis is performed on the mentioned features, so that user preference to the products can be elicited, as well as enabling the system to explain why a user prefers the product. The authors use a bag-of-words approach with tf-idf for feature extraction, and create a tree structure to represent a hierarchy of concepts. Recommendation is produced by similarity comparisons between user-preferred features with identified products and their associated features. While the study showed good effectiveness in terms of eliciting features for predefined product queries, it is uncertain how useful the approach would be in terms of finding features or concepts related to an individual Twitter user's timeline (the content of which is often highly variable).

3.3 Sentiment Analysis in Twitter

In terms of actually performing sentiment analysis, many novel approaches have been proposed over the years. As mentioned earlier, the use of ANNs has been a popular field of research, also within natural language processing. Analogously to traditional methods of sentiment analysis, the proposed approaches have generally been applied to datasets of relatively formal nature. For instance, Kim [60] concluded that their proposed use of CNNs were on par with or improved upon the (at the time) state-of-the-art approaches in several tasks, including sentence-level sentiment analysis. Similarly, Zhang *et al.* [61] compare character-level (using encoded sequences of characters as inputs) CNNs to traditional models as well as other forms of neural networks in the task of text classification. The results presented show comparable performance, with traditional models having an upper edge on smaller datasets, while the neural networks show their strength when the amount of training samples reach millions.

As for the case of application of ANNs to shorter texts such as those in social media, research in this direction was largely hindered due to the lack of substantial (in terms of size), annotated datasets for training and testing systems. This has changed in recent years however, with the proposed approaches having shown promising results. dos Santos and Gatti [62] present a neural network architecture exploiting character-level features and word embeddings to perform sentiment analysis on short texts, including Tweets. By using both sets of features, the approach achieves slightly improved accuracy (between 2%-4%) compared to established classifier methods when applied to the Stanford Twitter Sentiment dataset. While they use *word2vec* for learning word embeddings and perform training using Wikipedia as a source, using pre-trained, general purpose embeddings such as those available from Stanford's *GloVe* could be an improvement. The pre-trained embeddings are usually trained on even larger datasets, and consequently have larger vocabularies. Embeddings trained using specific sources, such as on Tweets also exist.

The series of Semantic Evaluation[2] (SemEval, organized by SIGLEX, a group within

[2] https://en.wikipedia.org/wiki/SemEval

the Association for Computational Linguistics[3]) workshops have in recent years also included exercises involving Twitter analysis, including the task of sentiment analysis. Specifically, the task of predicting whether a Tweet carries *positive*, *negative*, or *neutral* sentiment is of interest to this thesis. Severyn and Moschitti [63], participated in SemEval 2015 and were among the top performing teams in the Twitter sentiment analysis tasks by using word embeddings with a CNN. The authors highlight the notable improvement of using word embeddings trained on a large amount of Tweets, and an even further improvement with distant supervision with the use of positive emoticons in Tweets as loose labels.

Results from SemEval 2016 [64] and 2017 [65] show a trend of many teams using deep learning techniques, including either general purpose or trained, task-specific word embeddings. These were also often used in combination with deep learning models such as CNNs and LSTM (RNNs) being the most common among the top performing teams. It is evident that the use of neural networks has come to stay in natural language processing, though further research may reveal refinements and improvements.

3.4 Named Entity Recognition in Twitter

Also related to recommender systems and sentiment analysis is the field of named entity recognition (NER), in which we wish to identify and label the named entities referenced in Tweets. While being a well-studied field in traditional text, with tools such as the Stanford NER capable of achieving fairly good performance, but falls short when applied to Tweets. Ratinov and Roth [66] found that the average F1-score of the Stanford NER which is trained on the CoNLL03 shared task data set, drops from 90.8% to 45.8% when applied to Tweets.

Liu *et al.* [67] propose a solution to tackle the challenges of identifying named entities in the context of Twitter. Using a combination of a k-Nearest Neighbor (k-NN) algorithm and a CRF model in a semi-supervised framework, they aim to capture and exploit both the global coarse evidence and the fine-grained evidence within individual Tweets, respectively. The semi-supervised framework also allows the classifier to be retrained using the its confidently labeled outputs. The authors generally found that the inclusion of gazetteer and lexical features for the CRF model significantly improved performance. Additionally, experimenting with the replacement of the k-NN classifier with others such as Maximum Entropy and Support Vector Machines revealed comparable performance, though k-NN was superior in terms of retraining. They also found that CRF performed considerably better than its competitors. Results from their study showed positive effects of using CRF and semi-supervised learning, though remarked that the short and noisy nature of Tweets were problematic, and suggested future development of Tweet normalization (i.e. normalizing the textual content) to alleviate the issue.

Li *et al.* [68] also examine named entity recognition in Tweets, and propose an unsupervised approach exploiting information from web resources to build local and global contexts instead of relying on linguistic features. Essentially the approach is a 2-step approach, the first step using the global context to segment a Tweet, producing candidate named entities. The second step involves using a random walk model to exploit the local

[3] `https://www.aclweb.org/portal/sigs`

context. Results from their study show comparable performance with other state-of-the-art solutions, however does not address the problem of labeling the named entity type. For most applications, including ours, having a named entity without an associated label is not of much use. A solution could be to use a knowledge base to detect and elicit entity types and variations.

Ritter *et al.* [69] address the challenges of Twitter by rebuilding the NLP pipeline, from part-of-speech tagging to chunking and finally named entity recognition. Their study shows that the use of features generated from POS tagging and chunking aid in segmenting named entities, and that their approach in training on Tweets show significant improvement compared to existing state-of-the-art tools trained on traditional corpora. An obvious issue though is the substantial amount manual annotation needed to be effective, as Tweets are highly diverse (in terms of topics as well as language) and contain many different types of entities. Additionally, named entities occur rather infrequently in Tweets, requiring an even larger amount of Tweets to be collected and analyzed.

3.5 Summary

While all of the closest related works examine the task of recommendation in social media, they all tackle slightly different subtasks or have differing premises or assumptions. For example, the works of Yang *et al.* [52] and Silva *et al.* [53] experiment with datasets where explicit ratings for items are readily available. However, for Twitter data or social media networks in general, one cannot make the same assumptions. Thus, the methods suggested by the authors might not be applicable in the domain of Twitter or within the scope of the tasks that this thesis aims to explore. Armentano *et al.* [54] focus on generally recommending a movie given public Tweets about the movie, which is somewhat close to our objective. However, the main objective of our thesis is to examine the possibility of using any arbitrary user's entire Twitter profile to infer a set of movies that user would (hopefully) like. Considering this, Gupta *et al.* [55] appears to have a more similar objective, although pertains to the use of Facebook instead of Twitter. While Facebook users can explicitly list content (e.g. movies, music, people) they like, this information must be either explicitly or implicitly found on a user's Twitter timeline. That is, a user must either have Tweeted about the entity that they like, or must in some way have an implicit connection to the entity so that we can infer their likes. This connection could for instance be a followee related to the entity (or being the entity itself), or someone in their social circle themselves having Tweeted about the entity. Additionally, accessing the needed data from the Facebook API requires explicit permission from the users involved, while user timelines on Twitter are usually open to the public. For our thesis the work of Gupta *et al.* [55] is thus very relevant, though we require more information to be inferred. Using movie specific features from `IMDb` as they suggest is also something we will explore in our approach. The works of Zhang and Lei [56] and Chen *et al.* [57] also have similar objectives, with the exploration of social circles being especially relevant to our work.

Having reviewed the studies regarding the task of recommendation in social media, we see that there is a variety of tasks and approaches explored - ranging from general to specific domains. However, while some are close, none of the works directly address the objective of our thesis: to attempt to recommend an entity or product to a user given their

Twitter profile. Furthermore, most of these are a couple of years old or older. It would be interesting to see new studies utilizing general approaches and improvements devised since then.

The other works mentioned showcase state-of-the-art approaches with regards to other tasks relevant to our thesis, and may prove useful when combined for our experimentation and proposed approach.

Chapter 4

The Two-Step Recommender Approach

This chapter starts with section 4.1, detailing our overall theoretical approach. Section 4.2 describes the experiments and findings with regards to choosing an appropriate sentiment analysis method. The final approach to creating and implementing the system and its modules are detailed in sections 4.3, which also includes the acquisition and details of the data to be used in our system for development and testing. Finally, the approaches and experiments to be performed with the recommender system itself are defined in section 4.4.

4.1 Theoretical Approach

As introduced in section 1.3, the main objective of this thesis is to provide personalized movie recommendations given on a user's Twitter profile. This section describes a theoretical approach for producing recommendations.

4.1.1 Feature Extraction for User Profiles

By having access to a user's timeline on Twitter, one gains insight to the topics and themes that the user may find important or interesting. This information is valuable as it potentially enables a recommender system to elicit user preferences based on the content available from prior Tweets, which can enhance an existing user or act as the basis for a new user in the recommender system - alleviating the common cold start problem in the latter case.

With Twitter, we can also exploit the social networking aspect. A user may share preferences with their followers or followees, or vice versa, representing a possible network effect. As with much of human preferences, this is highly complex and dependent on domain and individual - although the bandwagon effect is very much a real phenomenon in society. Additionally, these effects might be stronger if the users are mutually follow-

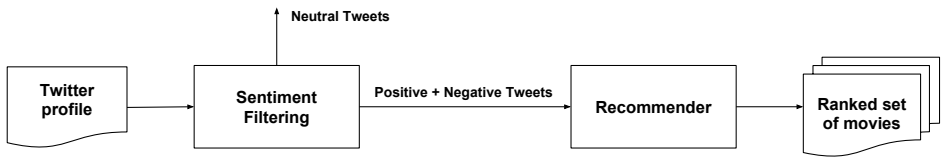


Figure 4.1: The overall general two-step recommender system approach

ing each other, i.e. they are *friends*. Zhang and Lei [56] showed that user satisfaction of accuracy in recommendations were higher when considering the followees' interests. They also suggested using tie-strength (i.e. measuring the similarity between two users' interests), as well as considering common connections' tie-strength of interests.

As discussed earlier, eliciting features for a given user on Twitter might be difficult due to the sparsity and short nature of Tweets. Adding features using data from a user's social circle could help remedy this. A novel system could attempt to detect users with more expressed or explicit opinions. A followee with a profile consisting just Retweets would have less effect on a user than a followee with many self-authored Tweets. Additionally, even stronger weights could be given to followees with high levels of interaction with the user, e.g. Retweeting or liking the user's Tweets, or mentioning the user in their Tweets.

An approach for eliciting user features could be to use the user's (as well as their top-k influential followees in their social circle) Tweets pertaining to or related to any movies, filtering for Tweets with sentiments classified as positive only, using these as basis for creating top-k feature vectors of user preferences. Alternatively, one could also consider using negative Tweets to map user dislikes as well. Multiple Tweets about a particular topic or movie could also imply stronger sentiments, a form of implicit feedback that could be worth looking at.

4.1.2 Feature Extraction for Movie Profiles

What features of a movie can constitute the movie profile to be used for recommendation? Examples include plot summaries, synopses, keywords, cast members, or (aggregated) user reviews - most of which are available on online databases and communities.

For the longer texts, such as the user reviews, one could use topic modeling or tf-idf to elicit the important aspects related to the movie. Using sentiment analysis to cluster the aspects is also a possibility. Effects of decay in the temporal dimension, as well as popularity could also be interesting to examine. Reviews immediately after release may be biased or generally positive due to expectations or hype. Often the most dedicated fans will be early to watch and be vocal about their opinions. With time, a movie might show a downward trend in ratings, although the converse could be true as well.

4.1.3 Two-Step Recommender

With the theoretical approach to feature extraction in mind, we propose our *two-step recommender approach* illustrated in figure 4.1. By taking a Twitter profile and its associated Tweets as input, we perform analysis and matching with regards to a prepared database of products (in our case, movies), and produce a ranked set of results as output.

While other steps and components are involved in the pipeline, we will focus our experiments on the two steps we hypothesize as the main points of interest for our research.

Step 1: Sentiment Filtering Due to the nature of Tweets having content of high variety, it is desirable to keep only the Tweets expressing sentiments and discard those of neutral nature as they do not express any aspects that the user either likes or dislikes. These neutral Tweets would generally have content of mundane or irrelevant nature, for example news reports or advertisements. The rationale behind this step is that by filtering out the neutral information, while the remaining (positive and negative) is expected to be a better representation of the user's preferences.

Step 2: Recommendation With the information filtered for neutral aspects, the recommender system can now match the user's likes with movies containing such aspects, or at least closely related ones. Conversely, the system can also perform matching of negative aspects to assign a factor of dissimilarity. Ideally then, the movies closest to the user likes while having the lowest dissimilarity score will be ranked higher. The actual calculation of similarity is an implementation detail. What we have described here is essentially a content-based recommender system. Listing 1 summarizes the overall approach to recommending a set of movies in pseudocode.

We extend the similarity calculation phase of the recommender by including a Named Entity Recognizer to extract named entities referenced by the user. Analogously to finding user features, we then have to analyze these Tweets to elicit positive and negative aspects related to these entities, as well as the overall expressed sentiment on the entity. Direct matching to the referenced and related entities is then performed, e.g. a user having expressed positive sentiment towards an actor is likely to also be positive to a movie featuring said actor.

The use of features extracted from the user's social circle is also factored in. Rather than using the entire social circle, the approach uses a subset of k followees, extracts features from this set, and adds them to the user's existing feature set for enrichment.

A variation of the recommender method is to use collaborative filtering. Matching is then performed by calculating similarity between a user and all other users, with the requirement that the other users must have expressed sentiments towards at least one movie. This is known as user-based CF, with the intuition being that similar users will like similar movies. Another alternative we examine is item-based CF, given that the user fulfills the same requirement as defined in the case of user-based CF. Rather than finding similar users, we compare movies to find those similar to the one(s) they already like. Finally, a hybrid approach implementing a combination of the aforementioned methods is also a recommender technique we experiment with.

```
    # Create user profile
    for user in users:
        for tweet in user.tweets:
            # Label Tweets by their sentiments
            tweet.sentiment = sentiment(tweet.text)
            # Create user profile by extracting likes and dislikes of user
            if tweet.sentiment == 'positive':
                # user.likes is initially empty
                user.likes += extract_features(user.tweet)
            elif tweet.sentiment == 'negative':
                # user.dislikes is initially empty
                user.dislikes += extract_features(user.tweet)
            # Add features/aspects of social circle
            if include_social_circle_interests:
                # Find top-k most influential followees (based on e.g. interaction)
                top_friends = find_top_friends(max_users=k, user.friends)
                for friend in top_friends:
                    # Extract and add their likes/dislikes to the user's set
                    # Possibly add some form of factoring in
                    # (so that friends' features will not take precedence)
                    user.likes += extract_likes(friend.tweets)
                    user.dislikes += extract_dislikes(friend.tweets)

    # Recommend movies using profiles created
    for user in users:
        # Initialize empty dictionary
        scores = {}
        for movie in movies:
            # Calculate similarity between user and movie
            # Method should factor in for dissimilarities
            # i.e. decrease score if the movie contains any user dislikes
            scores[movie] = similarity(user, movie)
        # Return set of movies and their associated scores
        # (should sort by descending score)
        return scores
```

Listing 1: Pseudocode for a theoretical, content-based recommendation strategy

4.2 Sentiment Analysis

In order to filter Tweets for the first step of our approach, we must first be able to label them by sentiment. Due to the magnitude of data we are working with, it is desirable to perform this task automatically. Thus we choose to experiment with and compare different machine learning classifiers as well as simpler lexicon and/or rule-based tools.

Preliminary work [70] showed that the use of VADER, "a lexicon and rule-based sen-

timent analysis tool that is specifically attuned to sentiments expressed in social media" [71] proved satisfactory, though with certain limitations due to the nature of ambiguous Twitter language. Hutto and Gilbert [71] evaluated their tool against other baseline machine learning models and found that their approach outperformed these in most domains, especially when applied to a Twitter corpus. We will however perform our own evaluation of the tool compared to some of these same baseline classifiers, as well as a selection of neural network models.

