

Examensarbete
15 högskolepoäng, grundnivå

Förbättring av filmrekommendationer genom social media matchning

Improving movie recommendations
through social media matching

Roman Kuroptev
Anton Lagerlöf

Examen: kandidatexamen 180 hp
Huvudområde: datavetenskap
Program: systemutvecklare
Datum för slutseminarium: 2019-05-31

Handledare: Dimitris Paraschakis
Examinator: Johan Holmberg

Sammanfattning

Rekommendationssystem är idag väsentliga för att navigera den enorma mängd produkter tillgängliga via internet. Då social media i form av Twitter vid tidigare tillfällen använts för att generera filmrekommendationer har detta främst varit för att hantera cold-start, ett vanligt drabbande problem för collaborative-filtering. I detta arbete adresseras istället hur top-k rekommendationer påverkas vid integrering av social media data i rekommendationssystemet. För att svara på denna fråga har en prototyp av nytt slag utvecklats inom processmodellen för Design Science. Systemet rankar om top-k rekommendationer baserat på resultatet av social matchning där användares Tweets matchas med nyckelord för filmer genom latent semantic indexing (LSI) similarity. Prototypen evalueras genom experiment som adresserar funktionalitet, noggrannhet, konsekvens och prestanda. Resultatet visar att mätetalen NDCG och MAP för top-k rekommendationer förbättras med social matching jämfört med att enbart använda collaborative filtering.

Abstract

Recommender systems are a crucial part of navigating the vast number of products on the internet. Social media, in the form of Twitter microblogs, has been previously used to produce movie recommendations, yet this has mainly been to solve cold-start, a common problem in collaborative filtering environments. This work addresses how top-k recommendations in a collaborative filtering environment are affected when augmented with social media data. To answer this question a novel prototype is developed following a design science process model. This system re-ranks top-k recommendations based on a social matching process where Tweets are matched with movie keywords through latent semantic indexing (LSI) similarity. The prototype is evaluated through experiments regarding functionality, accuracy, consistency, and performance. The results show that NDCG and MAP metrics of the top-k recommendations improve with social matching compared to only using the collaborative filtering algorithms.

Glossary

Top-k A ranked list of recommendations where k is the size of the list. Best recommendations are in the top and less relevant items are further down in the list.

Social-matching The process of matching all words in a user's Twitter timeline with keywords for a movie.

Cold Start problem: A situation in collaborative filtering algorithms where not enough ratings, for either a user or an item, exist to be able to produce accurate recommendations.

Corpus A collection of written texts.

Instance A working system created through design science. Also referred to as instantiation.

Artifact Models, methods, constructs, instantiations or design theories created through design science.

MAP A metric for evaluating all recommendations generated by a recommendation system. Takes the ranking of relevant items into account.

NDCG Similar metric to MAP used to evaluate the ranking of recommendations generated by a recommender system. But unlike MAP uses real-valued relevance instead of binary.

Twitter-timeline A collection of all tweets posted by a Twitter user.

IMDb Internet Movie Database.

LSI Latent Semantic Indexing is a technique to analyze the relationships between terms in a document.

NLP Natural Language processing.

Table of contents

1.	Introduction	1
1.1	Research question	2
1.1.1	Objective	2
1.2	The structure of this thesis	2
2.	Background	3
2.1	Literature review	3
2.2	Recommender systems	4
2.2.1	Overview	4
2.2.2	Content-based filtering	4
2.2.3	Collaborative filtering	5
2.2.4	Model-based collaborative filtering using Matrix factorization	6
2.2.5	Memory-based collaborative filtering using KNN	6
2.2.6	Twitter as a data source	7
2.3	LSI	7
2.3.1	Vector space model	7
2.3.2	Latent Semantic Index	8
2.3.3	Cosine similarity	9
2.4	Evaluating recommender systems	9
2.4.1	Item ranking prediction	9
2.4.1.1	Mean average precision	10
2.4.1.2	Normalized Cumulative Discounted Gain	10
3.	Method	12
3.1	Design creation methodology	12
3.1.1	Activity 1	13
3.1.2	Activity 2	13
3.1.3	Activity 3	14
3.1.4	Activity 4	14
3.1.5	Activity 5	15
3.1.6	Activity 6	15

3.2 The data	16
3.2.1 MovieTweetings	16
3.2.2 Pre-processing the dataset	16
3.2.3 Sentiment classification	17
3.2.4 Twitter Data pre-processing	17
3.3 Experiments & Evaluation	18
3.3.1 Selected DS evaluation criteria	18
Criteria 1: Functionality	18
Criteria 2: Consistency (consistent behavior, stability)	18
Criteria 3: Accuracy (correctness and preciseness)	18
Criteria 4: Performance	18
3.3.2 Experimental overview	18
3.3.3 Main instance parameters	19
Purpose and parameter tuning	19
3.3.4 Two underlying algorithms	20
3.3.5 Randomization of top-k	20
3.3.6 Experiments	21
Experiment 1: Weighted Matrix Factorization (E1)	21
Experiment 2: Weighted Matrix Factorization with precision at k threshold (E2)	21
Experiment 3: KNN(E3)	22
Experiment 4: KNN with precision at k threshold (E4)	22
Experiment 5: Optimal user (E5)	23
Experiment 6: Execution time of Artefact (E6)	24
Experiment 7: Execution time of offline- versus online-social matching (E7)	24
3.4 Method discussion	25
4. Design & implementation	27
4.1 The Design Phase	27
4.2 Implementation	29
4.2.1 The Recommender	29
MyMediaLite	30

4.2.2 Social Matching	30
Movie Corpus	32
OMDB	32
MovieDB	32
WordNet	32
MediaWiki	32
5. Result	33
5.1 General statistics of the systems	33
5.2 Experiment results	35
Experiment 1: Weighted Matrix Factorization (E1)	35
Table 1: Results for WRMF algorithm and social matching.	35
Experiment 2: Weighted Matrix Factorization with precision at k threshold(E2)	35
Experiment 3: KNN (E3)	36
Table 3: Results for the KNN algorithm with social matching.	36
Experiment 4: KNN with precision at k threshold(E4)	36
Table 4: Results for KNN with social matching and precision at k threshold.	36
Experiment 5: Optimal user (E5)	36
Table 5: Results for Optimal user experiment	36
Experiment 6: Execution time of Artefact (E6)	37
Experiment 7: Execution time of offline- versus online-social matching (E7)	37
6. Analysis	38
6.1 Prototype implementation analysis	38
6.2 General analysis	38
7. Discussion	41
8. Conclusion	42
9. Future Work	42

1. Introduction

Today, digital information is of abundance. More and more shopping, as well as consumption of media, has moved from real-world physical stores to the online arena. Whereas the limitations of physical shopping are set by how many articles can fit in the available physical space, online shopping and consumption of media, however, is not bound by these limitations. As of January 2018, a customer browsing the inventory of e-commerce mogul amazon.com was exposed to as many as 564 million items when accessing us marketplace (How Many Products Does Amazon Sell Worldwide – January 2018, 2018).

Recommendation systems play an important role in helping users find what they might like from this ever-increasing sea of information. By utilizing a user’s prior activity, a prediction can be made on whether he/she will like an item or not. According to research by McKinsey, as much as 35% of Amazon sales are a result of recommendations, and for the on-demand streaming service Netflix, an estimated 75% (MacKenzie, Meyer, & Noble, 2013).

Leading players in the digital arena, such as Amazon, YouTube and Netflix have demonstrated the value recommendation systems offer on both revenue and user experience. According to a 2015 paper published by Netflix, the company estimates their recommendation systems save up to as much as 1bn dollars per year. The authors of the paper state that “When produced and used correctly, recommendations lead to meaningful increases in overall engagement with the product (e.g., streaming hours) and lower subscription cancellation rates” (Gomez-Uribe & Hunt, December 2015).

The demand for well-performing recommender systems is high, making efforts of improving the underlying techniques which make these systems possible, relevant.

Another heavy player in the digital arena is social media. Here opinionated texts are regularly shared by millions of users on various topics and events. Safe to say, social networks are a rich source of users prior, and current opinions.

In this thesis, we propose SwarmBoost, a novel system integrating user data from Twitter into a recommender system environment. The proposed system, based on MovieTweetings (Dooms, Pessemier, & Martens, 2013), a dataset of Twitter user IDs and their corresponding movie ratings, is utilized to perform movie-timeline matching. SwarmBoost aims to discover if a Twitter users timeline can improve the recommendation results for two recommendation algorithm techniques. The system developed using the Design & Creation methodology in combination with experiments for evaluation tries to answer this by reranking user top-k movie recommendations.

Evaluation of the system is performed with a technical experiment to measure NDCG and MAP, two common evaluation metrics. Further, the artifact is evaluated based on four design science research criteria.

1.1 Research question

1. How can incorporating personal social media data, from Twitter, into a recommender system affect top-k recommendations produced by traditional collaborative filtering algorithms?
2. How does social matching effect metrics NDCG and MAP?

1.1.1 Objective

Design and implement an artifact that gives answers to our research questions. This Artefact is a recommendation system that uses social media in the form of Twitter to improve traditional recommendation algorithms. Taking the approach of improving collaborative filtering by reranking top-k recommendations using Twitter as a data source is to our knowledge a novel problem.

1.2 The structure of this thesis

The introduction is followed by a background to the topic and used techniques. It contains both relevant information on recommender systems as well as techniques used for movie-timeline matching and common metrics for evaluating recommender system performance.

Subsequently, a method section introduces the Design & Creation methodology and how it has been applied. The method also contains the proposed experiments and what they aim at evaluating from a Design & Creation viewpoint.

Design & Implementation describes the Artefact that was implemented from the design requirements established in the design phase.

The Results section presents the performed experiments in matrices accompanied by a brief explanation.

This is followed by an analysis discussion and conclusion of the results and ends with identified gaps for future work.

2. Background

2.1 Literature review

Although this work proposes a novel system by re-ranking top-k recommendations through social matching, studies augmenting collaborative filtering using Twitter data have been made. Similarly, studies showing the viability of latent semantic indexing (LSI) when applied to micro-blogs motivate it being the chosen technique for the timeline-movie matching procedure. In what manner Twitter has been utilized in collaborative filtering recommendation systems varies.