4.2.1 Word Representations

As detailed in section 2.5, machine learning classifiers require a representation of raw text to be able to perform tasks. In other words, we need to transform the texts into a machine-readable format - a numerical vector. We implement a method that takes raw text as input, tokenizes it before passing it on to the feature extractors and vectorizers, and finally returns a numerical vector representation. Multiple variations are available here, with the specific methods experimented with described in the following sections.

Tf-idf Vectorizer The baseline approach is to use tf-idf term weights, as mentioned earlier. This approach has been shown to be fairly simple, yet effective in various domains, though might not be as useful in the Twitter setting due to the large possible variety in terms of vocabulary. A possible disadvantage of this method could be the poor scalability with the growing set of documents and vocabulary, as the mapping from term indices to term strings are stored in memory. We will utilize the Python machine learning library `scikit-learn` [72] to transform (preprocess, tokenize) a collection of Tweets into a sparse, tf-idf-weighted document-term matrix.

In terms of specific parameters, we will extract both unigrams and bigrams. The minimum cut-off during vocabulary generation, *min_df* is set to 5, i.e. any terms not appearing in at least 5 documents are ignored. Analogously, the maximum cut-off *max_df* is set to 80%, i.e. any terms appearing in more than 80% of the documents are ignored. The rationale behind these choices is to reduce the vocabulary and ignore terms of low importance e.g. misspelled terms or common stop-word terms. Additionally, we apply normalization using the L^2 norm (also known as an Euclidian norm[1]) to account for documents being of different lengths.

Hashing Vectorizer (HV) Using the feature hashing method described in section 2.5 alleviates the scability problem of the vectorizer method described earlier, though with the downside of not being able to perform inverse transformations on the numerical features to reveal the original strings. Again, we use `scikit-learn` to implement this method, and extract both unigrams and bigrams here as well. This method does not apply tf-idf transformation to the resulting document-term matrix, which opens for the possibility of examining of whether such a transformation has a significant effect when applied to a set of short documents such as Tweets.

[1] `https://en.wikipedia.org/wiki/Norm_(mathematics)#Euclidean_norm`

Table 4.1: Summary of pre-trained word embedding models used in our experiments

Model	Tokens	Vocab	Description
word2vec-GN	∼100 billion	3 million	Trained on Google News dataset
GloVe-6B	∼6 billion	400k	Trained on Wikipedia 2014 + Gigaword 5
GloVe-42B	∼42 billion	1.9 million	Trained on Common Crawl Dataset
GloVe-840B	∼840 billion	2.2 million	Trained on Common Crawl Dataset
GloVe-Twitter	∼27 billion	1.2 million	Trained on 2 billion Tweets
ConceptNet Numberbatch	n/a	426k	Ensemble model, combination of data from ConceptNet, word2vec and GloVe

Word Embeddings While the previous methods produce sparse matrices, it may be desirable to use dense vector representations instead, especially in the case of scalability. As discussed earlier, word embeddings have been shown to be able to capture relationships and regularities in language and represent these aspects in the vector space. A consequence of this is the property of better resilience against noise due to the ability of learning unique words and the context in which they are used. While it is advantageous to train a custom model on data within a specific domain, which in our case is Twitter, we would need substantial amounts of data to train on. Fortunately, there are several pre-trained models available for use, trained using either *word2vec* or *GloVe* on various sources of data.

Using these models, a term can then be transformed into a dense n-dimensional vector representation by lookup on a trained model. The transformation of a document or Tweet of variable length into a single, uniform-length feature vector has multiple approaches. A simple solution would be to use the average of the term vectors, setting out-of-vocabulary terms to a zero vector. Another would be to find the maximum and minimum term vectors and concatenate them, which would result in a feature space twice as large. It is uncertain if this approach leads to loss of information as we essentially discard all words except two, though this is beyond the scope of this thesis. We opt to use the averaged vector to represent a Tweet.

As for the pre-trained models, we will evaluate the performance of 6 different models, summarized in table 4.1. The selection of models vary in terms of underlying model, the amount of tokens/data constituting the training foundation, the resulting vocabulary, as well as the source or domain of the training data. We see that the models from *word2vec*[2], and *GloVe*[3] are trained on a significant amount of data, using various sources. *ConceptNet Numberbatch*[4] [73] takes an interesting approach to the word embedding models by using an ensemble of models as well as linking to a semantic network (ConceptNet[5]). With the authors reporting improved performance compared to some of the other models, the

[2] `https://code.google.com/archive/p/word2vec/`

[3] `https://nlp.stanford.edu/projects/glove/`

[4] `https://github.com/commonsense/conceptnet-numberbatch`

[5] `http://conceptnet.io/`

inclusion of the model in our experiments is warranted.

4.2.2 Classifiers

After passing data through the previous methods, the resulting word representations are ready for the machine learning classifiers for training and evaluation. We will experiment with multiple baseline classifiers, as well as comparing them to variations of neural network classifiers. An overview of the different classifiers can be found in table 4.2.

Table 4.2: Overview of classifiers for sentiment analysis experiments

ID	Classifier	Description
1	VADER	Lexicon and rule-based classifier
2	LinearSVC	SVM with a linear kernel
3	GaussianNB	Gaussian Naive Bayes, compatible with continuous values
4	LogRes	Logistic regression
6	MLP	Multi-layer perceptron neural network
7	CNN	Convolutional neural network
8	CNN-LSTM	Neural networ consisting of a convolutional layer followed by a long short-term memory layer
9	LSTM	Long short-term memory (recurrent neural network)
10	LSTM-bi	Bidirectional LSTM

The VADER classifier (ID 1) is used in a black-box manner, taking raw text as input and returning a sentiment as output. Classifiers 2-4 are our baselines, being the traditionally used machine learning classifiers. A specific choice of the Gaussian Naive Bayes variant is to support the usage of continuous values from the word embeddings. `scikit-learn` provides modules and classes implementing these, with adjustable parameters. For our usage however, we will keep them at their default values. The inputs to these classifiers are either averaged word embedding vectors, or output sparse matrices from the tf-idf or hashing vectorizer.

Classifiers 6-10 are different variations of neural networks. Usage of the `Keras` [74] library will help us implement these networks. All Tweets are first tokenized and vectorized with tf-idf applied, yielding a sequence of word indices where the word of rank i in the dataset (starting at 1) has index i. In order to use the word embeddings with the neural networks, we have to normalize the inputs by reshaping them into the fixed dimensions the network expects. We choose to set the size of sequences to 100, roughly reflecting the mean length of Tweets in our dataset. Sequences shorter than this are padded with zeros at the end, while those longer are truncated.

All of the networks are trained over 2 *epochs*[6] each, with the exception of the MLP network which is trained for 100 epochs. These numbers are chosen to prevent overfitting the training data to the models.

We will now describe the models, and unless elaborated upon, the common layers such as the `Dense`, `Dropout`, or input and output layers have the same function regardless of network. The input numbers beside each layer in the figures indicate the number of neurons

[6] An *epoch* is a single pass or iteration through all of the training examples.

in that layer. `Dense` layers are fully connected (that is, every input neuron is connected to every output neuron) layers of neurons, typically used to transform the dimension of the vectors. The `Dropout` layers represent the regularization technique which randomly disables neurons with a probability p in the hidden layers to prevent overfitting.

MLP The multilayer perceptron network is a simple network with 2 hidden, fully connected layers (the `Dense` layers). Figure 4.2 illustrates the model. It takes in a series of vectors of length 300, matching the dimension of our word embedding models. The final output layer (as for all of the neural networks) has an output of length 3, representing the number of classes for our classification task.

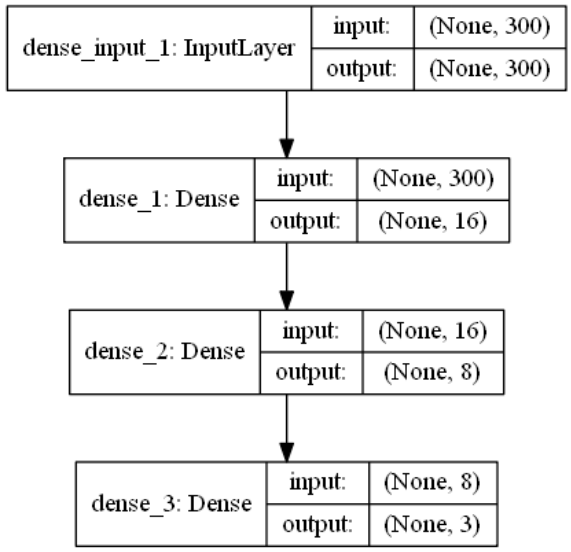


Figure 4.2: The multilayer perceptron network

CNN Illustrated in figure 4.3 is the convolutional neural network. Similarly to the MLP network, the first layer takes in a sequence of length 100. Following this is an `Embedding` layer, effectively embedding the sequences into the word embedding space of the pre-trained models. The convolution operation is then performed, with a max pooling layer to down-sample the data. Recall that the convolution essentially reduces the sequence into smaller features by sampling the surrounding spatially correlated parts (in our case, words). A series of fully connected `Dense` layers with `Dropout` regularization follows before being outputted.

CNN-LSTM Similar to the CNN architecture with an LSTM layer appended, shown in figure 4.4. The `MaxPooling1D` layer reduces the feature size from 96 to 24 (25% of the original size). This pooling layer selects the largest values from each feature map (convoluted feature), producing a subsample of the previous layer - effectively reducing the

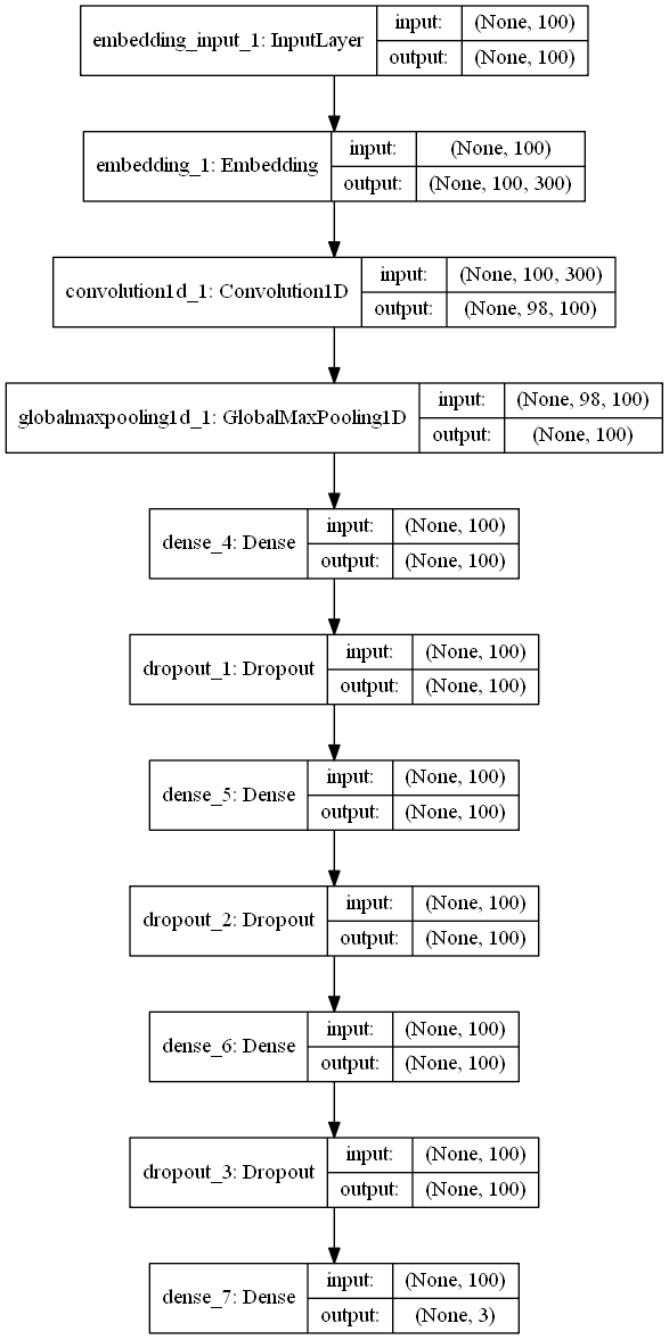


Figure 4.3: The convolutional neural network

dimensionality and outputting a fixed size output matrix. Using the LSTM layer generally aids in encoding and representing the sequence information.

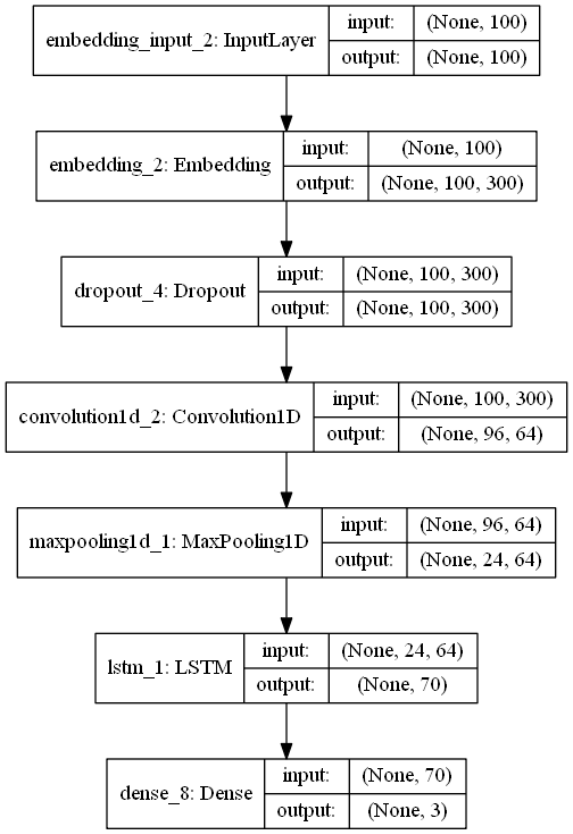


Figure 4.4: The convolutional neural network with a long short-term memory layer appended

LSTM (Recurrent Neural Network) The LSTM model is illustrated in figure 4.5. As Keras is a high-level API, the specific implementation of the long short-term network is not visible, though follows the architecture as defined by Hochreiter and Schmidhuber [47].

LSTM-bi This model is similar to the LSTM model, however, it adds a bidirectional wrapper. See figure 4.6. The wrapper essentially creates a second, LSTM in the other direction which processes the input backwards and outputs the reversed sequence, and merges the output of both layers by concatenation.

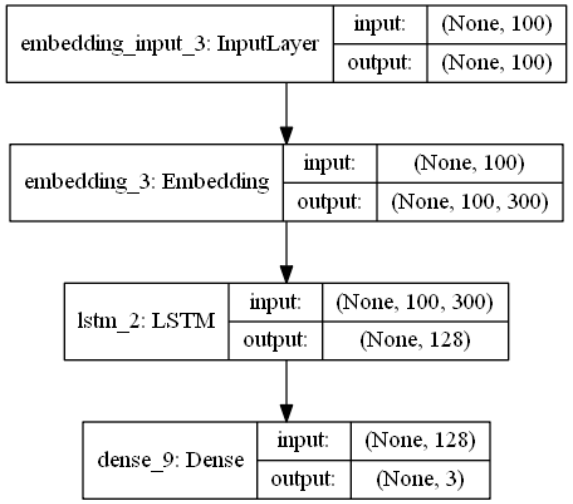


Figure 4.5: The long short-term memory network

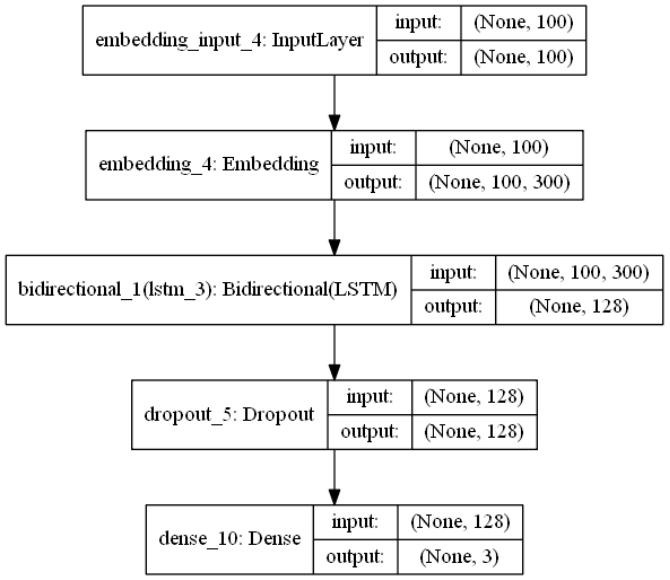


Figure 4.6: The bidirectional long short-term network

4.2.3 Evaluation and Results

Evaluation For evaluation of the sentiment analysis classifiers we use the manually annotated dataset available from SemEval 2017, task 4[7] [65]. This set is comprised of manu-

[7] Available at `http://alt.qcri.org/semeval2017/task4/data/uploads/download.zip`

ally annotated data used in previous years, which after the removal of duplicate and deleted Tweets amount to a total of 40027 Tweets. Of those, 6004 are labeled as `'negative'`, 18201 are `'neutral'`, and 15822 are `'positive'`.

Datasets for SemEval 2018 are also available [75]. However, the related tasks here pertain to regression or ordinal sentiment classification. To limit the scope of this thesis, we choose to quantify the Tweets within the three aforementioned classes in the SemEval 2017 dataset: positive, negative, and neutral.

To prevent overfitting during the assessment of our classifiers, cross-validation will be used. More specifically, we will be using stratified k-fold cross-validation. In essence, the dataset is randomly partitioned into k (in our case, $k = 10$) equally sized subsets, such that the proportion of class samples is representative of the entire dataset. During validation, one of these subsets are used for testing the prediction accuracy of the classifier, while the rest are used for training. The process is repeated for the remaining folds so that every subset is used exactly once for testing. `scikit-learn` provides interfaces to make this process convenient. Predicted test samples along with their real values from each non-overlapping fold are concatenated and returned as a single set of predictions. A classification metric such as the F_1 score (the harmonic mean of precision and recall, see equation 4.1) can then be computed for each label.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.1}$$

where

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}, \text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

As we perform the task of multi-class classification with an imbalanced set of samples, the output F_1-scores for each class are weight-averaged into a single score, taking the class imbalance into account.

Results The results are summarized in table 4.3, showing the weight-averaged F_1-scores for combinations of classifiers and word representations. Highlighted are the best classifiers for each type of word representation where applicable.

Due to long training times and comparable performance to the other classifiers, the LSTM classifiers (9 and 10) were not tested with all of the pre-trained word embedding models, but rather just the best performing ones based on tests with the other classifiers. The neural network classifiers were also not tested with the tf-idf and hashing vectorizers due to issues with compatibility.

From the results we see that the baseline approaches (SVM, NB, LogRes) all perform best with the standard tf-idf and hashing vectorizer representations, though shows comparable performance with certain word embeddings. However, the neural network classifiers, with the exception of the MLP classifier, consistently outperform the baseline approaches - which is a fairly expected outcome.

Three embedding models stand out, namely *word2vec-GoogleNews*, *GloVe-840B*, and *ConceptNet Numberbatch* (CNNB). Though they are very close, the largest model produces the best results. However, *CNNB* performs surprisingly well considering the much

Table 4.3: F_1-scores for combinations of sentiment classifiers and word representations

	Word Representation								
Classifier	**None**	**tf-idf**	**HV**	**w2v-GN**	**GloVe-6B**	**GloVe-42B**	**GloVe-840B**	**GloVe-Twitter**	**CNet Nbatch**
VADER	0.56								
SVM-linear		0.61	**0.62**	0.61	0.58	0.60	0.62	0.59	0.60
GaussianNB		0.56	0.56	0.49	0.48	0.48	0.51	0.49	0.46
LogRes		**0.62**	0.60	0.61	0.58	0.60	0.62	0.59	0.59
MLP				0.61	0.57	0.59	0.61	0.63	0.61
CNN				**0.64**	**0.63**	**0.64**	**0.64**	0.63	**0.65**
CNN-LSTM				**0.64**	0.62	**0.64**	**0.64**	**0.64**	**0.65**
LSTM				0.63			0.63	0.63	**0.65**
LSTM-bi				0.63			0.64	0.64	0.63

smaller vocabulary size, and consistently edges out the other models for the neural network classifiers.

The other three models (*GloVe-6B*, *GloVe-42B*, *GloVe-Twitter*) perform slightly worse, likely due to being trained on fewer samples. The only exception being the model trained on Twitter having the overall best performance, presumably due to being trained on a similar domain.

For comparison, the top scoring system for the equivalent task during SemEval 2016 achieved an F_1 score of 0.633. However, it should be noted that the system was likely evaluated using only the dataset for that year, while in our case we used all the datasets from previous years.

Results from SemEval 2017 [65] show an improvement in scores, with the best performing system achieving an F_1 score of 0.685. The authors also noted the increased usage of deep learning techniques (e.g. CNN and LSTM) by the participating teams compared to previous years, as well as the use of external datasets as lexicons or word embeddings. Nevertheless, our results exhibit state-of-the-art performance in terms of classifying Tweet polarities.

4.3 Final System Overview

Figure 4.7 shows the overall view of the system and pipeline. The data is piped in from either the Search or Stream API, depending on the data needed. Each Tweet is then run through the steps of preprocessing, the output of which will be the basis for recommendation. Most of the system is implemented in Python[8]. Each part will be detailed in the following sections.

4.3.1 Datasets

Obtaining Movie Datasets from IMDb To create a movie profile, we need reliable data. The Internet Movie Database (IMDb) is one of the largest online databases for information

[8] `https://www.python.org/`

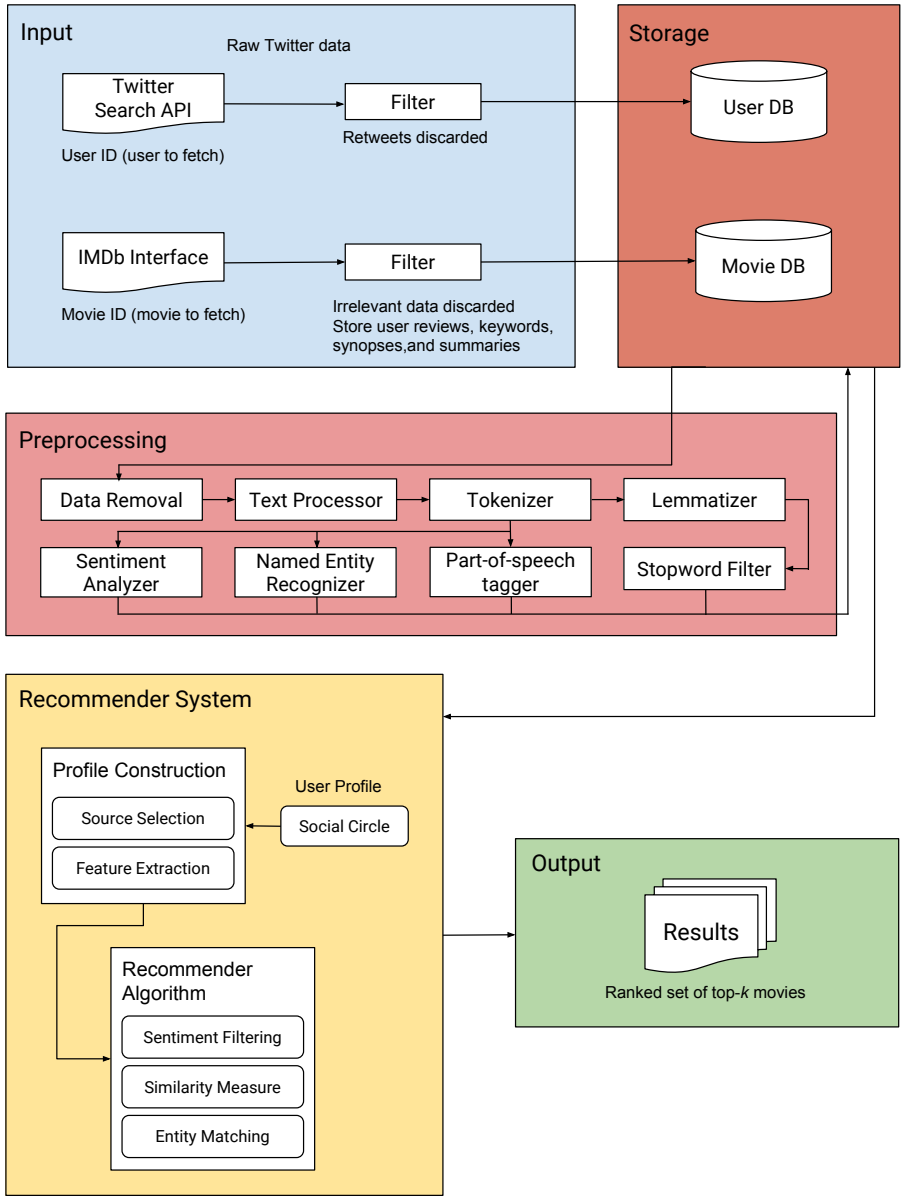


Figure 4.7: The overall view of the system and its components

related to movies and television. While they do not provide an open API for accessing their data, a subset is made available to download from their website[9]. We use data related to the top 250 English movies (as of February 2017), including cast, movie keywords, plot summaries, synopses, and user reviews.

Obtaining and Labeling User Datasets from Twitter Ideally, we would gather a set of genuine Twitter users for usability testing of the system as the output from recommender systems can be measured on multiple subjective qualities, e.g. user satisfaction, serendipity, or diversity. However, due to restrictions with regards to time, the selection of such a set of users is an unattainable task. Therefore, we will evaluate our recommender system automatically using a representative and annotated set gathered from Twitter with associated preference labels assigned manually. This results in the aforementioned qualities being difficult to evaluate, as we do not have access to user feedback with regards to the output associated with that user. We elaborate on the evaluation scheme in section 4.5.

As mentioned in section 2.2.1, the Twitter APIs have certain limitations in terms of extracting data. Generally, for our system, the Search API is used to fetch previously published Tweets matching certain parameters, as well as specific user timelines with accompanying information i.e. users they follow and users they're being followed by. The Streaming API is set up to listen and store incoming Tweets as they are published, also filtered by given parameters. The API is accessed using the Tweepy[10] library, which provides convenience methods for simplified access and error handling.

To be able to evaluate our system, we need annotated data. However, with the challenges and ambiguity of Tweets, the task of automatically and accurately detecting entities or movies a user likes is difficult. Therefore, we manually annotate and create our own set of users for our experiments. After choosing 35 random movies from the collection of movies, we set to find 10 users for each movie having explicitly expressed positive sentiments towards the movie. That is, we manually searched for users using the Twitter Search API for Tweets containing both the name of a movie, as well as the phrases *favorite*, or *love*. These search criteria results in finding users whom definitely like the given movie and have explicitly expressed so, i.e. we expect a user who has Tweeted e.g. "*Inception is my favorite movie of all time*" to find this movie at the top of the results from a recommender system. Two human annotators were involved in the selection of a representative set of users using this methodology, which resulted in a set of 348 users, with all of the users having at least one movie they like, and two of these users having two movies. The users were selected using majority voting between the two annotators, discarding samples receiving diverging or disagreeing votes. A total of 717 566 Tweets were extracted from the profiles.

We also extract information related to each user's social circle for our experiments. Ideally, we would only use a smaller selection of followees with high importance, which in most cases are close friends. This would however require us to download and analyze the profiles of all the followees, which for some users can be up to thousands of users. This is a task that scales poorly especially with our data access to Twitter being restricted. Nevertheless, it could be argued that users only follow other users whom they have some

[9] Information courtesy of IMDb (http://www.imdb.com). Used with permission.

[10] `http://www.tweepy.org/`

form of interest in following and thus would likely have similar interests. Assuming this is true, we therefore randomly choose 10 followees for each user to represent the user's set of friends for our experiments, and extract these users' profiles the same way we extracted the annotated set of users. The amount of users for this set totals to 3301 users, a ways from the theoretical 3500 unique followees due to some users having overlapping followees. Nevertheless, the set of Tweets associated with these followees amount to a total of 5 118 208 Tweets.

```
{
  "text": "OMG DOCTOR STRANGE WAS SO GOOD",
  "created_at": "Fri Nov 11 15:47:46 +0000 2016",
  "user": {
    "id_str": "2330335434",
    "created_at": "Thu Feb 06 14:01:08 +0000 2014",
    "time_zone": "Singapore",
    "utc_offset": 28800,
    "friends_count": 60,
    "followers_count": 81
  },
  "favorite_count": 0,
  "retweet_count": 0,
  "in_reply_to_status_id_str": null,
  "in_reply_to_user_id_str": null,
  "geo": null,
  "coordinates": null,
  "entities": {
    "hashtags": [],
    "symbols": [],
    "urls": [],
    "user_mentions": []
  }
}
```

Listing 2: Truncated sample Tweet returned from the Twitter API

Data Format The data returned from either Twitter API are formatted using JSON. A truncated example Tweet can be seen in listing 2. There is a substantial amount of data sent and associated with each Tweet, and the truncated example lists the most interesting ones. Obviously we are most interested in the `text` property containing the actual Tweet update. However, other properties such as `created_at`, `geo`, `coordinates` and `in_reply_to_user_id_str` can potentially provide interesting data for analysis and insight in terms of temporal, spatial, or social dimensions.

4.3.2 Preprocessing

After the raw Tweet is fetched from Twitter, categorized according to its associated user and stored, it is ready for the preprocessing module. The collection of Tweets for each user is loaded sequentially, and every single Tweet goes through processing. First it is stripped of any unnecessary or irrelevant data, before being run through a natural language processing pipeline before being stored in the database. This pipeline will be detailed in the following subsections.

Text Processing The Tweet text itself is first stripped of any URLs and username mentions as these are assumed to not be important for analyzing the text. Problems with using hashtags and converting them to regular text were elaborated on back in section 2.2.2. We will however attempt to use hashtags by removing the hashtag character, and splitting the compounded words if they exist using regular expressions, though this only works tags where the combination of terms are delimited with capital letters.

The Python library Textacy[11] is then used to clean and normalize the text. Multiple spaces are replaced by a single one, and the text is also stripped of leading and trailing whitespace. Failing Unicode is fixed, and English contractions are replaced by their unshortened forms. This text will be the basis for the subsequent processing steps, and will be referred to as `text_filtered`.

The normalized text is then tokenized using the spaCy[12], a high performance Python library for Natural Language Processing. Its tokenization standards are based on the OntoNotes 5 corpus[13]. Each token is lemmatized and converted to lowercase. Lemmatization data is taken from WordNet[14]. Any token with a match in a specified set of stopwords or symbols are discarded. The set of stopwords include the ones defined in the NLTK [76] and scikit-learn [72] libraries, as well as our own set of defined stopwords for the movie domain specifically. These include words such as `movie, director, premiere, theatre, review, starring` which bear no significant meaning or importance with regards to features or statistics, at least for our purposes.

The output here is stored as `text_preprocessed`. For the purpose of feature extraction later we also store a copy of the list of tokens where only tokens with the tag `NOUN` are kept, stored as `text_nouns`.