Nair, Moh & Moh (2016) utilize users Twitter profile to try alleviating cold-start, a problem often occurring in collaborative filtering environments. This problem occurs when a new user enters the system. When no previous ratings for the user exist, the recommender system does not know what to recommend to the user. Similar to this study, the authors model the Twitter-based dataset of MovieTweatings in order to create a collaborative filtering environment, combined with the Twitter timeline for 10 000 of the dataset's users. Utilizing the plot of movies and their corresponding genre(s), a bag of words approach was used to compare each pre-processed user tweet with a movie plot to calculate cosine similarity. If the similarity was greater than the pre-defined threshold the genres for the movie were retained. In this manner, a list mapping each user and their corresponding frequency of genres were generated. To evaluate the system the list mapping users and top-k genres were used against the ratings for users in the MovieTweatings dataset. The metrics Accuracy (correct recommendations / total possible recommendations) and MAE (Incorrect recommendations / total possible recommendations) were used for evaluation. The results show an accuracy of 72%. Although taking a different approach, using genre, the study demonstrates the viability in matching user tweets with movie keywords to generate recommendations in a collaborative filtering environment based on the MovieTweatings dataset.

Alahmadi & Zeng (2015) similarly aim to alleviate cold-start in a collaborative filtering environment recommending movies. The proposed system, H-ISTS, is based on the idea that users tend to have high satisfaction when recommendations derive from friends or acquaintances. Hence the study does not consider the timeline of users, but the intricate relationships Twitter contains. The system utilizes the user's connections on Twitter to perform multi-scale sentiment classification on tweets about movies. Thereafter so-called 'social trust' between the user and his/her connections is modeled by analyzing re-tweet actions and following/follower lists. Trust and sentiment values are then applied to the SVR model to create rating estimation. Several baselines are used in the evaluation phase.

LSI has been used to develop a plot-based recommender system to find similarity between the plot of user purchased movies and the plot of movies stored in a database (Bergamaschi, Po, & Sorrentino, 2014). They compared LDA and LSI for the best performance in the system. LSI showed the best results. Acosta, Lamaute, Luo, Finkelstein, & Cotoranu, (2017) studied the possibilities of using word2vec in sentiment analysis of Twitter data. Although their focus was not on LSI and similarity, they studied the possibility of finding term similarity between tweets. The results show that it was not efficient to use LSI to match single tweets to each other since tweets often were very limited and short which was not enough information for LSI and cosine similarity to find clear similarity results.

LSI was also used by Bansal et al (2016) to build a similar recommendation system as this work aims to do. The recommender system built by Bansal et al (2016) focuses on genre prediction to recommend movies based on matching tweets about a user's one favorite movie with genre-specific movie data extracted from IMDB using LSI and Euclidean distance. The results indicate that predictions had a 70% accuracy. The difference between Acosta et al (2017) and Bansal et al (2016) is that the later used a Twitter data corpus consisting of a large number of tweets instead of single tweets which proved to be enough data for LSI and similarity matching. This shows that LSI can be efficient at finding document and term similarity between Twitter data and movie data if enough Twitter data is used.

2.2 Recommender systems

2.2.1 Overview

Traditionally there are two major categories within recommendation system (RS) techniques, content-based and collaborative filtering. Although taking different approaches, both share the common aim of predicting what items a user may like from a group of candidates in order to recommend relevant items to the user.

Two collaborative filtering techniques, Matrix Factorization (MF) and k-nearest neighbors (KNN) are used in this work as underlying algorithms for SwarmBoost. Even though this study has chosen collaborative filtering as its preferred technique, both content-based and collaborative-filtering are presented below in the hope of increasing the understanding of the overall concept of recommendation systems.

2.2.2 Content-based filtering

Content-based filtering relies on the attributes of items to filter out items from a set of candidates. It is based on the idea that if a user liked an item I , it is deemed likely that he will enjoy items with high similarity to that of I . It uses the attributes of the items to make its predictions, thus relies heavily on rich metadata. This makes it suitable for recommending items where this is the case, such as news articles and publications. To make its predictions it models the relationship between items and calculate their similarity (Aggarwal, 2016). Figure 1 below depicts the concept of Content-based filtering. Squares represent items in the recommender system. For instance, in the domain of e-commerce squares would here represent products sharing a high similarity.

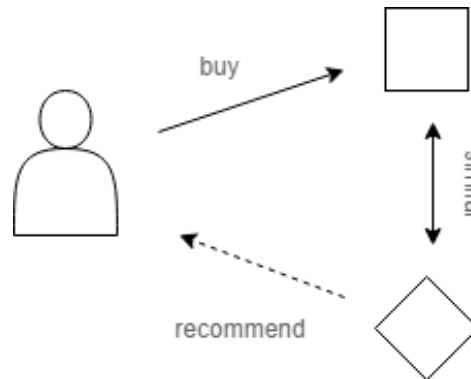


Figure 1. Depicts the concept of content-based filtering.

2.2.3 Collaborative filtering

Collaborative filtering is based on the idea that similar users, i.e. users who have liked similar items in the past, probably have a similar taste. If let's say user **A** and **B** have a high similarity of past ratings, an assumption can be made that items rated highly by **B** can be recommended to user **A**. Recommendations are generated from the utilization of matrix mapping users and items together. This 'user-item matrix' contains the behavior of each user, i.e. rating given to items in the past. Collaborative filtering is suited for environments where item metadata is limited, such as in the scenario of recommending music or movies. Generally, Collaborative filtering is more accurate than the content-based approach presented above (Koren, Bells, & Volinsky, 2009), although suffering from the often referred to the cold-start problem. This problem arises when a new user or item enters the system. When no past ratings are available, the system cannot generate accurate recommendations (Isinkaye, Folajimi, & Ojokoh, 2015).

In the case of this study, the cold-start problem was not deemed reason enough to choose a different approach due to the possibility of mitigating the problem by removing cold-start users from the used dataset. Collaborative filtering has also been heavily studied and improved and is today used in many large-scale recommendation environments motivating the choice of the technique as the choice of algorithm for our research.

Collaborative filtering is further divided into the two sub-categories of memory-and model-based techniques. One method from each of these sub-categories, used by SwarmBoost is presented below. Figure 2 depicts the concept of collaborative filtering.

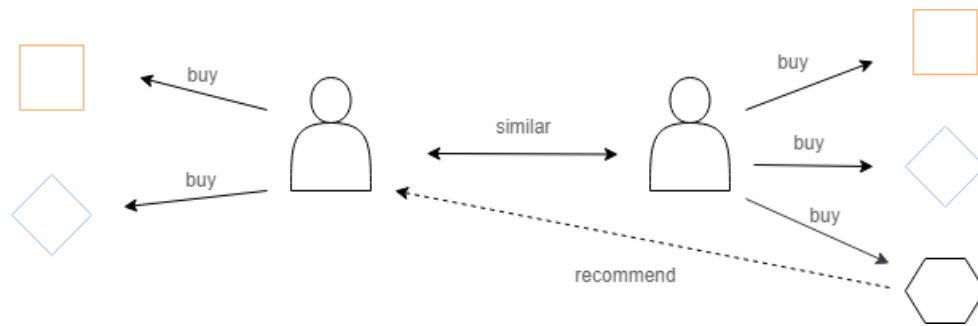


Figure 2. Depicts the concept of collaborative filtering.

2.2.4 Model-based collaborative filtering using Matrix factorization

As users in a movie recommendation environment only rate a very small part of the items in the user-item matrix, it is common that collaborative filtering is affected by data sparsity. Matrix factorization has shown to be effective in mitigating this issue. This is achieved by reducing the user-item matrix into much smaller matrices to uncover latent features in the dataset.

The decomposition results in an association between each user and a user-factors vector, similarly each item gets associated with an item-factors vector. Predictions for how a user would rate an item can then be made by calculating the inner product. This work uses Weighted Matrix Factorization (Hu, Koren, & Volinsky, 2008), a modification of standard Matrix Factorization as it has shown good results for the dataset during initial experimentation.

2.2.5 Memory-based collaborative filtering using KNN

Memory-based collaborative filtering, often referred to as neighborhood-based filtering can be either user or item based. In this work the user- based approach is used and is, therefore, the one being described in further detail.

The user-based approach uses historical user rating data to compute the similarity between users to produce its item recommendations. To compute similarity the user-item matrix described under section collaborative filtering is utilized. Users with the high similarity of historical ratings are considered part of the same neighborhood. Recommendations for a user can then be made from items not rated by the user, but by users sharing that same neighborhood. The main advantage of this technique is its simplicity, making it easy to implement and debug. On the other hand, the disadvantage of this technique is that it is prone to data sparsity. This means that if no neighbors of a user have rated item I, it is not possible to recommend an item I to the user (Aggarwal, 2016).

2.2.6 Twitter as a data source

Twitter, with its 326 million active users as of 2018 (Twitter, 2018) is a rich data source. Opinionated texts are regularly shared by users on various topics and events.

A large extent of this data published on the site is public and available. Accessing Twitter programmatically is straightforward through the official Twitter API and established open-source libraries making it a common choice as a data source in both private projects and academic research. Although research using Twitter user data in a collaborative filtering environment has been done, it has mainly been to address the cold-start problem (Alahmadi & Zeng, 2015; Nair et al., 2016).

2.3 LSI

2.3.1 Vector space model

Natural language processing (NLP) is needed in the part of the recommender system that matches social data with movies. A document containing terms can be represented in several ways when it comes to NLP. A naive approach is the Bag of words model where terms are represented by an unordered list of the words (Zhang, Jin, & Zhou, December 2010). This model does not take word positions or semantic context into account. Only the occurrence of the term in the document is represented.

	Tweet1	Tweet2	Tweet3
Term1	4	0	2
Term2	2	1	4
Term3	0	2	2

Figure 3. A Document term matrix consisting of four documents and three terms. The weights for each term are the frequency of occurrence.

A more complicated approach is the vector space model. Here the document and its terms are represented as multidimensional vectors where each dimension is a term found in the vector (Salton, Wong, & Yang, 1975). Figure 3 represents a document-term matrix. If the term exists in the document, its value in the matrix is non-zero. The document term matrix is one way to model a document, but it also suggests a different way. By modeling the terms as points in multi-dimensional term space the vector space model, which can be seen in figure 4, is created.

The value of a term is called weight and there are different ways to compute this weight. It can be a binary value representing if the term is present, it can be the frequency of occurrence or it can be a term frequency-inverse document frequency (TF-IDF) value. TF-IDF is a numerical statistic approach to setting weight and thereby significance for a term in a document (Salton, Wong, & Yang, 1975).