As for the data related to movies, not much preprocessing is required. Only the user reviews are sent through the text processing and sentiment analysis labeling steps similarly to the Tweets as described previously, while the synopses and summaries are text processed only.

Named Entity Extraction Extraction of the named entities from `text_filtered` is performed using spaCy, implementing a custom algorithm for entity recognition, the details of which are described by the author on GitHub[15]. Essentially it uses a greedy linear model with weights learned using an implementation of the Averaged Perceptron algorithm as

[11] `https://github.com/chartbeat-labs/textacy`
[12] `https://spacy.io/`
[13] `https://spacy.io/docs/api/annotation#tokenization`
[14] `https://spacy.io/docs/api/annotation#lemmatization`
[15] `https://github.com/explosion/spaCy/issues/491#issuecomment-245702851`

originally described by Collins [77].

The model is trained on the OntoNotes 5[16] corpus consisting of data from a variety of sources with accompanying annotations (word sense disambiguation, with some connected to an ontology and coreference). There are approximately 1.3 million English words in the corpus with sources from news, broadcasts, conversations, and the Web.

Jiang *et al.* [78] compares the named entity recognition models provided in spaCy as well as other publicly available tools (e.g. Stanford's CoreNLP[17], NLTK), showing that spaCy's model was the 2nd best performing model when evaluated against a set of Wikipedia articles in CoNLL format (IOB). Table 4.4 shows the results of their evaluation.

Table 4.4: Comparison of named entity recognition tools, from Jiang *et al.* [78]

System	Precision	Recall	F-measure
spaCy	0.724	0.6514	0.6858
CoreNLP	0.7914	0.7327	0.7609
NLTK	0.5136	0.6532	0.575
LingPipe	0.5412	0.5357	0.5384

We also discard recognized entities with tags pertaining to temporal or numerical types as these have no value for our purposes, specifically the tags `'TIME'`, `'CARDINAL'`, `'DATE'`, `'PERCENT'`, `'MONEY'`, `'QUANTITY'`, and `'ORDINAL'`.

Part-of-Speech Tagging spaCy is also used to assign part-of-speech tags on `text_filtered`, and are then stored as `text_pos_tags`. The tagger[18] implemented in spaCy uses a greedy Averaged Perceptron trained on the OntoNotes 5 corpus, similarly to the Named Entity Recognizer described earlier. It is mapped to the Universal Dependencies[19] tag set. The part-of-speech tags are useful for filtering out words matching certain tags. While our system only uses the tags to filter for nouns with regards to feature extraction, the possibility of using other tags for other purposes exist, e.g. finding commonly used adjectives in the dataset as a means of describing public opinion or sentiment.

Sentiment Analysis Based on the results presented in 4.2, we choose to use one of the neural network classifiers with the `ConceptNet Numberbatch` (CNNB) word embedding model. As there were no significant differences between the classifiers in terms of accuracy, we select the fastest (in terms of training, prediction times are similar), namely the *convolutional neural network* (CNN). The network is trained on the entirety of the SemEval dataset used for testing, and is stored for fast predictions. Labeling and predictions are performed on the `text_filtered` data, which does include stopwords. The reasoning behind this is to keep any contextual clues that the neural network can use due to it being trained on complete Tweets, which in turn leads to improved predictions.

[16] `https://catalog.ldc.upenn.edu/LDC2013T19`
[17] `http://stanfordnlp.github.io/CoreNLP/`
[18] `https://spacy.io/docs/api/annotation#pos-tagging`
[19] `http://universaldependencies.org/u/pos/`

4.4 Recommender System

This section details the experimental setup for the recommender step, following the labeled data outputted from the sentiment analysis step in the preprocessing module. We describe the various possibilities and selection of modules in which our experiments and results will be based on.

4.4.1 Feature Extraction for Profile Construction

While we can use features such as genre or named entities in our construction of user and movie profiles, the actual text content of the Tweets are of high interest as they have potential to represent a user's variety in preferences. To extract features (i.e. important terms, keywords) from the collection of Tweets, we wish to compare the use of tf-idf and topic modeling. Analogously, we do the same experiments for the extraction of movie features from the collection of user reviews, synopses, and/or summaries. These features can also be clustered into groups of positive and negative associations using the results from the sentiment analysis step as mentioned earlier.

Tf-idf The tf-idf transformation is fairly straight forward, and is set up like the tf-idf vectorizer described in section 4.2. We use the entire collection of raw Tweet texts, tokenize the texts, apply tf-idf transformation, and output a sorted list of (term, weight)-pairs. We further restrict this set to only return the top 100 features.

Topic modeling Alternatively, after the tf-idf transformation, we can use topic modeling to discover topics and commonly found terms used within said topics. `scikit-learn` is used to implement the latent Dirichlet allocation[20] model, which we use to detect up to 10 different topics with the top 100 (term, weight)-pairs for each topic. A set of the non-overlapping terms is constructed, and again a list of the top 100 terms sorted by weight is returned.

Social Circles To increase the feature space, we can extract and use the aforementioned features from the user's followees' profiles. These features can be weighted based on e.g. interaction or similarity between the user and the followee, though these weights are not implemented for our system. The features extracted from the followees are added to the user's profile, with no specific weighting applied, i.e. features from both the user and the followee have the same importance, and overlapping features have their scores averaged.

Part-of-Speech Tagging As an additional step to the previous methods, we can use part-of-speech tagging to filter for candidate words to represent a user or movie. In our case we wish to experiment with the use of only words tagged as `NOUN` (i.e. `text_nouns`, the list of tokens we stored during the preprocessing step) to restrict the candidate set of features. This does however rely on the quality of the POS-tagger mentioned earlier.

[20] `https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation`

4.4.2 Similarity Measures

To produce recommendations, we must define some form of method to compare the user and movie profiles. For features like named entities extracted from a user's Tweets, we can do a comparison with the entities in the movie profile to check their existence. Matches can be added to construct a similarity score if the named entities referenced are associated with positive sentiments expressed by the user. Conversely, a negative factor can be added for named entities with negative association. Similarly, matching can also be performed between users, or between movies.

As for the comparison of text features extracted previously, we can exploit the use of word embeddings to calculate similarity. A user or movie profile can then be represented using a single vector, the creation of which can be done by averaging word vectors for the set of terms. Cosine similarity can then be performed between any two vectors. Ideally then, a user will like a movie where their vectors have high similarity. Equivalently, two movies with similar content or features will have vectors whose similarity is high.

Equally to the sentiment analysis step, we can also compare the pre-trained word embeddings here. However, initial testing exhibited similar results to the ones found in section 4.2: while some models were consistently worse, the difference between the top performing models were miniscule.

Another potential measure of similarity is the word mover's distance (WMD) introduced by Kusner *et al.* [79]. This approach exploits word embeddings to measure similarity between any two documents by comparing the amount of distance required to move (in terms of Euclidean distance in the embedding space) from the terms in one document to the ones in the other document, taking into account terms with similar semantics even though the documents do not share any exact terms. The calculation of this distance is then interpreted as an instance of the well-studied earth mover's distance[21] transportation problem. For our system we use the `gensim` [80] library which provides a readily available implementation of WMD.

4.4.3 Recommender Techniques

In terms of recommending movies to users, we have multiple approaches. While we briefly described some of these previously in section 4.1, the approaches shall be reiterated and further detailed next.

Content-based We compare the profiles of users and movies directly, and calculate similarity between the two. The approach is the same as the one defined in listing 1.

User-based collaborative filtering Let the set of users $U_l = \{u \mid l(u) > 0\}$, where $l(u)$ is the amount of movies that a user u likes, and let U be the set of all users. For any user $u \in U$, calculate similarity between u and all users u_i where $u_i \in U_l, u \neq u_i$. Then, add the movies liked by u_i, as well as the similarity score between u and u_i to produce a ranked set of (movie, score)-pairs.

[21] `https://en.wikipedia.org/wiki/Earth_mover%27s_distance`

Item-based collaborative filtering Again, we use the set of users U_l as defined earlier. Let M be the set of movies. For any user $u_i \in U_l$, there exists at least one movie $m_j \in M$ which u_i likes. Then, for all $m_k \in M, m_k \neq m_j$, calculate similarity scores between m_j and m_k. The movies with their associated similarity scores are added to the set of ranked results for the user u_i.

Hybrid As a combination of the user- and item-based collaborative filtering techniques, our hybrid approach first uses the results outputted from the user-based CF as foundation, of which the set of movies in the results are defined as M_u. Building on top of this, we apply the item-based collaborative filtering for each movie $m_l \in M_u$, where M_u is sorted by score in descending order. This effectively expands the existing result set by adding (movie, score)-pairs produced by the item-based CF. If the set already contains a movie, the existing and new scores are weight-averaged, with the existing score having a weight of 0.9 and the other score with 0.1. The reasoning here is that the first round of expansions are added from the item-based CF using the most similar user. As the similarity of users decrease with each round of expansion, the importance of their movies scores are also reduced accordingly.

4.5 Evaluation

The assessment of a recommender system is a difficult task. Results from the system are curated for each user, and thus the quality of results are subjective to the user in terms of multiple factors. Ideally, they should capture and reflect the wide range of a user's preferences, while still providing serendipitous suggestions. Evaluation of such qualitative factors requires us to have a set of real users to test the system on and inquire in terms of user satisfaction. Unfortunately, we do not have a means of receiving feedback from our set of users for evaluation.

However, to quantify the performance of our system, we can use the "liked movie" label assigned to each user. Theoretically, given that the system returns a ranked top-*k* set of movies for a user, the user's liked movie(s) should be among the top results. A form of *recall* can then be measured, i.e. the amount of relevant items (which in our case is either one or two) which are selected and returned in the set. Precision on the other hand cannot be measured reliably, as we do not know how many items of the returned set are truly relevant to the user. The selection of *k* is thus critical for the evaluation: too large and we end up with too many results, too small and the label is less likely to appear.

We select *k* to be 20, which we believe is a fair number of results to return. The metric for performance assessment is defined to be the existence of the label within the ranked set of movies for each user, i.e. 1 if true and 0 if false, divided by the number of users in our dataset. In other words, for each user, if the liked movie label for the user exists in the predicted recommender set for that user, the "accuracy" or hit ratio would by our definition be 1.0 (i.e. 100%). While this is not an ideal measure to evaluate our system, it does provide some indication of how well the system performs, at least in terms of recognizing a movie that we know a user likes. Equation 4.2 summarizes the evaluation metric used.

$$\frac{\sum_{u \in N}^{N} f(u)}{|N|} \tag{4.2}$$

where *N* is the set of users, and

$$f(u) = \begin{cases} 1 & \text{if the user labels for } u \text{ are found in the set of top-20 returned results} \\ 0 & \text{otherwise} \end{cases}$$

4.6 Summary of Experimental Setup

We now summarize the different variables laying the foundations for our experimental setup.

Sentiment Filtering There are four possible choices for the sentiment filtering: no filtering, filtering for likes only, dislikes only, or both likes and dislikes.

User Profile The basis for feature extraction can be either `text_filtered`, `text_preprocessed`, or the part-of-speech tag filtered `text_nouns`.

Movie Profile This is equivalent to User Profile when the user reviews are the source for feature extraction. However, in cases where the sources are either keywords, synopses or summaries, the sentiment labeling is disabled, due to these sources being objective descriptions of the movie. Combinations of sources are also possible options.

Feature Extraction Feature extraction can be applied after the sources for the profiles are selected. Choices here are the *tf-idf* or the *topic modeling* approaches.

Social Circles For the user profiles, we can include their followees' features, with the sentiment filtering options applicable here as well. That is, we can choose to only include the followees' likes and/or dislikes, or all the features with no filtering.

Similarity Measures After extracting the features, we can use word mover's distance or alternatively cosine similarity to calculate similarities between any two sets of features. In the latter case, the set of features must be converted to an averaged term vector. Additionally, we use named entity recognition to extract positively as well as negatively associated entities for the users. We match these entities with the movie cast, and accordingly assign a positive or negative factor to the similarity score. Moreover, we also experiment with the top three pre-trained word embedding models found during the sentiment analysis experiments, i.e. the *ConceptNet Numberbatch*, *GloVe-840B*, and *GloVe-Twitter* models.

Recommender Techniques The four different recommender algorithms (content-based, item-based CF, user-based CF, hybrid) are tested and evaluated with the various combinations of the variables described earlier. Additionally, for each combination of recommender and variables, we also measure the impact of the sentiment filtering options.

Chapter 5

Results and Discussion

In this chapter the different results outputted from the system are showcased and discussed. In particular, we will be examining the output after the preprocessing in section 5.1. Following this is section 5.2, where we evaluate the results of our experiments with the recommender system. Finally, in section 5.3, we summarize and discuss the significance of our results.

5.1 Preprocessing

5.1.1 Output

A truncated example of the results after the preprocessing can be seen in listing 3, the results of which are discussed in the following subsections.

5.1.2 Text Processing

Looking at `text_preprocessed` compared to the original `text`, we see that all the steps in the text processing pipeline have been applied. The text is now a list of lowercased, lemmatized tokens (e.g. from `don't` to `do not`, and the hashtags have been removed as well as the punctuation. Stopwords have also been removed (e.g. `with, is, a, top, all, time, not, me`). These steps are all important for the feature extraction, as we want to discover and cluster important terms together, even though they have minor variances, e.g. `Gladiator, GLADIATOR!, gladiator, gladiator!`

`text_nouns` has `Gladiator` as the only entry, as `movie` and `time` are filtered stopwords. While this describes the Tweet's overall topic well, a fair amount of contextual information is lost during the filtering. `text_preprocessed` is similar, though with the inclusion of `Russel` and `Crowe`. The overall impact of the loss of information is uncertain, and will be subject to experimentation in our recommender system later, though one could hypothesize that having these keywords as well as the associated sentiment label would be sufficient and that additional information (at least in this case) would increase noise.

```
{
  "id": "828852778885750784",
  "named_entities": [{"label": "PERSON", "text": "Russel Crowe"}],
  "sentiment_label": "positive",
  "text": "Gladiator with Russel Crowe is a top 5 all time movie don't @ me",
  "text_filtered": "Gladiator with Russel Crowe is a top 5 all time movie do not me",
  "text_preprocessed": [
    "gladiator",
    "russel",
    "crowe"
  ],
  "text_nouns": [
    "gladiator"
  ],
  "text_pos_tags": [
    [
      ["Gladiator", "NOUN"], ["with", "ADP"], ["Russel", "PROPN"], ["Crowe", "PROPN"],
    ↪  ["is","VERB"], ["a", "DET"], ["top", "ADJ"], ["5", "NUM"], ["all", "DET"], ["time",
    ↪  "NOUN"], ["movie", "NOUN"], ["do", "VERB"], ["not", "ADV"], ["me", "PRON"]
    ]
  ]
}
```


Listing 3: Truncated example Tweet after preprocessing

5.1.3 Part-of-Speech Tagging

The part-of-speech tagger does a fair job in classifying all the tokens, although it labels `Gladiator` as a noun when in this context it refers to the proper noun referencing the movie starring `Russell Crowe`. Such contextual and domain specific information is difficult for a pre-trained model to recognize and utilize. Similar edge cases appear when working with text of informal nature, especially in social media such as Twitter. A possible improvement would be to train up a model on a social media corpus so that such cases are detected, although it would require either a significant amount of annotated data or advances in technology in order to automatically detect and understand such implied or incomplete sentences.

5.1.4 Named Entity Extraction

As seen in the `named_entities` field, the system correctly recognizes `Russell Crowe` as an entity as well as providing the correct associated labeling. Though as mentioned previously, it fails to recognize `Gladiator` as an entity due to the ambiguity of the word in this case, in which it may be interpreted as either a gladiator from ancient Rome or as the name of the popular movie. While it is trivial for humans to deduce the correct interpretation using the contextual information, machines generally have more difficulties with this. In other instances, typically when the word does not appear at the start of a sentence and is properly capitalized, the method correctly recognizes and labels it as a proper noun. This shows that the recognition and accuracy of the Named Entity Recognition method highly depends on the input Tweets. Analogous to the part-of-speech tagger, improvements could be made with more training.

5.1.5 Sentiment Analysis

In the case of sentiment analysis, we see in the `sentiment_label` field that the neural network has assigned the Tweet a `positive` label, which for this case is correct. The original text ends with *"don't @ me"*, which is a somewhat common Twitter phrase used when the original author has made an adamant statement, disregarding opinions or mentions from other users with regards to said statement. This demonstrates the power of using a trained classifier for sentiment analysis. While a rule-based sentiment classifier may have assigned the entire Tweet a negative factor due to phrase containing a negative adverb (i.e. *not*), the trained classifier has learned that such phrases usually do not correlate with the associated sentiment of the phrase it follows.

As with the Named Entity Recognition method, there are cases where the sentiment analysis method has provided erroneous sentiment ratings. Two examples of this are shown in listing 4. In the first Tweet we see the system having classified the Tweet as neutral, though closer inspection of the text reveals that the Tweet itself actually expresses a negative sentiment as it compares the movie to `Green Lantern`, a movie that received generally unfavorable reviews[1]. Additionally, the author of the Tweet uses the phrase *save your cash*, a somewhat ambiguous phrase implying that the movie was not worth the

[1] `https://en.wikipedia.org/wiki/Green_Lantern_(film)#Reception_2`

money spent on the admission ticket. It is worth noting that this example is a fairly difficult sentence to analyze with very domain specific references and ambiguous language.

```
[
    {
        "sentiment_label": "neutral",
        "text": "Just saw #DoctorStrange save your cash! Except for 3D & effects,
 ↪  DoctorStrange is the Green Lantern of Marvel movies! Review coming soon!"
    },
    {
        "sentiment_label": "positive",
        "text": "Much to do today and I'm so glad because it will keep my mind off of the
 ↪  inauguration of the worst. #boycotttheinauguration"
    }
]
```

Listing 4: Examples of Tweets with erroneous sentiment ratings

The second Tweet showcases another case of ambiguity. While the first part of the sentence ("*Much to do today and I'm so glad*") can be interpreted as positive, the second part ("*because it will keep my mind off of the inauguration of the worst*") involves the main topic of the Tweet, as well as implicitly expressing negative sentiments towards the subject to be inaugurated by referring to him as "*the worst*". The neural network sentiment classifier however believes the Tweet expresses an overall positive sentiment. A human annotator would likely classify it as negative.

Unfortunately, such edge cases are difficult to identify and classify, and the only remedy is to adapt and create new rules or provide annotations on similar cases for training. Nevertheless, our classifier performs adequately in most cases. The actual impact of this sentiment labeling in terms of filtering for our recommender system will be explored in the following section.

5.2 Recommender System

5.2.1 Results

In section 4.4 we defined the different variables to experiment with in our recommender system. As it would take an inordinate amount of time to test every single combination of those variables, we rather start with a base set of selections and gradually changed variables while attempting to maximize the accuracy. The tested combinations are shown in table 5.1.

For each of these combinations, we calculate the impact of sentiment filtering. We either include all the Tweets or reviews for the user and movie profiles respectively, and filter by either positive label, negative label, or a combination of both. Analogously, the filtering also affects entity matching by matching all entities, positively associated entities, negatively associated entities, or both positive and negative. We elaborate upon and

Table 5.1: Variable combinations for the recommender system experiments

	Variables					
ID	**User Profile**	**Movie Profile**	**Feature Extraction**	**Social Circles**	**Similarity Measures**	**Recommender Technique**
1	text_preprocessed	reviews	tfidf	no	cosine + no entities + CNNB	content-based
2	"	summaries	"	"	"	"
3	"	keywords	"	"	"	"
4	"	summaries + synopses	"	"	"	"
5	"	summaries + synopses + keywords	"	"	"	"
6	"	summaries + synopses	"	"	word mover's + no entities + CNNB	"
7	"	"	"	yes	cosine + no entities + CNNB	"
8	text_nouns	"	"	"	"	"
9	text_filtered	"	"	"	"	"
10	text_preprocessed	"	"	"	cosine + no entities + GloVe-840B	"
11	"	"	"	"	cosine + no entities + GloVe-Twitter	"
12	"	"	topic modeling	"	"	"
13	"	"	tfidf	"	cosine + entity matching + CNNB	"
14	"	"	"	"	"	user-based CF
15	"	"	"	"	"	item-based CF
16	"	"	"	"	"	hybrid

discuss the results in the following subsections, which showcase the accuracy scores for each combination listed in table 5.1. Calculation of these accuracy scores was described back in section 4.5. The results presented are the averaged accuracy scores of 10 runs for each combination. The process of defining an evaluation metric for assessing the recommender system performance was outlined in section 4.5. We will now recall and apply the definition of accuracy as defined in equation 4.2.

5.2.2 Movie Profile

Initially, our experiments begin with the base combination, i.e. the combination with ID 1. The sources for movie profile generation in this combination are the user reviews extracted earlier. For combinations 2-6 we then vary the sources used, and measure their impact on our accuracy scores. Figure 5.1 shows and compares the scores for each of the different sources and combinations thereof. Testing with synopses only was not possible due to some movies missing this entry. The combination of summaries and synopses (ID 4) yields the best scores. Using only keywords result in fairly poor accuracy scores, likely due to the loss or lack of information compared to the other sources. Interestingly enough, the addition of keywords to the combination of summaries and synopses (ID 5) results in reduced performance, which indicates that the keywords may be inaccurate or less suitable as features.

As for the user reviews (ID 1), these perform better than the summaries alone, yet proves to be inadequate when compared to ID 4. It is worth noting that in the case of positive sentiment filtering, the user reviews do perform better. This is likely due to the generally positive sentiments associated with these reviews as the movies they concern are among the top rated of all time. We can also observe how the sentiment filtering have no positive impact for the other sources, due to these generally being objective descriptions of a given movie while user reviews are subjective most of the times.

Nevertheless, we choose the combination of summaries and synopses as the best source for our movie profiles for the rest of the experiments.

5.2.3 Similarity Measures

Building further on the best combination found previously (ID 4), we now compare the use of cosine similarity and word mover's similarity (WMS, ID 6), the results of which are shown in figure 5.2. Word mover's similarity generally edges out the cosine similarity measure, however the difference is insubstantial when considering that processing times are increased by a factor of hundreds. Processing takes mere minutes when using cosine similarity, while taking hours when using the WMS. This may be due to library used for word mover's distance calculation being inefficient, however the slight improvement in accuracy is not worth the increase in time complexity.

An interesting observation is the sentiment filtering having a larger effect on the WMS measure compared to the cosine measure, especially for the positive terms. This could indicate that the pre-trained word embedding model used for vector creation and consequently cosine similarity is less fit for capturing positive terms, but rather performs better on fairly neutral sources of text.

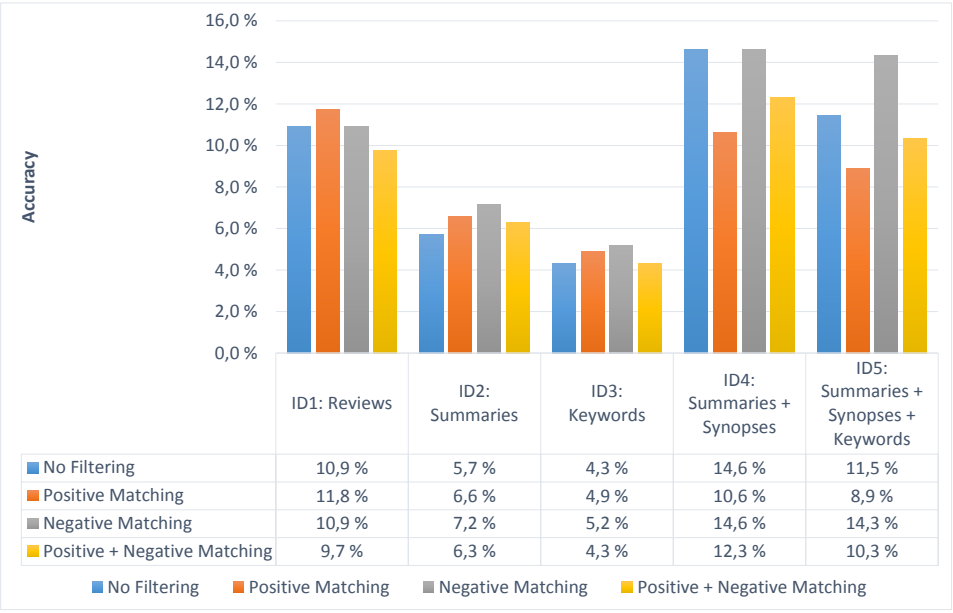


Figure 5.1: Comparison of results for variations of sources for movie profiles

Due to the comparable performance and issues with time complexity, we continue with ID 4 as our preferred combination.

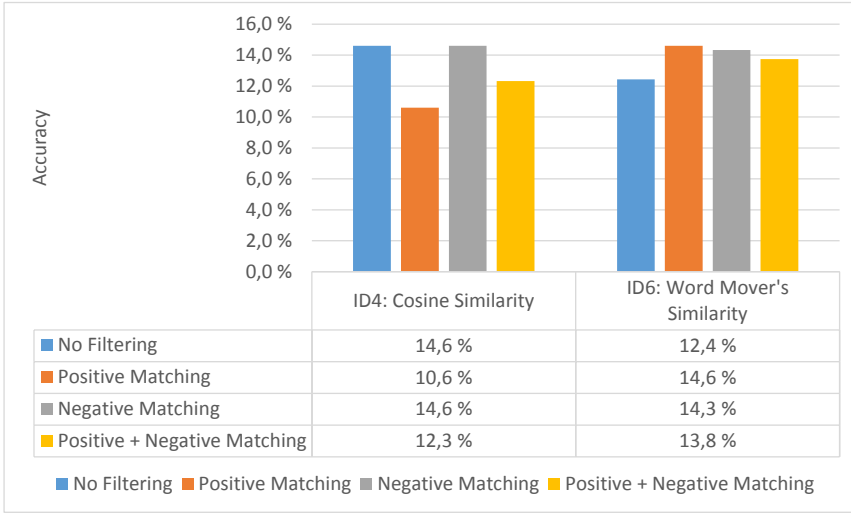
5.2.4 Social Circles

Figure 5.3 illustrates the difference in results when adding features from a user's social circle to said user's profile. We see an increase in accuracy scores across the board when including the social circle. While we hypothesized this outcome earlier, it is still an interesting result as the user's social circle in our case is defined to be any 10 randomly selected followees. If the social circle was found using a more novel approach, we would likely observe even better results. A possible approach could be to measure the strength of connection between the user and each followee, e.g. the amount of interaction (likes, retweets, -mentions), and find the most influential or close followees for that user.

In our case however, the random selection of followees to constitute a user's social circle appears to be effective enough, and the addition of features from said social circle has been shown to be an improvement at the very least. Thus, the addition of social circles in our combination serves as the preferred one for the coming experiments.

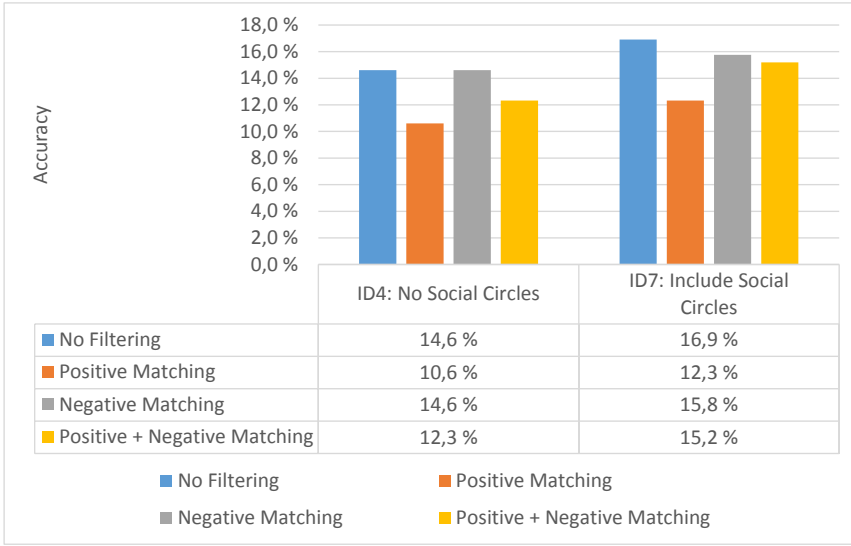
5.2.5 User Profile

As for the user profiles, our initial baseline combination uses the preprocessed text as the source. The other two possibilities is to either use the filtered text (ID 8), which only removes URLs, hashtags and other symbols - closely resembling the original texts in the



	ID4: Cosine Similarity	ID6: Word Mover's Similarity
No Filtering	14,6 %	12,4 %
Positive Matching	10,6 %	14,6 %
Negative Matching	14,6 %	14,3 %
Positive + Negative Matching	12,3 %	13,8 %

Figure 5.2: Comparison of results for variations of similarity measures



	ID4: No Social Circles	ID7: Include Social Circles
No Filtering	14,6 %	16,9 %
Positive Matching	10,6 %	12,3 %
Negative Matching	14,6 %	15,8 %
Positive + Negative Matching	12,3 %	15,2 %

Figure 5.3: Comparison of results when including features from the social circle

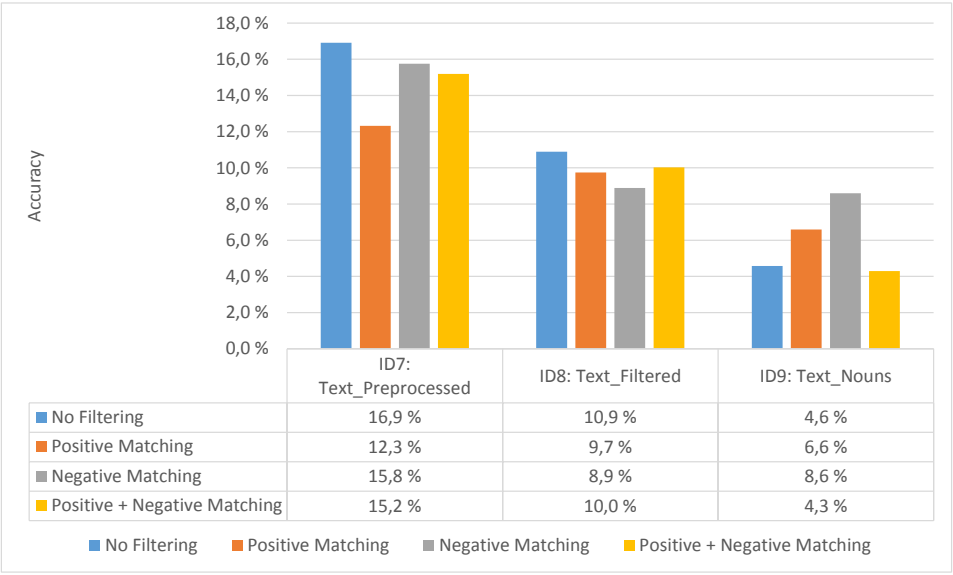


Figure 5.4: Comparison of results for variations of sources for user profiles

Tweets, or to use the part-of-speech filtered text (ID 9) which only includes nouns used in the Tweets. The effects of these various sources used are shown in figure 5.4.

Using only nouns result in the poorest scores, which is expected due to the loss of information. It could be hypothesized that the nouns are more descriptive or indicative of the themes or topics concerning a collection of documents than other classes of words. Ideally then, with the removal of non-nouns we ultimately reduce the amount of noise and stopwords which generally bear no meaning. Our experiments however reveal that this is not the case, and that other classes of words bear importance as well.