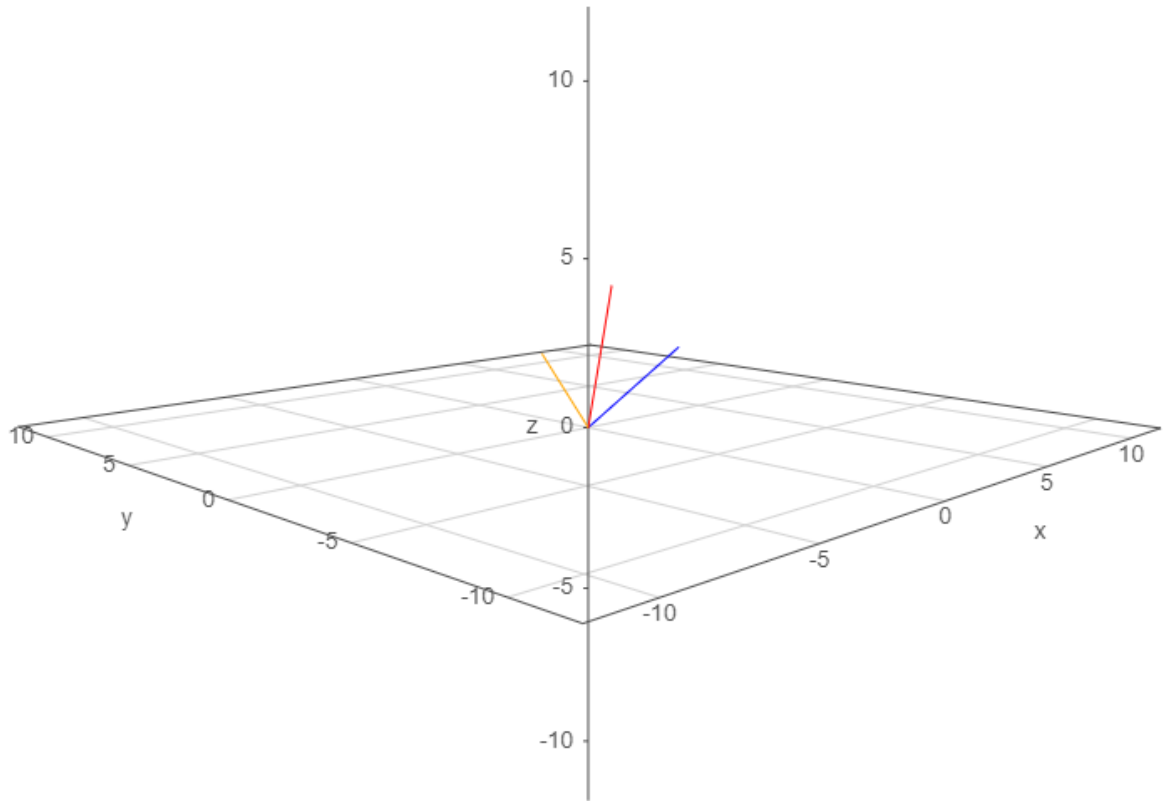


Figure 4. Graphical representation of vector space document matrix. **Blue vector** = Tweet1, **Red vector** = Tweet2, **Orange vector** = Tweet3.

2.3.2 Latent Semantic Index

Latent Semantic Indexing is a mathematical statistical technique used to extract relations between documents and queries. (Deerwester, Dumais, Furnas, & Landauer, 1990)

Converting a document into a vector space representation is called indexing. After converting the document into a vector space document term matrix C with frequency or TF-IDF weight for each term, the next step is to use singular value decomposition SVD.

This is a mathematical generalization technique used to reduce dimensionality in the document term matrix (Jagdale, Deshmukh, & S.G.Chodhary, 2016). First, the matrix is split into three different component matrices. One for each row V (the terms), one for each column U (documents) in derived orthogonal factor values and the third is a diagonal matrix containing scaling values. This diagonal matrix is needed to be able to reconstruct the original matrix by matrix-multiplication. By zeroing out some the lower order values in the diagonal matrix before reconstructing the matrix C' the co-occurring terms come closer together and the resulting matrix C' becomes more compact and efficient than C .

$$C = U\Sigma V^T$$

2.3.3 Cosine similarity

Calculating the similarity between new queries and the document is done by adding the query as a new document and calculating the cosine similarity from the resulting query vector and the other vectors (Huang, 2008).

This is done by calculating the cosine angle of the two vectors. For example the cosine angle between two vectors in figure 4.

$$\text{similarity} = \cos(\theta) = \frac{A * B}{|A||B|}$$

Given two vectors A and B representing a document and a query, the cosine angle is calculated through dot product and magnitude of the vectors resulting in 1 as the maximum correlation, 0 when there is no correlation, and -1 if there is a negative correlation.

2.4 Evaluating recommender systems

The following section describes different metrics used to evaluate recommender system performance.

2.4.1 Item ranking prediction

Recommendation systems try to find items to recommend for a user. This can either be done through predicting a rating that a user would give or by predicting if the user would use the item at all. (Ricci, Rokach, & Shapira, 2010)

A list of prediction items can be assembled through item prediction. The ranking of the items in the list shows the highest deemed relevant items in the top.

Item prediction can be done either online or offline. In offline prediction, a data set consisting of usage history is collected and used. One user is selected and some of the usage history is hidden. The recommender system is then asked to predict the items that were hidden. With the items in the history that are not hidden as input, the recommender system tries to predict the hidden items.

In the offline scenario, it is assumed that items not in the collected dataset are items the user is not interested in, even if the user potentially would have been interested but never knew about the item.

The following section describes the Mean average precision (MAP) and Normalized Cumulative Discounted Gain (NDCG). Two metrics from information retrieval that are used to evaluate the ranking of the predicted items.

2.4.1.1 Mean average precision

Mean average precision uses both precision and recall. (Manning, 2008)

Precision is the amount of correct recommended items (true positive) divided by the amount of all recommended documents (sum of true positive and false positive). This metric is used to show the probability of correct items recommend that are relevant to the user's preference. In other words, how many of total recommended where correct.

Recall is the amount of correct recommended items (true positive) divided by the sum of all correct items (true positive) and all false negative items. This metric is used to show the probability of a relevant item being recommended by the system. In other words, how many of the total possible correct items were recommended to the user.

Average precision considers the ranking order in the recommendation list. This is achieved by plotting a curve where precision $p(r)$ is a function of recall.

k is the rank of the item in the list. n is the number of items in the list, $P(k)$ is precision at cutoff k in the list $rel(k)$ is an indicator function of the item at k is relevant.

$$AveP = \frac{\sum_{k=1}^n (P(k) \times rel(k))}{\text{number of relevant documents}}$$

Mean average precision is the mean of the average precision scores for all item recommendations for all users observed when evaluating the recommender system. (Manning, 2008)

$$MAP = \frac{\sum_{q=1}^Q AveP(q)}{Q}$$

2.4.1.2 Normalized Cumulative Discounted Gain

Normalized Cumulative Discounted Gain NDCG is an information retrieval method that is used to measure the correctness of the ranking produced by a recommendation system. (Järvelin & Kekäläinen, 2002).

Cumulative gain is a predecessor to Discounted cumulative gain.

$$\text{Cumulative Gain at } p = CG_p = \sum_{i=1}^p \text{rating}(i)$$

Where $rel i$ is the graded relevance of the result at position i .

Cumulative gain is a measure of the sum of graded relevancies. It does not take the ranking of the results into consideration.

Discounted cumulative gain (DCG) takes the ranking of the results into consideration by penalizing relevant documents being lower ranked in the result.

$$\text{Discounted } CG_p = DCG_p = \sum_{i=1}^p \frac{\text{rating}(i)}{\log_2(i+1)}$$

DCG accumulated at a particular rank position p

To acquire a measure that is not only relevant for one prediction list and one correct result query the DCG need to be normalized to a value between 0 and 1.0. This normalized discounted cumulative gain (NDCG) is achieved by producing a maximum possible DCG through position p , an ideal DCG (IDCG) and dividing the produced DCG with the IDCG.

$$\text{Ideal } DCG_p = IDCG_p = \sum_{i=1}^{|\text{REL}|} \frac{\text{rating}(i)}{\log_2(i+1)}$$

$$\text{Normalized } DCG_p = \frac{DCG_p}{IDCG_p}$$

3. Method

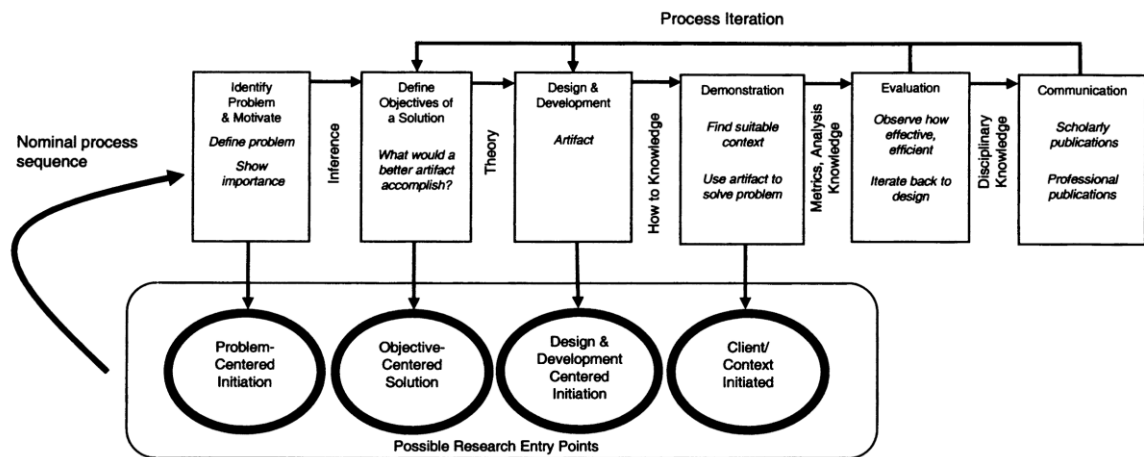
3.1 Design creation methodology

The main research method used for answering the research questions stated earlier is Design and creation since all of our research questions can be answered through design and creation of a recommender system artifact through proof of concept demonstration and different evaluation metrics experiments.

Peffers, Tuunanen, & Rothenberger, (2007) define the purpose of Design research as *"Design science creates and evaluates IT artifacts intended to solve identified organizational problems."*(p.49) The field of design research has several proposed processes models to apply for design research. Archer, Eekels, J. Roozenburg, and Hevner all propose different process models with varying degrees of details and steps (p.53). But they all share similarities.

Peffers, Tuunanen et al (2007) propose a unified methodological guideline process that is a result from studying previous design research process models. By applying the model on four case studies the proposed process model is proven efficient and versatile. The process model itself consists of six different activities with four different entry points into the process depending on what is most applicable.

Figure 5: The Design & Creation unified methodological guideline process.



Source: (Peffer, Rothenberger, Tuunanen, & Vaezi, 2012)

In the following section, each Activity is described accompanying a small description of how the activity was applied in the used research method. The process model is applied in a waterfall sequence without iteration since enough results were achieved. A problem-centered entry point was most applicable since the nature of the research question and objective of the research was centered around the problem of integrating social media into recommender systems.

3.1.1 Activity 1

Description

“Problem identification and motivation. Define the specific research problem and justify the value of a solution. [...] Justifying the value of a solution accomplishes two things: Motivates the researcher and the audience of the research to pursue the solution and to accept the results and it helps to understand the reasoning associated with the researcher's understanding of the problem” (Peffers, Rothenberger, Tuunanen, & Vaezi, 2012).

Application

The problematization identified in Activity 1 is the following:

Recommendation systems are an important part of revenue generation and user experience. The growing number of items in the e-commerce market creates a need for the best possible recommendation system to maximize and ensure revenue generation for the businesses and to ensure the best possible service and user experience for customers.

Cookies and browsing history are widely used in e-commerce recommendation systems but social media activity is an untapped source of user information when it comes to recommendation systems. Extraction of useful user information from a user's social media posts is an uncomplicated way for a user to give more information about their preferences to a recommendation system. Much research has been done on applying social media to collaborative-filtering based recommender systems for alleviating the cold-start problem, yet research focusing on improving collaborative filtering performance is uncommon.

3.1.2 Activity 2

Description

“Define the objectives for a solution. Infer the objectives of a solution from the problem definition and knowledge of what is possible and feasible. The objectives can be quantitative, such as terms(?) in which a desirable solution would be better than current ones, or qualitative, such as a description of how a new Artefact is expected to support solutions to problems not hitherto addressed. “ (Peffers, Rothenberger, Tuunanen, & Vaezi, 2012).

Application

Relevant objectives defined during Activity 2 are the following:

Creating a recommendation system that uses personal social media data for a user in the recommendation process in a way that improves traditional existing recommendation systems. This is both a qualitative and quantitative objective since the fact that the system uses social media data is qualitative and the objective of actually improving existing recommendation system results is a quantitative objective.

3.1.3 Activity 3

Description

“Design and development. Create Artefact. Such Artefacts are potentially constructs, models, methods, or instantiations. Conceptually, a design research Artefact can be any designed object in which a research contribution is embedded in the design.” (Peffer, Rothenberger, Tuunanen, & Vaezi, 2012).

Application

During activity 3 the Artefact that is used to fulfill the objectives of activity 2 is designed and a prototype is implemented to be used as proof of concept. Further design requirements and design decisions are presented in the Design phase section. The Implementation itself is presented in the implementation section.

3.1.4 Activity 4

Description

“Demonstrate the use of the Artefact to solve one or more instances of the problem.” (Peffer, Rothenberger, Tuunanen, & Vaezi, 2012).

Application

To demonstrate the Artefact as described in Activity 4 the system is used in an experiment scenario with MovieTweatings, a dataset including movie ratings with IMDb IDs for the movies and Twitter user IDs. The dataset is split chronologically in order to predict user ratings and get results relevant to the objectives defined in Activity 2.

3.1.5 Activity 5

Description

“Evaluation. Observe and measure how well the Artefact supports a solution to the problem. This activity involves comparing the objectives (so we need to define this in objectives) of a solution to actual observed results from use of the Artefact in the demonstration. It requires knowledge of relevant metrics and analysis techniques. Depending on the nature of the problem venue and the Artefact, evaluation could take many forms. It could include items such as a comparison of the Artefact's functionality with the solution objectives from activity 2, objective quantitative performance measures such as budgets or items produced, the results of satisfaction surveys, client feedback, or simulations. It could include quantifiable measures of system performance, such as response time or availability. “

(Peffer, Rothenberger, Tuunanen, & Vaezi, 2012).

Application

Evaluation is performed in order to discover if and how well the proposed Artefact solves the proposed problem. Four of the terms presented by Hevner (2004), deemed most relevant, were selected to evaluate the instance. The following were selected: functionality, consistency, accuracy, and performance. What each of them denotes is presented in section 3.3.1.

A common evaluation technique in DS research, technical experiments are used for their evaluation (Peffer, Rothenberger, Tuunanen, & Vaezi, 2012). The experiments are also performed on different underlying collaborative filtering algorithms and randomization. The experiments and what they aim to evaluate is presented in greater detail in section 3.3.2.

3.1.6 Activity 6

Description

“Communicate the problem and its importance, the Artefact, its utility and novelty, the rigor of its design, and its effectiveness to researchers and other relevant audiences such as practicing professionals, when appropriate. In scholarly research publications, researchers might use the structure of this process to structure the paper, just as the nominal structure of an empirical research process (problem definition, literature review, hypothesis development, data collection, analysis, results, discussion, and conclusion) is a common structure for empirical research papers. Communication requires knowledge of the disciplinary culture.” (Peffer, Rothenberger, Tuunanen, & Vaezi, 2012).

Application

This Thesis paper is the first and foremost way of communicating the importance and knowledge stated in Activity 6.

3.2 The data

3.2.1 MovieTweetings

The dataset used as input for the recommender system was MovieTweetings, a dataset presented by (Dooms, Pessemier, & Martens, 2013). The significant difference between the MovieTweetings dataset and other movie rating datasets, like MovieLens for example, is that the data is obtained through extraction of IMDB movie user ratings from Twitter posts. After reviewing a movie on IMDB it is possible to share your review on Twitter through a share function. The tweet generated by this service on IMDB follows a uniform format which makes the extraction of data possible.

Since this dataset extraction project is automated and still running, new ratings are added every day. At the time of writing the dataset consists of 771,633 ratings from 56,808 users on 33,186 unique movie items.

New ratings make this dataset current and updated since it contains reviews of current and popular movies in comparison to static datasets like MovieLens.

The dataset not only contains user movie ratings and movie ID but also the Twitter ID of the user who rated the movie. This makes it possible to identify the user on Twitter and to extract the user's tweets from the Twitter API.

It is then possible to match a corpus of words for each movie with the corpus of words for all of the user's tweets to determine how many similarities there between the two corpora.

3.2.2 Pre-processing the dataset

The data used for evaluating the recommendation system consists of 100 000 ratings picked from the MovieTweetings dataset.

During extraction of the 100 000 ratings, cold start user and movies with less than 30 occurrences in rating were discarded. The remaining ratings were split between a train dataset and a test dataset with an 80% 20% split which is a common split ration (Ricci, Rokach, & Shapira, 2010). The train and test datasets are ordered by the user and chronologically by date of the rating.

This extracted and pre-processed dataset contains 1188 users and 2867 movies. The dates of the first rating of a user in the test dataset are also extracted and saved to be used as a cut off for tweet extraction from a user timeline. This is to ensure that no ratings in the test dataset are predicted using tweets that were posted after the rating in the test data set occurred.

3.2.3 Sentiment classification

After retrieving all user tweets and filtering out the tweets posted after the first rating occurred in the test dataset, the tweets are sentiment classified into neutral, negative or positive tweets. This is done with the text blob Python module (Textblob python module, n.d.). The sentiment classifier uses a Naive Bayes algorithm and is pre-trained on a movie review dataset. New text that is added for classification is also analyzed with Naive Bayes.

Only neutral and positive tweets are added to Twitter corpus for each user. The reason for this is to ensure that aspects of a movie, that the user is negative towards, are not matched with later on in the pipeline.

3.2.4 Twitter Data pre-processing

Twitter data contains lots of irrelevant noise that is undesired for natural language processing algorithms. URLs, symbols and meaningless words are frequent in the text tweets that are extracted from Twitter. To make the Twitter data parsable and more effective in sentiment classification and social matching it needs to be cleaned first. This process is called pre-processing Twitter data.

Angiani et al (2016) present an article where they analyze 15 different pre-processing techniques that are frequently used on Twitter data before sentiment analysis. Angiani et al (2016) argue that all techniques except for dictionary spell correction improve the result of sentiment analysis following the pre-processing.

By using the following techniques from Angiani et al (2016) study we make the Twitter data as parsable and meaningful as possible.

1. replace slang and abbreviations
2. replace contractions
3. remove numbers
4. remove emoticons
5. remove exclamation marks, question marks
6. remove punctuation
7. lowercase all characters
8. remove stop words
9. apply stemming to words
10. apply lemmatization

Techniques 1-7 are described as basic techniques to filter out unwanted letters and symbols to reduce noise. Stop words are words with no meaning for sentiment and topic analysis. The list of stop words that was used in this project comes from the Terrier platform (Terrier stop list, n.d.) and is an extensive and long stop word list compared with other common stop word list found as open source on the internet.

Stemming words means to reduce a word to the simplest possible and most uniform format for all word variations. Example of this is words like greater, greatest, greatly. All of them are reduced to the word great by stemming. But a problem with stemming is that it only cuts off the word at its simplest form. It can be a problem with words like *studies* and *studying* which would become *studi* and *study*.

This can be solved through Lemmatization which not only reduces the word by cutting it at its simplest form, but it also analyses it morphologically.

3.3 Experiments & Evaluation

3.3.1 Selected DS evaluation criteria

In this section, we present selected design science (DS) evaluation criteria, as well as how they will be evaluated. According to Hevner (2004) evaluation is deemed as a “crucial” part of the Design science framework. An artifact can be evaluated by terms of functionality, completeness, consistency, accuracy, performance, reliability, usability. Except for presenting the criteria, there is limited guidance regarding how and which of these should be performed. (Peffer, Rothenberger, Tuunanen, & Vaezi, 2012)

A survey presented by Peffer et al (2012) shows that the most common evaluation techniques in DS research are technical experiments and illustrative scenarios, based on their review of 142 analyzed DS articles. To evaluate our proposed instantiation, we have chosen to follow a similar path and perform a combination of technical experiments to evaluate the four above DS criteria proposed by Hevner (2004). The chosen criteria are presented below.