We also observe the reduced accuracy when using the near-original text when compared to the stopword-filtered sources, which exhibits the significance of stopword filtering effectively removing noise without sacrificing information.

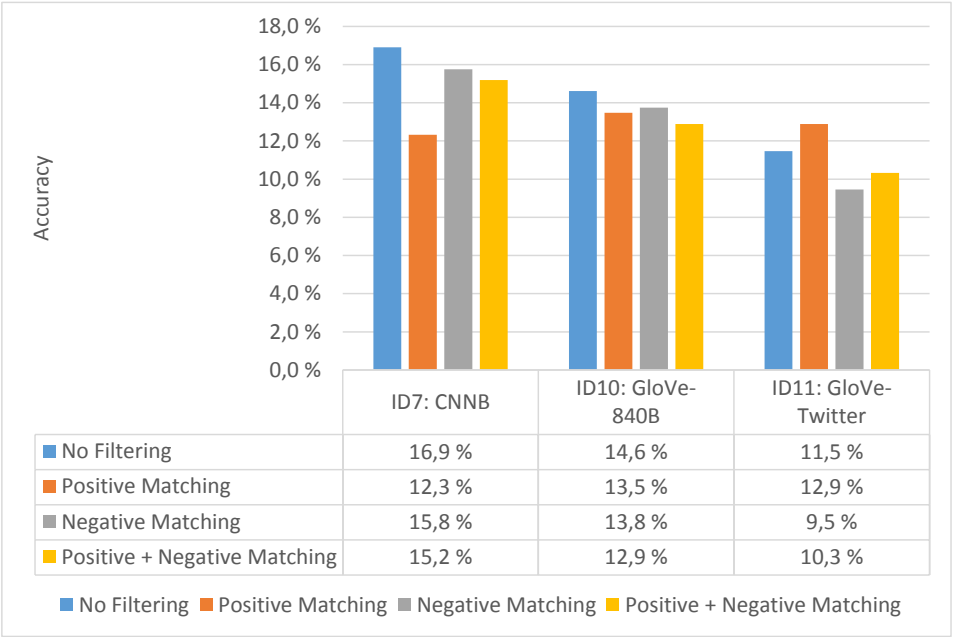
5.2.6 Word Embeddings

Figure 5.5 compare the different pre-trained word embedding models used for converting the set of extracted features into a single averaged term vector. Similarly to the results found during testing of the neural network sentiment classifiers in section 4.2.3, we see the ConceptNet Cumberbatch model outperforming the other two overall, with GloVe-840B edging out the Twitter-trained GloVe model. The latter observation is not surprising, given that GloVe-840B is trained on a significantly larger amount of samples and consequently has a larger vocabulary.

CNNB being an ensemble model proves to be more effective at capturing semantic similarities. An interesting observation however is the significantly reduced performance when filtering for positive sentiments, at least relative to the other types of filtering. It

would seem that the model is less effective at embedding positive terms, being surpassed by the other two models. The Twitter-trained GloVe model also performs considerably better in the case of positive filtering, relative to the other types of filtering. This is presumably because the positive terms used in the Tweets are unique to social media, which the model has been trained on.

We continue using CNNB for the remaining tests as it is the overall best performing model. While it performs relatively worse with positive sentiment filtering, the general trend so far has shown the sentiment filtering to rather reduce accuracy than increase it.



	ID7: CNNB	ID10: GloVe-840B	ID11: GloVe-Twitter
No Filtering	16,9 %	14,6 %	11,5 %
Positive Matching	12,3 %	13,5 %	12,9 %
Negative Matching	15,8 %	13,8 %	9,5 %
Positive + Negative Matching	15,2 %	12,9 %	10,3 %

Figure 5.5: Comparison of results for variations of pre-trained word embedding models

5.2.7 Feature Extraction

So far, the features for the user and movie profiles in our experiments have been extracted using tf-idf. The alternative is to use topic modeling and extract important features from there. As seen in figure 5.6 however, using the topic model results in substantially worse accuracy. The lack of proper selection and tuning of hyperparameters for optimal modeling is likely the culprit, leading to less than optimal features found and possibly ignoring many important terms that the tf-idf method captured. Another possibility is the short length of Tweets impacting the effectiveness of the statistical modeling. Ideally, we would observe an improvement in accuracy using topic modeling, though this requires more time devoted to fine-tuning the model. As this would extend outside the scope of this thesis, the use of tf-idf for feature extraction is the better choice in our case.

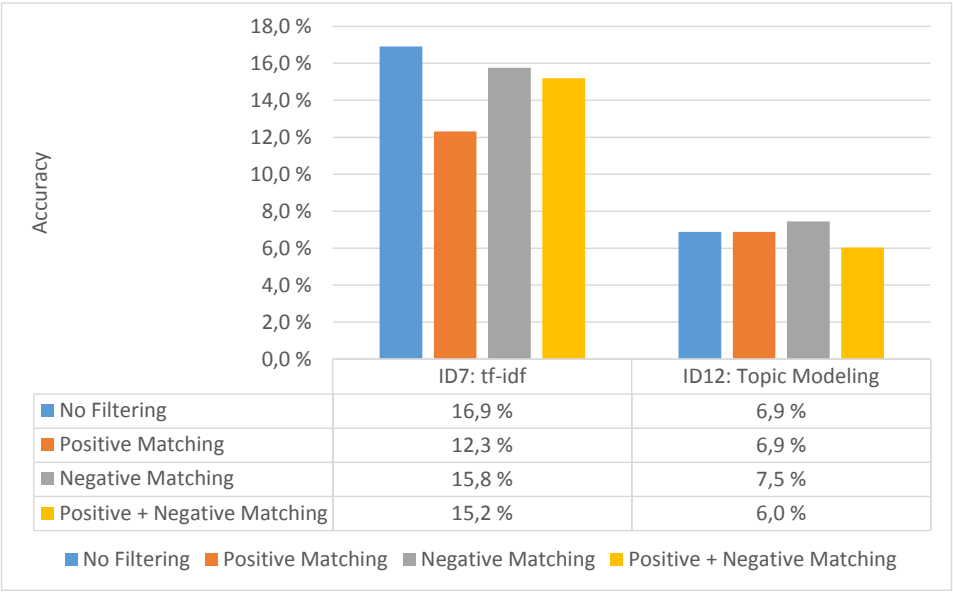


Figure 5.6: Comparison of results for variations of feature extraction methods

5.2.8 Entity Matching

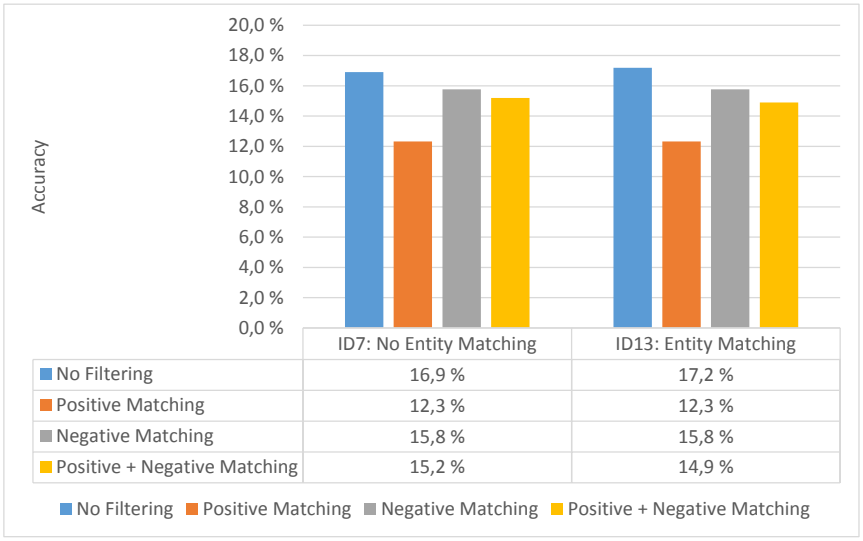
Earlier, we proposed the matching of named entities to produce a scoring factor, depending on the amount of positively or negatively named entity matches between a given user and a candidate movie. Ideally, positive matches would lead to a positive factor, and conversely negative matches add a negative factor to the similarity score. Figure 5.7 shows that the impact of the matching is minimal. There are multiple possible causes for this. One being the matching factor being too small to make any significant jump, although this is offset by the rationale of the factor being increased for each positive match - a small factor would mean that there were simply too few matches or a proportionate amount of positive and negative matches. It is also very likely that the users in our dataset generally did not Tweet about the named entities associated with the movies, as well as the named entity recognizer method failing to recognize named entities in the set of Tweets.

As the entity matching does not negatively impact the accuracy, and ideally would show an improvement for users displaying more explicit opinions with regards to named entities, we choose to include the entity matching.

5.2.9 Recommender Techniques

Having selected the variables that maximize the accuracy in the earlier steps, we now compare the recommender techniques in figure 5.8.

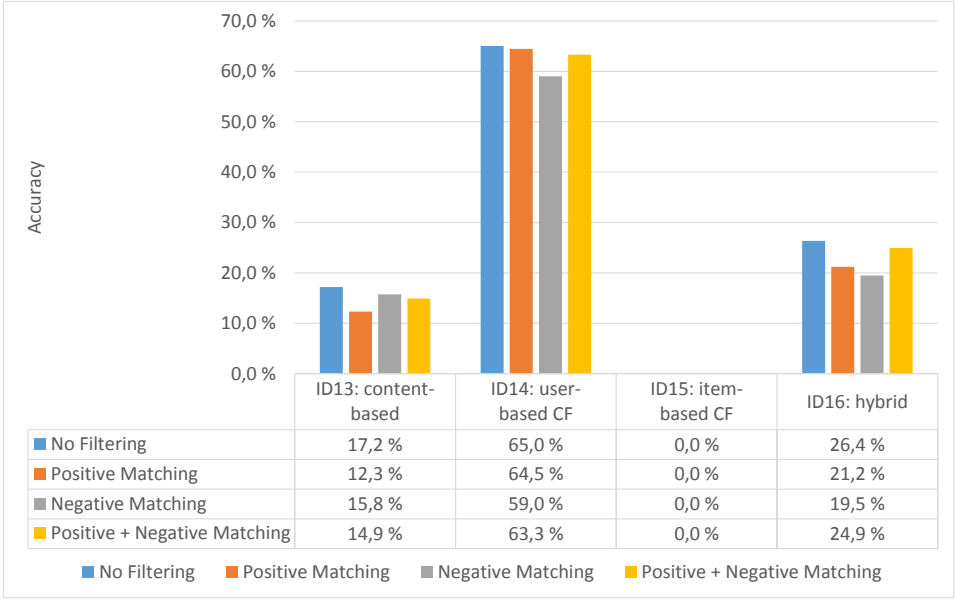
Content-based The baseline content-based approach leads to a maximum accuracy of around 17%, a rather unremarkable result. User preference elicitation is difficult when



	ID7: No Entity Matching	ID13: Entity Matching
No Filtering	16,9 %	17,2 %
Positive Matching	12,3 %	12,3 %
Negative Matching	15,8 %	15,8 %
Positive + Negative Matching	15,2 %	14,9 %

No Filtering Positive Matching Negative Matching Positive + Negative Matching

Figure 5.7: Comparison of results when including named entity matching



	ID13: content-based	ID14: user-based CF	ID15: item-based CF	ID16: hybrid
No Filtering	17,2 %	65,0 %	0,0 %	26,4 %
Positive Matching	12,3 %	64,5 %	0,0 %	21,2 %
Negative Matching	15,8 %	59,0 %	0,0 %	19,5 %
Positive + Negative Matching	14,9 %	63,3 %	0,0 %	24,9 %

No Filtering Positive Matching Negative Matching Positive + Negative Matching

Figure 5.8: Comparison of results for variations of recommender techniques

the data available is restricted, i.e. in the sense of both size (Twitter's retrieval restriction of 3200 recent Tweets) as well as the content of those Tweets themselves. Unless a user is genuinely interested in movies, the proportion of movie-related Tweets is minuscule - likely in the range of 1-20 Tweets for a sample size of 3200 Tweets. Additionally, such Tweets are generally non-specific in terms of movie aspects. A typical Twitter user will want to express their sentiments towards a movie, but will often not elaborate further. Attempting to elicit user preferences with regards to movies is thus a difficult task.

Using other aspects extracted for similarity comparisons works to some extent, though the results ultimately show that the amount of movie-related Tweets are too scarce for the content-based recommender to be effective in our case. Filtering for said Tweets is certainly a possibility, however would just lead to more similar movies being recommended rather than returning results providing the user with a serendipitous experience.

User-based collaborative filtering Using collaborative filtering between users result in quite the jump in accuracy, reaching 65% at the highest. While attempting to compare features between users and movies proved quite ineffective, comparing users to users on the other hand appears to be a superior approach. This is likely due to our features capturing a user's personality and preference, the comparison of which is more compatible than in the case of the movie profiles. A caveat with this approach is the requirement of labeling or otherwise detecting a user's liked movies.

Item-based collaborative filtering This is an interesting technique, as the results displayed show a 0% accuracy overall. In this approach we compare movie profiles to find the most similar ones, in which we compare a user's liked movie(s) to all the movies, and return the most similar ones. Any given movie will be most similar to itself, so it would always be at the top of the returned results. However, this is an undesirable behaviour, and thus we rather exclude the liked movie(s) from the comparison set of movies with the consequence of our evaluation metric being reduced to 0.

Listing 5 shows a sample output for a random user whose liked movie is *Star Wars: Episode V - The Empire Strikes Back*. The top 3 results are other movies in the *Star Wars* series, closely followed by other similar movies in terms of genre as well as style style, e.g. *The Lord of the Rings* which is also an epic fantasy series. The other movies found are also in related genres, e.g. war, adventure, sci-fi. This demonstrates the effectiveness of using word embedding vectors, and shows how such embeddings can recognize and cluster related genres on the basis of just the movies' plot summaries and synopses. By averaging a movies features into a single vector, this resulting vector effectively maps the movie into the embedding space - empowering simple comparison measures such as cosine similarity.

Hybrid While the item-based CF method excels at finding similar movies, this might not be ideal in terms of producing serendipitous or diverse recommendations. Our hybrid approach combines the user-based and item-based collaborative filtering methods to address all of the aforementioned concerns. We observe that this leads to an overall reduced accuracy compared to using only user-based CF, however this is due to caveat of exclusion with the item-based method. Properly balancing the weighting of scores between the two methods into a single score is also a likely issue. The approach still performs better than

```
ID        Title                                                          Score
------------------------------------------------------------------------------
tt0086190 Star Wars: Episode VI - Return of the Jedi                     0.9962
tt0076759 Star Wars                                                      0.9366
tt2488496 Star Wars: Episode VII - The Force Awakens                     0.8940
tt0167261 The Lord of the Rings: The Two Towers                          0.8024
tt1392190 Mad Max: Fury Road                                             0.7975
tt0167260 The Lord of the Rings: The Return of the King                  0.7921
tt0090605 Aliens                                                         0.7878
tt2015381 Guardians of the Galaxy                                        0.7864
tt0848228 The Avengers                                                   0.7624
tt0325980 Pirates of the Caribbean: The Curse of the Black Pearl         0.7591
tt0078788 Apocalypse Now                                                 0.7508
tt1201607 Harry Potter and the Deathly Hallows: Part 2                   0.7431
tt0032553 The Great Dictator                                             0.7400
tt0120737 The Lord of the Rings: The Fellowship of the Ring              0.7395
tt0172495 Gladiator                                                      0.7366
tt0097576 Indiana Jones and the Last Crusade                             0.7360
tt0082971 Raiders of the Lost Ark                                        0.7318
tt0056172 Lawrence of Arabia                                             0.7311
tt0468569 The Dark Knight                                                0.7280
```

Listing 5: Sample recommender output for item-based collaborative filtering method

the pure content-based recommender however, though evaluating the degree of diversity or serendipity for the produced recommendations is impossible without access to feedback from the test users.