Criteria 1: Functionality

Functionality denotes a general evaluation of the artifact. This is not evaluated by tests, but rather an overall assessment answering the broad but critical question “Does the solution solve the problem?”.

Criteria 2: Consistency (consistent behavior, stability)

Consistency is evaluated regarding how consistent results are for users in the dataset. Identified consistency issues include a complete lack of data for some users due to private accounts and users tweeting in foreign languages. Another consistency issue is regarding tweet quantity as some users have rich tweet history while others do not. Lack of data also holds true for some movies in the dataset, where API requests return empty.

Criteria 3: Accuracy (correctness and preciseness)

Accuracy denotes how correct and precise the system is. This is evaluated by running experiments with, as well as without the precision at k threshold.

Criteria 4: Performance

Performance is evaluated by testing the execution time of the instance.

3.3.2 Experimental overview

The experiments presented below are conducted to evaluate the instantiation and present the performance of the system. Results from the experiments will be used to evaluate the DS criteria outlined above. As stated in section 2.4 Evaluating recommender systems, multiple evaluation metrics (NDCG, MAP) are applied to the top-k recommendations before and after social matching to produce quantitative results. However, this is not the case for all

experiments. While some are well suited for metric comparison against baselines, others compare average LSI match scores of different users, such as experiment Optimal user (E5), or simply clocking the running time of the Artefact as in experiment E6 and E7. How the experiments are evaluated is further presented in the description of each experiment in the coming subsections.

3.3.3 Main instance parameters

Three parameters were tuned to find the best performing configuration for the first four experiments (E1-E4). These parameters are LSI threshold, precision at k threshold and top-k. The parameters and the purpose of the tuning are presented below.

Purpose and parameter tuning

The precision at k threshold is used to decide which users in the dataset to be considered during runtime. After the underlying collaborative-filtering has produced the top-k recommendations for a user, the precision of the recommendations is controlled by the usage of metric precision at k. This is done to safeguard against evaluating on users who don't have any correct recommendations in the top-k. Since the social matching is focused on re-ranking, at least one correct recommendation is needed in order to evaluate if re-ranking improved to top-k. When deciding on a threshold two aspects need to be considered. Firstly, a lower threshold means a larger part of users the dataset is utilized but suffers from lower precision which prohibits the social matching to perform sufficiently for these users. Secondly, a higher threshold means utilizing fewer users but a higher precision for these users. The precision matters since if the top-k consists of false recommendations only, re-ranking will not have any effect as the movies re-ranked weren't in the test data for the user, to begin with. Users who have precision at k result falling below the threshold are therefore discarded from the evaluation. Experiments will be conducted both with and without the precision at k threshold. This is done to show how the quality of the top-k effects how well the social matching performs.

The LSI threshold is contrary to the precision at k threshold controlled first after social matching for a user has been performed. Thus, contrary to the precision threshold, a user falling below the LSI threshold is still part of the used dataset. As social matching has been completed for a user u on top-k movies M , a match value for each movie is calculated. When combined and averaged it generates an average LSI match score for the user. This average LSI match score is then compared to an average LSI match score threshold. If it is below, the matching result is deemed as insufficient, and the results from the social matching are ignored. This situation can be the result of insufficient tweets, a lack of tweets in English or no movie data.

The size top-k parameter is regarding the number of movies selected for each user from the ranked recommendations produced by the underlying collaborative filtering algorithm. As with the thresholds above, multiple parameters were used, where one was decided on and will, therefore, be used for the experiments.

3.3.4 Two underlying algorithms

Experiments to be conducted are performed on different variations of the instance. These variations include the two different underlying collaborative filtering algorithms, Weighted Matrix Factorization (WRMF) and the neighborhood-based technique, KNN. This is done to uncover possible variation in the results. It is also making the results from the tests more trustworthy as they are not solely derived from one underlying collaborative filtering algorithm.

3.3.5 Randomization of top-k

Randomization of top-k is used by the first four experiments presented below (E1-E4) and is the procedure of shuffling the top-k recommendations produced by the underlying collaborative filtering algorithm for each user. This is done to uncover if the social matching module performs better than a random ranking of top-k recommendations.

3.3.6 Experiments

Experiment 1: Weighted Matrix Factorization (E1)

Experiment description

This experiment consists of running the main functionality of the instance when based on Weighted Matrix Factorization (WRMF). Design Science criteria accuracy is evaluated with metrics MAP and NDCG.

Instance:

WRMF + Social Match

WRMF Random top-k + social match

Baselines: WRMF Baseline, WRMF Random top-k

Evaluation metric(s): NDCG, MAP

Targeted DS criteria: Accuracy (correctness and preciseness)

Experiment 2: Weighted Matrix Factorization with precision at k threshold (E2)

Experiment description

This experiment is identical to experiment E1 except for the addition of the precision at k threshold. This is done to understand to what degree the precision of top-k effects how the instance performs.

Instance:

WRMF + Social Match

WRMF Random top-k+social match

Baselines: WRMF, WRMF Random top-k

Evaluation metric(s): NDCG, MAP

Targeted DS criteria: Accuracy (correctness and preciseness)

Experiment 3: KNN(E3)

Experiment description

This experiment consists of running the main functionality of the instance when based on KNN. Design Science criteria accuracy is evaluated with metrics MAP and NDCG.

Instance:

KNN + Social Match

KNN Random Top-k+social match

Baselines: KNN, KNN Random top-k

Evaluation metric(s): NDCG, MAP

Targeted DS criteria: Accuracy (correctness and preciseness)

Experiment 4: KNN with precision at k threshold (E4)

Experiment description

This experiment is identical to experiment E3 except for the addition of the precision at k threshold. This is done to understand to what degree the precision of top-k effects how the instance performs.

Instance:

KNN + Social Match

KNN Random top-k+social match

Baselines: KNN, KNN Random top-k

Evaluation metric(s): NDCG, MAP

Targeted DS criteria: Accuracy (correctness and preciseness)

Experiment 5: Optimal user (E5)

Experiment description

The aim of This experiment is to study how tweet quantity affects the average LSI match score. Is the system suitable for every kind of Twitter user or only a specific subgroup? The effect on average LSI match score by tweet quantity will be tested by splitting users tweets into subgroups. Four users who fit the criteria of tweet quantity (> 2500) whereas two write in English, one in both English and Spanish, and one writing in the Arabic alphabet were selected. The inclusion of foreign languages aims to demonstrate how the LSI match score is affected by partly, as well as fully incompatible data for the LSI similarity.

The results of average LSI match score for each user within each subset are then compared. Due to the direct comparison between average LSI match scores no baseline is used in this experiment.

Subsets:

Subset 1: 2500 tweets

Subset 2: 1250 tweets

Subset 3: 500 tweets

Evaluation metric(s): Average LSI score comparison (subset & user)

Targeted DS criteria: Consistency (consistent behaviour, stability)

Experiment 6: Execution time of Artefact (E6)

Experiment description

The running time of the system is in this test compared to the underlying collaborative filtering algorithm it is based on. This is done by performing two consecutive runs on the same dataset, timing the execution of both the recommendation systems.

Evaluation metric(s): Execution time

Targeted DS criteria: Performance

Experiment 7: Execution time of offline- versus online-social matching (E7)

Experiment description

Tweets for each user are obtained from the official Twitter API and saved to file for quick access. This is done for an obvious reason, improving running time performance. This test compares the execution time of the system in an offline- and online- matching scenario. The dataset used is not the 100 000-ratings dataset used other experiments but a 10 000-rating dataset that is extracted from Movietweetings and pre-processed the same way as the 100 000-rating datasets. The 10 000 rating dataset contains 110 users and 728 movies. The reason for using the smaller dataset is the very long time of execution in online matching.

Evaluation metric(s): Execution time

Targeted DS criteria: Performance

3.4 Method discussion

Although research using Twitter user data in a collaborative filtering environment has been done, it has mainly been to address the cold-start problem (Alahmadi & Zeng, 2015; Nair, et al., 2016). Taking the approach of improving collaborative filtering top-k recommendations using Twitter as a data source is to our knowledge a novel problem.

Combining Twitter and collaborative filtering in order to improve top-k recommendations seemed valid for research due to multiple reasons. Firstly, having access to MovieTweatings, a dataset containing over 700 thousand movie ratings being continuously updated, therefore being current, in which users are identified by their Twitter id. Secondly, the accessibility of Twitter user data through the official Twitter API, generous rate limits and an abundance of API wrapper modules, such as in this work much used Tweepy. This combination presented the right conditions to try and tackle the proposed problem.

Yet the dataset used comes with multiple drawbacks. Firstly, as users in the dataset are of different nationalities, the language of tweets varies, hindering social matching for some users. Secondly, user IDs in the dataset belong to either public or private Twitter accounts. When an attempt is made to fetch the user Twitter timeline of a private account, no tweets are returned. Nonetheless, the advantages of using the MovieTweatings dataset were considered to outweigh these drawbacks and so remained the dataset this work is based on.

Why a different social media data source such as Facebook was not chosen is mainly due to the nature of the MovieTweatings dataset. As previously stated, users in the dataset are identified by their Twitter id. Without having this link between users in the dataset and their corresponding Twitter profiles it is unclear how timeline-movie matching could have been accomplished. Furthermore, to the extent of our knowledge no alternative to the MovieTweatings dataset today exist which instead could have been used to answer the proposed research questions.

The choice to attempt improving top-k was made as it was deemed a clear objective when it comes to improving the traditional recommender systems. It is also a good approach for generating quantitative results with the use of common evaluation metrics for recommendation systems and comparison to a baseline.

The chosen evaluation metrics MAP and NDCG are deemed most appropriate because of several reasons. Since the main focus of improving the top-k recommendations is done through reranking the top-k, it is important to be able to measure how much better the recommendations are in the topmost part of the prediction list. Both MAP and NDCG take the order of recommendations into account which is important since we try to evaluate the accuracy of our top-k reranking. But the difference between MAP and NDCG is that MAP uses binary relevance and NDCG uses real-valued relevance. This means that items are not either good or bad in NDCG but how good or bad an item recommendation on a certain rank in the list is taken into account.

Our problem, being of a novel nature motivates the choice of building a system in order to answer the proposed research questions. Due to this need for implementation, Design & Creation combined with experiments were chosen as they were considered the most suitable methodologies for our purposes. According to Oates (2016), a disadvantage of Design & Creation as a research methodology is the possible need to defend that the work qualifies as actual research. It was therefore of high importance that a clear methodology was followed and presented, and that the created system was evaluated by terms recommended by the methodology. As creating a system was considered necessary to answer the research questions, following the methodology helped us avoid this pitfall.

Yet it is of importance to consider whether alternative methodology can answer the proposed research questions. One considered approach was the application of an additional methodology, performing a Systematic literature review followed by Design & Creation. The results from the first methodology would then, as done in this study but to a greater extent be used as a knowledge base, influencing decision making during Design & Creation. However, due to time constraints posed on our work, this approach was deemed not viable. Nonetheless, this approach could potentially have positively influenced the design choices of the prototype system presented in this work.

Alternatively, a greater focus could have been put on Twitter as a data source and how suited it is for augmenting a movie recommender system. To answer this, data generation could be conducted, followed by analysis to uncover how common movie references are on Twitter, as well as average tweet length and number of tweets containing words with sentiment information. The results from a work of this nature would potentially give further insight on whether Twitter is a good fit for researchers considering implementing Twitter timelines to improve recommendations.

However, for multiple reasons this approach was not chosen. Firstly, previous research implementing Twitter data in a recommender system show positive results, motivating further implementation. Secondly, as the problem statement of improving top-k is of a novel nature, Design & Creation was deemed most suitable.

4. Design & implementation

4.1 The Design Phase

In this section, the design phase that was part of activity 3 is described.

The design requirements for the movie that could be derived from the objectives specified during activity 2 were identified as the following:

- A recommendation system that takes social media data into consideration for movie recommendations.
- Improvement of existing traditional recommendation methods that can be measured.

Many different approaches were considered but in the end, an architecture that takes both user movie ratings and user social media data was decided to be the best design decision in order to be able to directly apply social media information extracted from tweets on recommendations from existing recommendation techniques.

Both personalized and non-personalized data extraction methods were considered during design.

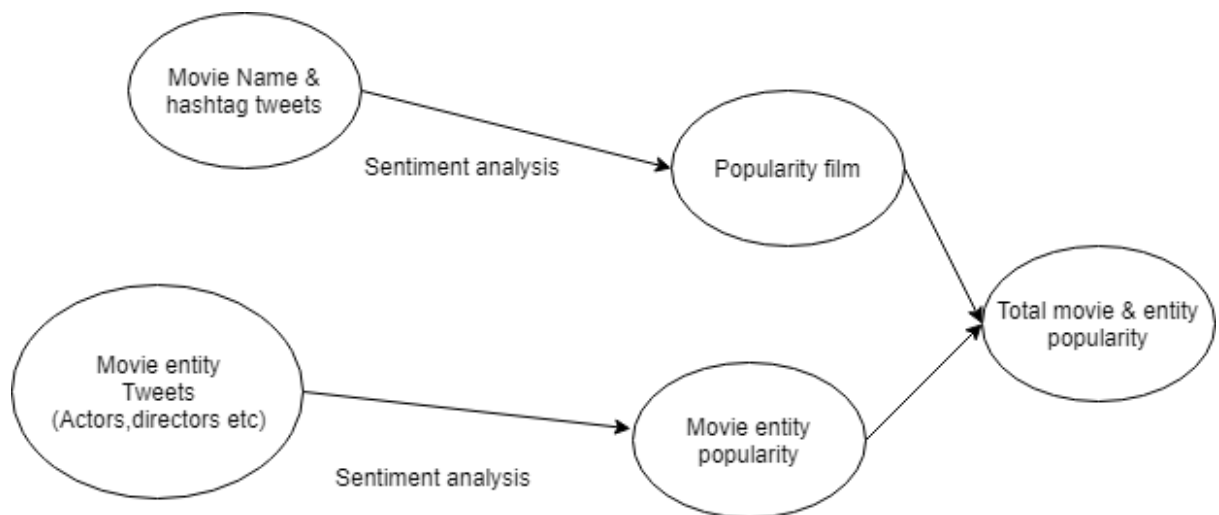


Figure 5 Non-personalized movie popularity extraction from Twitter.

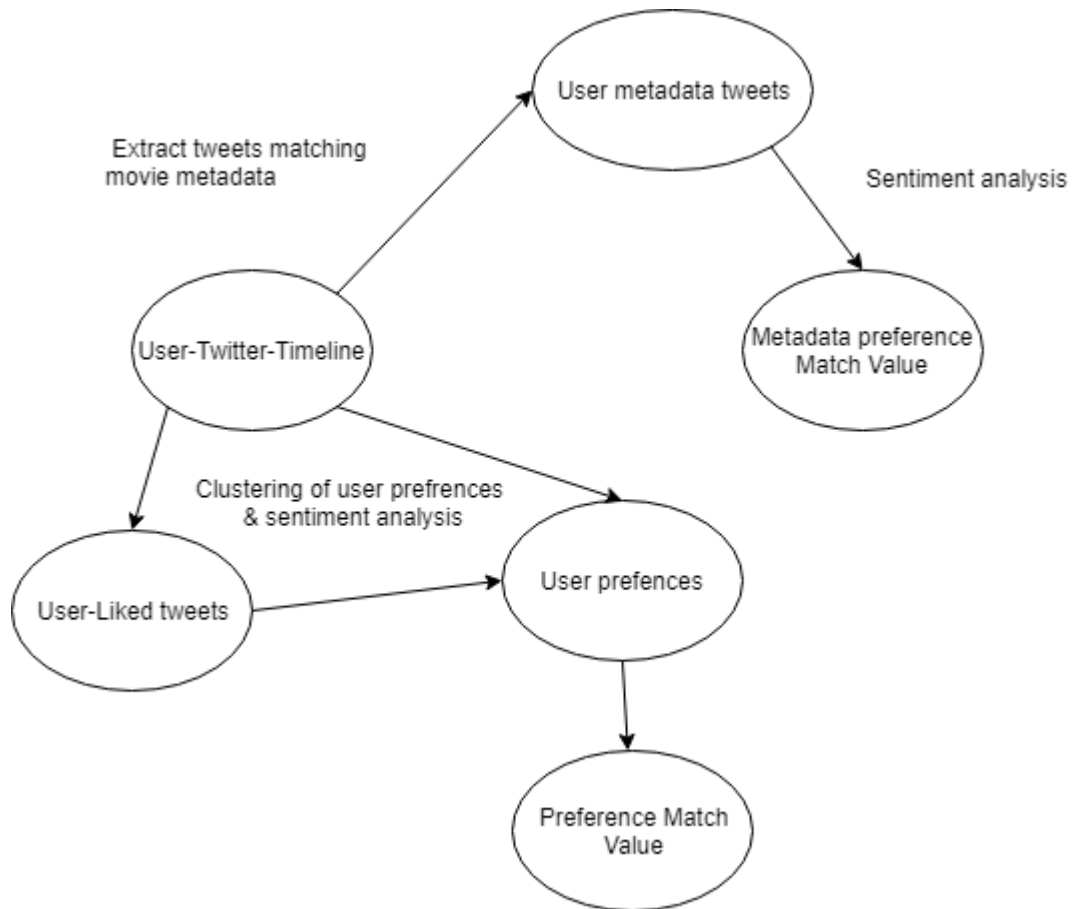


Figure 6. Personalized movie-matching techniques.

All of the techniques for extracting match values shown in figure 1 and figure 2 were deemed potentially useful and worth implementing.

One approach was to extract tweets containing words from movies metadata, such as actors, producers or directors, and use sentiment analysis to get a preference match value for the metadata.

Another approach was to extract all user tweets or user liked tweets and perform clustering of user preferences and sentiment analysis to match identified preferences with specific keywords and properties of movies to get a social match score.

During the design phase extracting topics through topic clustering and matching these topics with the movie, keywords were considered to be the best way to match the users Twitter data with a movie.

Later the matching via topic clustering approach was identified to not give any useful results during implementation. It was abandoned, and further literature study showed a potential solution through LSI and cosine similarity. This was then applied matching between user tweet corpus and movie keyword corpus. The approach gave the developed system

novelty since, to our knowledge, no other previous research has used LSI and cosine similarity to directly match personal Twitter social media data with a movie keyword corpus.

The non-personalized movie and movie entity popularity was implemented but since the non-personal tweets extracted through word querying were 14 days old at most, for free Twitter developer accounts, it was not possible to use them when trying to predict ratings that were dated much earlier. If an enterprise developer account would have been used this technique could have potentially been integrated into the system.

4.2 Implementation

The following section describes the Artefact that was implemented from the design requirements established in the design phase. All the data used as input for the recommender system and how it is pre-processed is also described.

4.2.1 The Recommender

The recommender system consists of two main modules, a collaborative filtering algorithm that uses an interchangeable recommender algorithm provided by the MyMediaLite library (Gantner, Rendle, Freudenthaler, & Schmidt-Thieme, 2011), and a social matcher module that matches movie data with the users Twitter data. The total recommender system takes the movie ratings with user Twitter IDs and movies as IMDB IDs as input and gives an improved list of Top-k recommendations for each user as output.

The collaborative filtering generates a list of recommendations for each user. This list contains all the recommendations in a certain order depending on the score each recommended movie received from the matrix factorization algorithm.

From this recommendation list a top-k list, where k is the number of items in the list, is extracted and sent to the social matching module. The social matching component evaluates the social match between each movie in the top-k recommendations and the users Twitter timeline resulting in a match score. By sorting the list on the new score for each movie the top-k is reranked into a potentially improved top-k recommendations list.

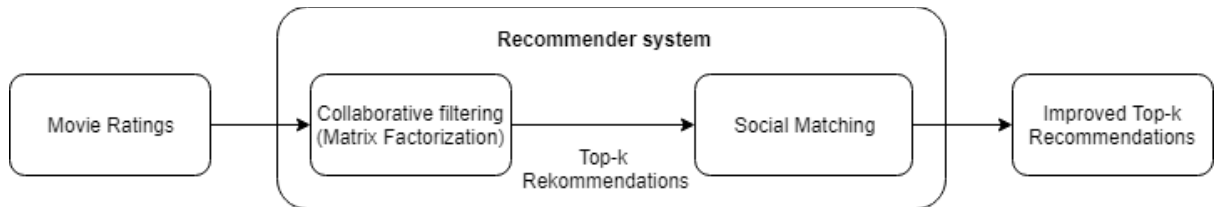


Figure 7. System diagram depicting the overall architecture.