5.3 Significance of Results

Having seen the results of our experiments and discussed them briefly, we now further elaborate on their significance and summarize. Although the content-based recommender was rather ineffective in our case, we observed potential in the user-based collaborative filtering approach, though with the requirement of user profile labeling.

5.3.1 Sentiment Filtering

While we hypothesized that sentiment filtering would have a positive impact on our results, the general trend throughout the experiments showed that accuracy was actually lowered when applying filtering. The choice of word embedding model had an impact as shown earlier, as well as the sentiment analysis module being imperfect. Nonetheless, the filtering of Tweet sentiments generally resulted in reduced accuracy, likely due to the loss of information. The concept of sentiment filtering is still an interesting one, and may have potential for future experiments.

5.3.2 Power of Word Embeddings

The use of word embeddings proved to be a very effective means of representing individual terms, as well as providing an efficient embedding space for both the user and movie profiles. Experiments showed that the effectiveness of the embedding models were highly dependant on the training sources as well as the size of these. The Twitter-trained model we used is an older one, and we predict that a newer model trained on an even larger amount of Tweets would be an improvement in terms of capturing semantic and syntactic relationships despite the intricacies of Twitter language.

5.3.3 Twitter Profiles for Recommendation

One of the goals of this thesis was to examine the possibility of using Twitter profiles as a means of providing insight to a user's likes and preferences, addressing the cold start problem of the recommender system. As Twitter users often Tweet about their immediate events and interests in their lives, one can deduce that they will seldom Tweet about movies unless they are specifically interested. Using Twitter timelines to detect aspects and entities related to movies they express opinions about is thus a difficult task due to the sparse frequency of such Tweets. By modeling a user's interest and preference in general rather than just those pertaining to movies, we observed that the comparison of similarity between the user and the set of movies was effective to some extent, however the approach would need more refinements for accurate predictions.

5.3.4 The Social Aspect

The addition of features from the social circle also proved successful in terms of increased accuracy. Even though the definition of social circle was quite loosely defined for our experiments, we were able to demonstrate the potential of exploiting such information for recommendation - while also potentially addressing the cold start problem in the case of a user having insufficient content for the elicitation of features or preferences.

Chapter 6

Conclusion and Future Work

In section 6.1 we conclude our findings and contributions and present our achievements with regards to the research questions introduced earlier. Finally, in section 6.2 we present propositions for future work.

6.1 Conclusion

Exploring the possibility of recommending content using social media as a source has been the driving motivation for this project. The primary focus of the project has been on examining and evaluating Twitter as a source for recommending movies. We gathered and categorized datasets by manually labeling users by their explicitly liked movies, as well as using a subset of an existing movie database for producing recommendations. A prototype system was implemented, handling everything from text processing to displaying meaningful data. Our *two-step recommender approach* processes the users' Tweets, extracts and aggregates features as well as named entities for profile representation and similarity comparisons, and ultimately attempts to produce a set of movies that any given user would like. These results would then ideally be presented to an end-user to interpret and aid in their personal decision-making with regards to finding a suitable movie to watch. While we were not able to perform usability testing with an actual set of Twitter users, the manual labeling of users provided us with a quantifiable measure of performance for the experiments with our system - showing considerable potential given that certain requirements are met, i.e. labeling users with their preferred or liked movies (expressed either implicitly or explicitly). Additionally, we showcased the system's capability of performing state-of-the-art classification on Tweets for automated labeling using supervised learning.

6.1.1 Goal Achievements

This section summarizes how the research questions introduced in section 1.3 have been answered.

Recall that we initially introduced two main research questions:

RQ 1 *How can we use Twitter profiles to recommend movies?*

RQ 2 *How viable is it to leverage Twitter profiles for recommendation with regards to the cold start problem?*

Answering the first question involves answering the three following subquestions:

RQ 1.1 *How can we identify and extract important features from a user's Twitter profile?*

The task of identifying and extracting features using the retrospective content from a Twitter profile aims to alleviate the cold start problem typically found for new users in recommender systems. In chapter 5, we saw that identifying and extracting features pertaining to a Twitter user was possible, though with varying results. Our approach of using *tf-idf* transformation on the collection of Tweets followed by extracting the most important terms was a simple one, though proved effective when compared to a more novel approach using topic modeling in our case. The use of named entities and part-of-speech tagging unfortunately did not have any significant impact in our experiments. In the former case, we found that the proportion of relevant named entities mentioned for our set of users was too sparse, as well as the automatic named entity recognizer likely missing certain entries. Eliciting movie-related features from Tweets was also problematic, mostly due to the general lack of Tweets pertaining to movies for the average user, which is further compounded by the restriction of only the most recent 3200 Tweets for any user from the Twitter API.

RQ 1.2 *How does a user's social circle affect these features?*

For our experiments, the definition of social circle was rather loose: comprising of only 10 randomly selected followees for each user with no specified requirement to the relationship between the user and the followee. Despite this, we observed an overall increase in recommender accuracy when including features from the users' social circles, showing great potential for further research. This improved accuracy implies that users follow people whom they share preferences or similarities with, or that these followees are influential with regards to these users. In general, we would expect peers within the same social circle to have similar interests to a certain extent. Conversely, humans are complex individuals with equally complex sets of interests and hobbies.

We predict that a more novel approach would improve upon our results, e.g. by exploiting relationships between the user and followee to find a subset of influential followees. The level of influence could be calculated using factors such as level of interaction between the users: retweets, likes, mentions; whether the user and followee are mutual followers - implying that they are friends or otherwise know each other to some degree. Again, this assumes that a user is likely to share many, if not all, of the preferences with the influential people in their social circle.

RQ 1.3 *How can we use these features to recommend movies to users?*

Using the set of candidate terms extracted earlier as features, we then needed a way to compare users and movies based on said features. Word mover's distance was shown to be a fairly accurate method of calculating the similarity between two such sets of features, however the time complexity of the method was multiple orders of magnitude higher when compared to the baseline approach of using word embeddings with cosine similarity with comparable accuracy. Various well-studied recommender techniques were experimented with, using the metadata available to enhance their performance - though with the requirement of having a reliable movie database at our disposal. We also considered multiple factors when comparing profiles for recommendation: sentiment filtering, named entity matching, using information from a user's social circle. While not all of these proved to be beneficial for the outcome, most showed potential for further refinements.

The content-based recommender approach of comparing user and movie profiles was effective to some extent, however was severely limited due to the general lack of movie-related Tweets. We observed that collaborative filtering performed considerably better than the content-based counterpart, though requires automatic labeling of users or explicit feedback or information from users pertaining to movies that they like.

RQ 2 *How viable is it to leverage Twitter profiles for recommendation with regards to the cold start problem?*

One of the main motivations for the works presented in this thesis was to explore the possibility leveraging Twitter profiles in order to elicit user preference and interest.

The results demonstrated by our prototype implementation of a two-step recommender system has shown that using social media profiles as an initial starting point for recommendation indeed is a viable approach to tackle the classic cold start problem, at least to some degree. As briefly mentioned in section 5.3, the attempt to detect and model preferences to specific entities (in our case, movies) was difficult due to the sparseness and diversity of a typical Twitter user's data basis. However, we found that modeling for general user interest and preference showed improved results, likely due to the larger amount of data available. Furthermore, the issue of users with low amount of participation or engagement (e.g. small social circles, few or even no Tweets) may also impact the feasibility of the aforementioned approach.

Ultimately, while we have found promising results, we still believe that further refinement and experimentation with real user feedback is needed to properly assess the viability of using these profiles in real world scenarios.

6.2 Future Work

The following subsections describe proposals for future work on our system. These include suggestions for improvements, extensions, or other approaches.

6.2.1 Improvements to the Current System

From the results in chapter 5, we see the potential for several improvements to our system. The preprocessing pipeline can be improved, particularly the named entity recognition and sentiment analysis modules, as these are integral parts of the system. The named entity

recognition model needs more training to improve, and in an ideal world it would be able to detect and correctly label all named entities in any Tweet. If that were reality we could automatically filter and categorize any movie or entity given a Tweet, as well as aiding in the task of annotating user profiles with movies or entities that they have expressed sentiments about. Connecting to an external semantic knowledge base for the detection of entities and concepts is also a possible extension.

As for the sentiment analysis, it would also be ideal to achieve a higher classification accuracy. Even with significant advances within natural language processing, machines still struggle to understand the human language with all of its intricacies - a trait that is intensified by the informal and short nature of Tweets, which encourages users to be creative with expressions and semantics within the limitations of character length. We observed that the current state-of-the-art approach with artificial neural networks and word embeddings for text representation could capture syntactic and semantic relationships in text, although results depend on having access to a sizable set of training data as is the case for most machine learning methods. While we used pretrained word embedding models, it would be interesting to examine the effects of retrieving and training a custom model on a more recent set of Tweets in order to capture new contexts and terms that may have appeared among the Twitterverse. Another thing we did not consider was the use of emoji in terms of determining sentiment polarities and strength. It would be interesting to study the impact these have in terms of determining sentiments and polarities. Hashtags were also features largely ignored due to difficulties with regards to splitting compound hashtags, however it could be interesting to examine the significance of these similarly to the case of emojis.

Detecting and filtering spam and mundane/irrelevant Tweets could also have significant effects. Preliminary work ([70]) showed that a substantial amount of Tweets had no significance in terms of expressing subjective opinions, as well as negatively affecting the features and named entities extracted and suggested that eliminating these would likely lead to more informative and valuable results. While we experimented with sentiment filtering to filter out Tweets with no associated sentiment polarity, results showed little to no improvement in most cases. Filtering for spam and mundane Tweets is however open for further research. For instance, Pitsilis *et al.* [81] propose a detection scheme for distinguishing Tweets containing offensive language from normal text using an ensemble of RNN classifiers. Similarly to how we utilized deep learning methods for sentiment classification, one could apply such an approach to the task of filtering out spam and irrelevant Tweets.

Improvements performance-wise are also possible. Currently, the system performs several steps of redundant, offline batch processing (for experimental purposes) instead of calculating and aggregating results in real-time. The system could be adapted to do so however, and could be extended with e.g. Apache Spark[1] for scalability and parallel processing, handling both offline processing as well as real-time processing of data piped directly from a stream. Additionally, this enables the possibility of providing users with near real-time recommendations that could improve through the user's continued use. Adding a feedback loop from the users enabling them to rate the system's suggested recommendations could also serve as a basis for further improvements.

[1] `http://spark.apache.org/`

6.2.2 Feature Selection

The user profiles we constructed for comparison and recommendation used key terms extracted from the user's collection of Tweets as features. We also factored in named entity matching as well as sentiment analysis in these comparisons. However, a significant amount of metadata available from Twitter were never explored in our experiments, e.g. user location, time zone, preferred language, or geotagged Tweets (coordinates). Recommendation based on such geographical data is an interesting approach, though is restricted by the user's willingness and consent to provide such information. Using other characteristics such as demographics or behavior for the analysis and elicitation of user needs and preference are also potential perspectives to examine. The temporal dimension would also be worth looking at. Tastes may change with time, such that an older Tweet might have less relevancy or be less representational of a user's current taste or preference.

Another unexplored aspect is examining the context of Tweets, e.g. replies, conversation threads, or attached media such as images or videos. The text of the Tweets may not make sense without considering the context. The same can be said in the case of media, where there is a possibility of the media itself containing text or be a form of implicit reaction or expression. It would be interesting to extract the features from such media, e.g. by using optical character recognition or other image processing methods.

6.2.3 Social Circles

As discussed in section 6.1, the usage of features elicited from a user's social circle had a positive effect on the accuracy during testing despite our loose definition of social circle. The possibility of continuing with this idea using a more novel method such as the one outlined earlier to select the influential followees in the user is an interesting one - fully utilizing the social network data available from Twitter.

6.2.4 Machine Learning Classifiers & Recommender Systems

Various machine learning classifiers including ANNs were experimented with for the sentiment analysis step, though we reverted to more traditional, well-studied methods of comparison for producing recommendations. While we to some extent had embedded user preference in a vector space and were able to implement a model for training and evaluation of a neural network classifier for our set of movies, there were certain details hindering this theoretical approach. One is the lack of extensive labeling that the training and testing of machine learning classifiers requires; our annotated dataset is too small for a classifier to be effective. Given that a dataset of satisfactory size is obtained, using a machine learning classifier for recommendation is a possible approach for further studies.

A possible approach would be to use ANNs for the entire process. For instance, one could use the user and item embeddings (i.e. the real-valued vectors that represent a user or item) presented earlier as a basis for the input layers. A combination of ANNs could then be utilized to model both static preferences as well as dynamic ones, in the latter case using networks such as RNNs to learn time-variant features. Deriving features from the social circle information and using these as inputs is also a possible approach, e.g. clustering

or categorization of similar users based on demographics. In that way, one could capture both user-item similarities as well as user-user similarities in the feature vectors.

One could also look into more elaborate methods within NLP for understanding as well as eliciting structured information from Tweets. It would be interesting to be able to cluster and categorize sentiments and preferences with regards to items. Approaches using deep learning for understanding and parsing textual content, even in noisy contexts such as Tweets are interesting subjects for further research. Prominent examples here include the works of the Google NLU team[2] and the Snips NLU library[3].

6.2.5 Other Domains

While this thesis has primarily been focused on recommending products in the movie domain, the method of extracting features from a user's Twitter profile is a fairly general one, with the exception of the usage of domain-specific stopword filtering. Consequently, our suggested method recommendation is also a general one, and thus we believe that the two-step recommender approach could be adapted to other domains with minor modifications, given that a database of the products is available for feature extraction. Using a different social network such as Facebook for user profile construction is also a possibility, although taking advantage of the available data should be further experimented with, e.g. extracting information from liked pages, or using listed interests, places visited, demographic data as features.

6.2.6 User-Based Evaluation

As mentioned earlier in section 4.5, the evaluation of the recommender system involves qualities which are not easily judged without user feedback. Ideally, we would gather a set of representative Twitter users to test the recommender system and provide feedback on the results in terms of variety, serendipity, and relevance, among others. While we were not able to perform such an evaluation for our thesis due to time restrictions, it would be interesting to have user feedback to examine the effects of the various approaches experimented with with regards to the aforementioned properties.

[2] https://ai.google/research/teams/nlu
[3] https://github.com/snipsco/snips-nlu

Bibliography

[1] O. Varol *et al.*, "Online human-bot interactions: Detection, estimation, and characterization," *CoRR*, vol. abs/1703.03107, 2017. [Online]. Available: http://arxiv.org/abs/1703.03107.

[2] M. Efron, "Information search and retrieval in microblogs," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 6, pp. 996–1008, 2011, ISSN: 1532-2890. DOI: 10.1002/asi.21512. [Online]. Available: http://dx.doi.org/10.1002/asi.21512.

[3] P. N. Kralj *et al.*, "Sentiment of emojis," *PLOS ONE*, vol. 10, no. 12, pp. 1–22, Dec. 2015. DOI: 10.1371/journal.pone.0144296. [Online]. Available: http://dx.doi.org/10.1371%5C%2Fjournal.pone.0144296.

[4] P. Koehn and K. Knight, "Empirical methods for compound splitting," in *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, ser. EACL '03, Budapest, Hungary: Association for Computational Linguistics, 2003, pp. 187–193, ISBN: 1-333-56789-0. DOI: 10.3115/1067807.1067833. [Online]. Available: http://dx.doi.org/10.3115/1067807.1067833.

[5] K. Macherey *et al.*, "Language-independent compound splitting with morphological operations," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11, Portland, Oregon: Association for Computational Linguistics, 2011, pp. 1395–1404, ISBN: 978-1-932432-87-9. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002644.