MyMediaLite

The collaborative filtering algorithms used to get recommendations for each user are done with algorithms from the MyMediaLite C# library (Gantner, Rendle, Freudenthaler, & Schmidt-Thieme, 2011). This library is open source and contains a collection of recommendation algorithms. Code for the evaluation metrics MAP and NDCG are also provided by this library.

4.2.2 Social Matching

The social media matching module of the recommender system matches all movies in the top-k recommendation list passed on to it with the pre-processed tweets from the user. The dataflow inside the social matching module is shown in Figure 8. The social matching is done by assembling separate corpora of words for both a user's tweets and for each movie. The corpus of words for each movie is created through a pipeline of different Web APIs. The user tweet corpus is pre-processed as described in section 4.1.1. Each movie corpora is matched with the user's tweet corpora using LSI and cosine similarity. The list of resulting scores for each movie are sent back to the recommender system and the top-k is rearranged depending on the matching score of each movie.

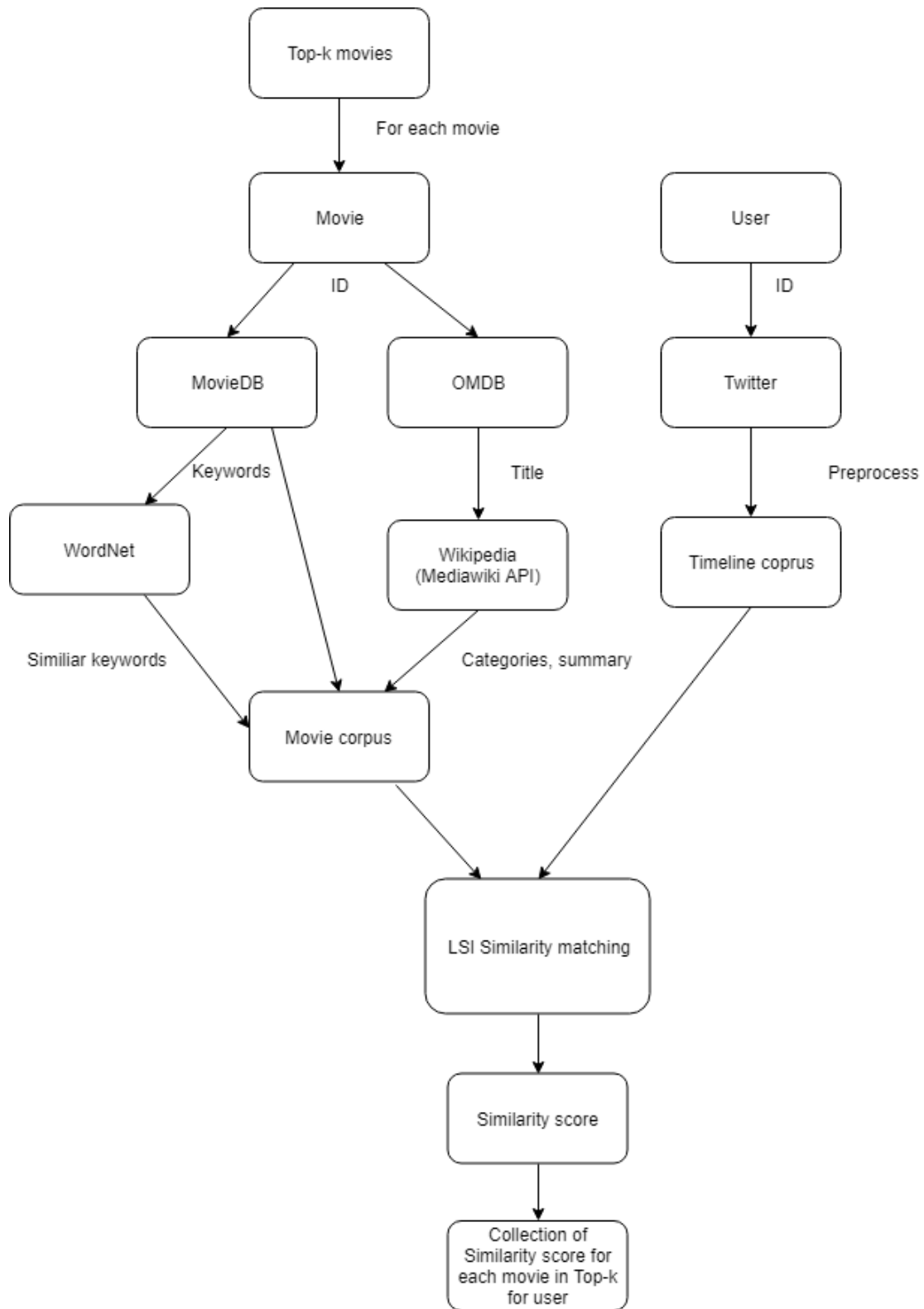


Figure 8. Social match module diagram depicting the data pipeline and social matching process in the social module.

Movie Corpus

For each movie, a corpus of words that describe the movie is generated. The pipeline for this process consists of OMDb, MovieDb, Wordnet, and MediaWiki. All of them are accessed through rest APIs that return data requested.

In a final step for creating the movie corpus, the collection of words extracted from the different web services need to be pre-processed with the same techniques as user tweets. This is necessary to ensure that the LSI algorithm has the best preconditions for matching similarity in the two corpora.

OMDB

OMDB is a user-generated Movie database Web service (Online movie Database API, n.d.). It can be accessed through a rest API. OMDb can be used to obtain movie metadata like title, year, genre, directors, actors, etc. It accepts the IMDB movie ID, that is found for each movie reviewed in the MovieTweetings dataset, as a query parameter for requests. This is used to convert the IMDB movie ID to a Title and year for each movie.

MovieDB

MovieDB is also a user-generated movie database that can be accessed through a REST API (The Movie database API, n.d.). It is used to request movie keywords. These are a collection of words that describe the movie.

WordNet

WordNet is a lexical database that can be used to get similar words, not only in the sense of word forms and strings of letters but also in the senses of words themselves (Miller, 1995). Through an HTTP API, similar words can be requested for a query. By querying the WordNet API with the keywords for each movie, the keywords are expanded to a bigger collection of words representing the movie.

MediaWiki

MediaWiki is the engine software on which Wikipedia is built on (MediaWiki API, n.d.). The MediaWiki API enables access to the data on Wikipedia through a web service. By querying the API with the Movie title converted from the movie ID through OMDb corpus of words for each movie is further expanded. The data that is added to the corpus of each movie from Wikipedia through MediaWiki API is categories and summary.

Categories are groupings of related pages to the Wikipedia movie article and summary is the summary of each Wikipedia movie article that can be found at the top of a Wikipedia article.

5. Result

In this section, results are presented in the form of general statistics about the system observed after running the dataset and experiments. Each experimental setup is presented in table form. Each table of results is accompanied by a brief description of the experimental setup and its result. Indices showing a result of specific interest are shown in bold.

5.1 General statistics of the systems

The 100 000-ratings dataset used for all experiments except for experimental setup 7 contains 1188 users and 2867 movies. From these 1188 users' tweets could only be extracted and used from 724 users since the other user's Twitter accounts were either private or deleted

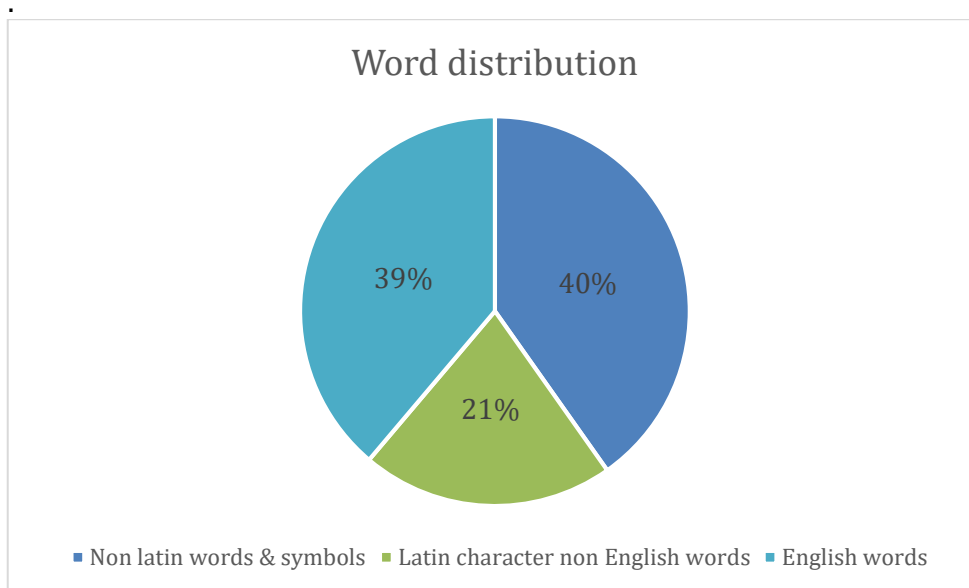


Figure 9 distribution of languages in tweets extracted

A total of 18 769 296 words, numbers, symbols, emoji, and web addresses were found in the tweets extracted from Twitter for the 724 users who did not have a private or deleted Twitter account. From these 18 769 296 only 11 220 676 were Latin character words and only 7 285 873 of these Latin character words were identified as English words through English dictionary lookup.