[6] R. Zafarani *et al.*, *Social media mining: An introduction*. Cambridge University Press, 2014.

[7] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. New York, NY, USA: Cambridge University Press, 2011, ISBN: 9781107015357.

[8] D. Jannach *et al.*, *Recommender Systems: An Introduction*, 1st. Cambridge University Press, 2010, ISBN: 9780521493369.

[9] B. Sarwar *et al.*, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW '01, Hong Kong, Hong Kong: ACM, 2001, pp. 285–295, ISBN: 1-58113-348-0. DOI: 10.1145/371920.372071. [Online]. Available: http://doi.acm.org/10.1145/371920.372071.

[10] R. Buettner, "Predicting user behavior in electronic markets based on personality-mining in large online social networks," *Electronic Markets*, pp. 1–19, 2016, ISSN: 1422-8890. DOI: 10.1007/s12525-016-0228-z. [Online]. Available: http://dx.doi.org/10.1007/s12525-016-0228-z.

[11] B. Liu, *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers, 2012, ISBN: 9781608458844.

[12] P. D. Turney, "Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 417–424. DOI: 10.3115/1073083.1073153. [Online]. Available: http://dx.doi.org/10.3115/1073083.1073153.

[13] J. Kreutzer and N. Witte, *Opinion mining using sentiwordnet*, 2013.

[14] D. Davidov *et al.*, "Enhanced sentiment learning using twitter hashtags and smileys," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, ser. COLING '10, Beijing, China: Association for Computational Linguistics, 2010, pp. 241–249. [Online]. Available: http://dl.acm.org/citation.cfm?id=1944566.1944594.

[15] G. Qiu *et al.*, "Expanding domain sentiment lexicon through double propagation," in *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, ser. IJCAI'09, Pasadena, California, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1199–1204. [Online]. Available: http://dl.acm.org/citation.cfm?id=1661445.1661637.

[16] X. Zhang *et al.*, "Sentiment analysis by augmenting expectation maximisation with lexical knowledge," in *Web Information Systems Engineering - WISE 2012: 13th International Conference, Paphos, Cyprus, November 28-30, 2012. Proceedings*, X. S. Wang *et al.*, Eds. Springer Berlin Heidelberg, 2012, pp. 30–43, ISBN: 978-3-642-35063-4. DOI: 10.1007/978-3-642-35063-4_3. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35063-4_3.

[17] Z. Zhai *et al.*, "Constrained lda for grouping product features in opinion mining," in *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I*, ser. PAKDD'11, Shenzhen, China: Springer-Verlag, 2011, pp. 448–459, ISBN: 978-3-642-20840-9. [Online]. Available: http://dl.acm.org/citation.cfm?id=2017863.2017907.

[18] V. Hatzivassiloglou and K. R. McKeown, "Predicting the semantic orientation of adjectives," in *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, ser. EACL '97, Madrid, Spain: Association for Computational Linguistics, 1997, pp. 174–181. DOI: 10.3115/979617.979640. [Online]. Available: http://dx.doi.org/10.3115/979617.979640.

[19] D. Freitag, "Machine learning for information extraction in informal domains," *Mach. Learn.*, vol. 39, no. 2-3, pp. 169–202, May 2000, ISSN: 0885-6125. DOI: 10.1023/A:1007601113994. [Online]. Available: http://dx.doi.org/10.1023/A:1007601113994.

[20] R. Grishman and B. Sundheim, "Message understanding conference-6: A brief history," in *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, ser. COLING '96, Copenhagen, Denmark: Association for Computational Linguistics, 1996, pp. 466–471. DOI: 10.3115/992628.992709. [Online]. Available: http://dx.doi.org/10.3115/992628.992709.

[21] W. Hua *et al.*, "Information extraction from microblogs: A survey," *Int. J. Software and Informatics*, vol. 6, pp. 495–522, 2012.

[22] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007, ISSN: 0378-4169. DOI: doi:10.1075/li.30.1.03nad. [Online]. Available: http://www.ingentaconnect.com/content/jbp/li/2007/00000030/00000001/art00002.

[23] D. M. Bikel *et al.*, "An algorithm that learns what‘s in a name," *Mach. Learn.*, vol. 34, no. 1-3, pp. 211–231, Feb. 1999, ISSN: 0885-6125. DOI: 10.1023/A:1007558221122. [Online]. Available: http://dx.doi.org/10.1023/A:1007558221122.

[24] K. Takeuchi and N. Collier, "Use of support vector machines in extended named entity recognition," in *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, ser. COLING-02, Association for Computational Linguistics, 2002, pp. 1–7. DOI: 10.3115/1118853.1118882. [Online]. Available: http://dx.doi.org/10.3115/1118853.1118882.

[25] J. D. Lafferty *et al.*, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01, Morgan Kaufmann Publishers Inc., 2001, pp. 282–289, ISBN: 1-55860-778-1. [Online]. Available: http://dl.acm.org/citation.cfm?id=645530.655813.

[26] J. R. Finkel *et al.*, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ser. ACL '05, Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 363–370. DOI: 10.3115/1219840.1219885. [Online]. Available: http://dx.doi.org/10.3115/1219840.1219885.

[27] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *CoRR*, vol. abs/1511.08308, 2015. [Online]. Available: http://arxiv.org/abs/1511.08308.

[28] X. Ma and E. H. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," *CoRR*, vol. abs/1603.01354, 2016. [Online]. Available: http://arxiv.org/abs/1603.01354.

[29] G. Lample *et al.*, "Neural architectures for named entity recognition," *CoRR*, vol. abs/1603.01360, 2016. [Online]. Available: http://arxiv.org/abs/1603.01360.

[30] K. Weinberger *et al.*, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09, Montreal, Quebec, Canada: ACM, 2009, pp. 1113–1120, ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553516. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553516.

[31] D. M. Blei *et al.*, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003, ISSN: 1532-4435. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944937.

[32] P. C. Barman *et al.*, "Non-negative matrix factorization based text mining: Feature extraction and classification," in *Neural Information Processing: 13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006. Proceedings, Part II*, I. King *et al.*, Eds. Springer Berlin Heidelberg, 2006, pp. 703–712, ISBN: 978-3-540-46482-2. DOI: 10.1007/11893257_78. [Online]. Available: http://dx.doi.org/10.1007/11893257_78.

[33] T. K. Landauer *et al.*, "An introduction to latent semantic analysis," *Discourse Processes*, vol. 25, no. 2-3, pp. 259–284, 1998. DOI: 10.1080/01638539809545028. eprint: http://dx.doi.org/10.1080/01638539809545028. [Online]. Available: http://dx.doi.org/10.1080/01638539809545028.

[34] W. Rui *et al.*, "Unsupervised feature selection for text classification via word embedding," in *2016 IEEE International Conference on Big Data Analysis (ICBDA)*, 2016, pp. 1–5. DOI: 10.1109/ICBDA.2016.7509787.

[35] T. Mikolov *et al.*, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. [Online]. Available: http://arxiv.org/abs/1310.4546.

[36] X. Rong, "Word2vec parameter learning explained," *CoRR*, vol. abs/1411.2738, 2014. [Online]. Available: http://arxiv.org/abs/1411.2738.

[37] T. Mikolov *et al.*, "Linguistic regularities in continuous space word representations," in *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, 2013, pp. 746–751. [Online]. Available: http://aclweb.org/anthology/N/N13/N13-1090.pdf.

[38] J. Pennington *et al.*, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[39] A. Go *et al.*, "Twitter sentiment classification using distant supervision," *Processing*, pp. 1–6, 2009. [Online]. Available: http://www.stanford.edu/~alecmgo/papers/TwitterDistantSupervision09.pdf.

[40] M. McCord and M. Chuah, "Spam detection on twitter using traditional classifiers," in *Proceedings of the 8th International Conference on Autonomic and Trusted Computing*, ser. ATC'11, Berlin, Heidelberg: Springer-Verlag, 2011, pp. 175–186, ISBN: 978-3-642-23495-8. [Online]. Available: http://dl.acm.org/citation.cfm?id=2035700.2035717.

[41] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999, ISBN: 0-262-13360-1.

[42] J. D. M. Rennie *et al.*, "Tackling the poor assumptions of naive bayes text classifiers," in *In Proceedings of the Twentieth International Conference on Machine Learning*, 2003, pp. 616–623.

[43] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ser. ACL '12, Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 90–94. [Online]. Available: http://dl.acm.org/citation.cfm?id=2390665.2390688.

[44] F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," Project PARA, Cornell Aeronautical Laboratory, Ithaca, New York, Report 85-460-1, Jan. 1957.

[45] D. E. Rumelhart *et al.*, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," in, D. E. Rumelhart *et al.*, Eds., Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362, ISBN: 0-262-68053-X. [Online]. Available: http://dl.acm.org/citation.cfm?id=104279.104293.

[46] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982, ISSN: 0027-8424. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/6953413].

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[48] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997, ISSN: 1053-587X. DOI: 10.1109/78.650093. [Online]. Available: http://dx.doi.org/10.1109/78.650093.

[49] J. Chung *et al.*, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. arXiv: 1412.3555. [Online]. Available: http://arxiv.org/abs/1412.3555.

[50] W. Yin *et al.*, "Comparative study of CNN and RNN for natural language processing," *CoRR*, vol. abs/1702.01923, 2017. arXiv: 1702.01923. [Online]. Available: http://arxiv.org/abs/1702.01923.

[51] A. Karpathy and F. Li, "Deep visual-semantic alignments for generating image descriptions," *CoRR*, vol. abs/1412.2306, 2014. [Online]. Available: http://arxiv.org/abs/1412.2306.

[52] X. Yang *et al.*, "On top-k recommendation using social networks," in *Proceedings of the Sixth ACM Conference on Recommender Systems*, ser. RecSys '12, Dublin, Ireland: ACM, 2012, pp. 67–74, ISBN: 978-1-4503-1270-7. DOI: 10.1145/2365952.2365969. [Online]. Available: http://doi.acm.org/10.1145/2365952.2365969.

[53] E. d. S. da Silva *et al.*, "Content-based social recommendation with poisson matrix factorization," in *Machine Learning and Knowledge Discovery in Databases*, M. Ceci *et al.*, Eds., Cham: Springer International Publishing, 2017, pp. 530–546, ISBN: 978-3-319-71249-9.

[54] M. G. Armentano *et al.*, "Movies recommendation based on opinion mining in twitter," in *Advances in Artificial Intelligence and Its Applications: 14th Mexican International Conference on Artificial Intelligence, MICAI 2015, Cuernavaca, Morelos, Mexico, October 25-31, 2015, Proceedings, Part II*, O. Pichardo Lagunas *et al.*, Eds. Cham: Springer International Publishing, 2015, pp. 80–91, ISBN: 978-3-319-27101-9. DOI: 10.1007/978-3-319-27101-9_6. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-27101-9_6.

[55] A. Gupta *et al.*, *Movie recommendations using social networks*, 2008.

[56] J. Zhang and Y. Lei, "Improving content recommendation in social streams via interest model," in *Computer and Information Science*, R. Lee, Ed. Springer International Publishing, 2015, pp. 57–70, ISBN: 978-3-319-10509-3. DOI: 10.1007/978-3-319-10509-3_5. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10509-3_5.

[57] J. Chen *et al.*, "Short and tweet: Experiments on recommending content from information streams," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10, Atlanta, Georgia, USA: ACM, 2010, pp. 1185–1194, ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753503. [Online]. Available: http://doi.acm.org/10.1145/1753326.1753503.

[58] H. Hamdan *et al.*, "Lsislif: Feature extraction and label weighting for sentiment analysis in twitter," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*, 2015.

[59] P. Zhao *et al.*, "Feature extraction from micro-blogs for comparison of products and services," in *Web Information Systems Engineering – WISE 2013: 14th International Conference, Nanjing, China, October 13-15, 2013, Proceedings, Part I*, X. Lin *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 82–91, ISBN: 978-3-642-41230-1. DOI: 10.1007/978-3-642-41230-1_7. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41230-1_7.

[60] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014. [Online]. Available: http://arxiv.org/abs/1408.5882.

[61] X. Zhang *et al.*, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems 28*, C. Cortes *et al.*, Eds., Curran Associates, Inc., 2015, pp. 649–657. [Online]. Available: http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf.

[62] C. dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, Dublin, Ireland: Dublin City University and Association for Computational Linguistics, 2014, pp. 69–78. [Online]. Available: http://www.aclweb.org/anthology/C14-1008.

[63] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15, New York, NY, USA: ACM, 2015, pp. 959–962, ISBN: 978-1-4503-3621-5. DOI: 10.1145/2766462.2767830. [Online]. Available: http://doi.acm.org/10.1145/2766462.2767830.

[64] P. Nakov *et al.*, "Semeval-2016 task 4: Sentiment analysis in twitter," in *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@ NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, S. Bethard *et al.*, Eds., The Association for Computer Linguistics, 2016, pp. 1–18. [Online]. Available: http://aclweb.org/anthology/S/S16/S16-1001.pdf.

[65] S. Rosenthal *et al.*, "SemEval-2017 task 4: Sentiment analysis in Twitter," in *Proceedings of the 11th International Workshop on Semantic Evaluation*, ser. SemEval '17, Vancouver, Canada: Association for Computational Linguistics, Aug. 2017.

[66] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, ser. CoNLL '09, Boulder, Colorado: Association for Computational Linguistics, 2009, pp. 147–155, ISBN: 978-1-932432-29-9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1596374.1596399.

[67] X. Liu *et al.*, "Recognizing named entities in tweets," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11, Portland, Oregon: Association for Computational Linguistics, 2011, pp. 359–367, ISBN: 978-1-932432-87-9. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002519.

[68] C. Li *et al.*, "Twiner: Named entity recognition in targeted twitter stream," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '12, Portland, Oregon, USA: ACM, 2012, pp. 721–730, ISBN: 978-1-4503-1472-5. DOI: 10.1145/2348283.2348380. [Online]. Available: http://doi.acm.org/10.1145/2348283.2348380.

[69] A. Ritter *et al.*, "Named entity recognition in tweets: An experimental study," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11, Edinburgh, United Kingdom: Association for Computational Linguistics, 2011, pp. 1524–1534, ISBN: 978-1-937284-11-4. [Online]. Available: http://dl.acm.org/citation.cfm?id=2145432.2145595.

[70] T. H. Nguyen, "Recommendations in social media," Specialization Project in Computer Science (TDT4501), Dept. of Computer, Information Science, Norwegian University of Science, and Technology, 2016.

[71] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, Ann Arbor, MI, 2014.

[72] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[73] R. Speer *et al.*, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, S. P. Singh and S. Markovitch, Eds., AAAI Press, 2017, pp. 4444–4451. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972.

[74] F. Chollet *et al.*, *Keras*, https://github.com/fchollet/keras, 2015.

[75] S. M. Mohammad and S. Kiritchenko, "Understanding emotions: A dataset of tweets to study interactions between affect categories," in *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference*, Miyazaki, Japan, 2018.

[76] S. Bird *et al.*, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[77] M. Collins, "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms," in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, ser. EMNLP '02, Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–8. DOI: 10.3115/1118693.1118694. [Online]. Available: http://dx.doi.org/10.3115/1118693.1118694.

[78] R. Jiang *et al.*, *Evaluating and combining named entity recognition systems*, 2016.

[79] M. J. Kusner *et al.*, "From word embeddings to document distances," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 957–966. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045221.

[80] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," English, in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, http://is.muni.cz/publication/884893/en, Valletta, Malta: ELRA, May 2010, pp. 45–50.

[81] G. K. Pitsilis *et al.*, "Detecting offensive language in tweets using deep learning," *CoRR*, vol. abs/1801.04433, 2018. arXiv: 1801.04433. [Online]. Available: http://arxiv.org/abs/1801.04433.