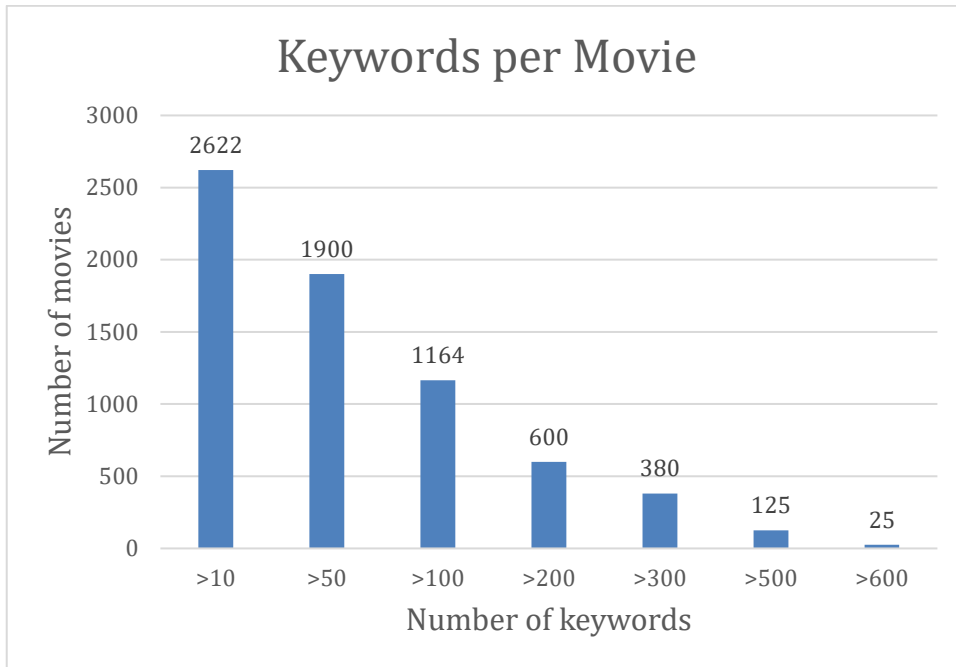


Figure 10 Diagram depicting amount of movies with how many keywords

The diagram in figure 10 shows the number of movies which had a number of keywords that is greater than the value on the x-axis. From the total 2867 movies, 2622 movies in the dataset got at least 10 keywords. 245 movies got less than 10 keywords. On the other side of the spectrum, 25 movies got at least 600 keywords and one movie got 900 keywords extracted through the movie corpus creation pipeline.

5.2 Experiment results

All experiments are performed with a Top-k of 20 and an LSI threshold of 2,5. Experiments without precision at k threshold are performed on all 1188 users in the dataset. Experiments with precision at k threshold are performed with a 0.05 threshold and on only the number of users left after the filtering. This amount varies depending on the algorithm.

Experiments were performed on a computer with an Intel core i5 2.30Ghz processor and 8GB RAM. The internet connection used for experiments has a measured performance of 76.8 Mbit/s.

Experiment 1: Weighted Matrix Factorization (E1)

Table 1: Results for WRMF algorithm and social matching.

System variation	MAP	NDCG
Social Match	0,01579382	0,04947823
Baseline	0,01554792	0,04907862
social match w. Random top-k	0,01544806	0,04955074
Baseline w. Random top-k	0,01565387	0,04953103

Table 1 shows the MAP and NDCG metrics for running the system with the 100 000 rating datasets and the weighted matrix factorization algorithm. Baseline metrics show the results of the algorithm without social matching compared to metrics for running the dataset with WRMF and social matching. Metrics for Baseline with random top-k are compared to running social matching on the randomized top-k produced by WRMF.

Experiment 2: Weighted Matrix Factorization with precision at k threshold(E2)

Table2: Results for WRMF and social matching with precision at k threshold.

System variation	MAP	NDCG
Social Match	0,0395695	0,1224214
Baseline	0,03881587	0,1214256
social match w. Random top-k	0,03831337	0,1211768
Baseline w. Random top-k	0,03606809	0,1196291

Table 2 shows MAP and NDCG metrics for the system with the WRMF algorithm and with precision at k threshold 0.05. The filtering with the precision at k threshold resulted in 486 users of the 1188 total users in the dataset. Here as well baseline WRMF is compared to WRMF with social matching and a baseline with randomized WRMF top-k compared to randomized WRMF top-k with a social match.

Experiment 3: KNN (E3)

Table 3: Results for the KNN algorithm with social matching.

System variation	MAP	NDCG
Social Match	0,01689172	0,05260321
Baseline	0,01676626	0,05238869
social match w. Random top-k	0,01689433	0,05266033
Baseline w. Random top-k	0,01516998	0,05094282

Experiment 4: KNN with precision at k threshold(E4)

Table 4: Results for KNN with social matching and precision at k threshold.

System variation	MAP	NDCG
Social Match	0,0400777	0,1248551
Baseline	0,03983662	0,1244755
social match w. Random top-k	0,03763609	0,1235119
Baseline w. Random top-k	0,03835126	0,122566

Table 4 shows MAP and NDCG metrics for the system with the KNN algorithm and precision at k threshold of 0,05. The filtering with the precision threshold resulted in 500 users of the 1188 total users in the dataset.

Experiment 5: Optimal user (E5)

Table 5: Results for Optimal user experiment

User	$N=2500$	$N=1250$	$N=500$
User 1 (Lang: English)	16.317	7.048	3.154
User 2 (Lang: English)	19.831	7.834	4.358
User 3 (Lang: Spanish & English)	12.070	–	–
User 4 (Lang: Arabic)	1.096	–	–

Table 5 shows LSI matching scores for four different users. User 1-2 are English language users. Users 3 uses both Latin and English in tweets and User 4 uses only Arabic letters.

Experiment 6: Execution time of Artefact (E6)

Table6: Results from Baseline vs social match execution time experiment.

System	Execution-time (HH:MM:SS: MS)
Instance (KNN w. Social matching)	00:00:13.56
Baseline (KNN)	00:06:29.01

The table above shows results from Experiment 6 where execution time between KNN without social matching as a baseline is compared with KNN with social matching. This is an offline experiment since user tweets and movies needed already were saved to file before the experiment.

Experiment 7: Execution time of offline- versus online-social matching (E7)

Table 7: Results from Offline versus online social matching execution time.

System	Execution-time (HH:MM:SS: MS)
Instance (Offline)	00:00:30.20
Instance (Online)	02:24:22.98

The table above shows the execution time of running the recommender system online without previously saved data for users and movies, compared to the execution time of the system with user Twitter data and movie data saved to file. The collaborative filtering algorithm used is KNN. Unlike all other experiments the dataset used is a 10 000 ratings dataset instead of the 100 000 ratings dataset. Offline results for the experiment are 30.2 seconds and online results are 2 hours 24 minutes and 24.98 seconds.

6. Analysis

6.1 Prototype implementation analysis

The prototype design that was finally implemented was a result of trial and error since not all features and approaches showed to be useful after implementation. The non-personalized matching was abandoned, and the technique used to find similarity between the movie and tweet corpus changed from the direct matching of words to LSI and cosine similarity. These changes to the design were necessary to create a prototype recommender system with the desired functionality of using social media data. It was also necessary since it gave the best-expected circumstances to achieve improvement of the top-k.

The overall architecture of the prototype system has several advantages. Separating the recommender algorithm module and the social matching into two separate modules connected by REST APIs has the advantage of replacing either of the modules with a different implementation that does not have to be in a specific programming language. Another advantage of the separation and loosely coupled modules is that a different recommender algorithm module with new recommender algorithms can easily be exchanged in place of the recommender algorithm module used in this prototype. Separating the two modules can also be used to implement improvements in the future to increase performance since computation can be distributed on several servers through concurrent programming and asynchronous API request. This was not implemented in the prototype, but it was a possibility that could have been implemented if there had been more time.

The pipeline of APIs used to create the movie corpus in the social matching module is deemed sufficient since it produces a large and descriptive enough movie corpus for a majority of movies as can be seen in figure 10. Expanding the keywords gathered from MovieDB (The Movie database API, n.d.) with synonyms from WordNet (Miller, 1995) significantly improved the movie corpus since more descriptive words with the same meaning could be used in the social matching through LSI and cosine similarity.

Additional or different APIs could be used in the movie corpus pipeline to get more and better describing movie keywords if redundancy is taken into account.

6.2 General analysis

The most relevant way to measure if social matching improves traditional collaborative filtering algorithms is to compare the NDCG and MAP metrics from baseline experiments with the metrics from social matching experiments. As described in 2.4 and in 3.4, NDCG and MAP evaluate the recommender system based on the ranking of the recommendations. The higher up a correct recommendation is in the top-k the higher the NDCG and MAP metric value will be. The system showed clear improvements when comparing baseline metrics with social matching metrics for both algorithms. The improvement shown by the increased metrics was even clearer when comparing a randomized top-k produced by the algorithms with social matching. Improved NDCG and MAP metrics show that recommendations were reranked with correct and therefore more relevant recommendations higher up in the top-k. The improvement of metrics over random baseline isolate that social matching can improve the top-k independent of the top-k order given by a recommendation algorithm. That is why and how results from the experiments performed answer our second research question:

How does social matching effect metrics NDCG and MAP?

The highest increase in both NDCG and MAP was observed with the KNN algorithm, top-k of 20, Precision filtering threshold of at least 0.5 and LSI filtering threshold of at least 2.5. Through the evaluation of the results related to the design science criteria presented in 3.3.1, it is possible to answer our first research question

How can a recommendation system that uses personal information about a user from Twitter, affect top-k recommendations of traditional collaborative filtering?

Regarding the functionality criteria, the prototype uses social media in the form of incorporating the user's Twitter timeline to affect the top-k recommendations produced, which fulfills the functionality criteria.

The second criteria used to evaluate the prototype is Consistency. The results from Optimal user experiments show that the number of tweets significantly affects the average LSI score. It also shows that the average LSI score for users tweeting non-Latin languages is too low to have any significance for social matching. User 3 who tweets in both English and Spanish had a lower average LSI than English language users, but not so low that it was insignificant for social matching. A reason for that can be that many entities like names and objects are the same in both English and Spanish which enables the LSI similarity to still work, unlike non-Latin character languages like Arabic or Chinese where LSI cannot find any similarities because of the different characters used.

Experiments without precision filtering show that recommendations generated by the collaborative filtering algorithms have a low precision for many users in the dataset. Possible reasons for this are data sparsity and a high number of different movies in the dataset.

The precision filtering was necessary to filter out users who did not have enough relevant items in their test data. Social matching could not improve the recommendations of users with too low precision since the correct items were not present in the recommendations and therefore could not be reranked. A precision score of 0.05 ensures that recommendations for the user generated by the collaborative filtering algorithms at least have one correct item that can be reranked to improve MAP and NDCG. Results without precision filtering show little to no increase in MAP and NDCG but results after precision filtering show a clear increase for social matching compared to algorithm baseline. This shows that Improving recommendations generated by collaborative filtering is only possible when recommendations generated by the algorithm are over a certain precision at k threshold, in this case, 0.05.

Results shown in general statistics show factors that affected consistency and performance negatively. The fact that very little to no data could be gathered for 424 of 1188 users and 245 of 2622 movies impacted the social matching negatively. Another important factor that negatively affected the results is that only 40 percent of user tweets were identified as English and the cut-off date used to prevent using time irrelevant tweets, further lowered this amount of relevant English tweets significantly.

Since the experiments are performed in a real scenario where user tweets and movie keywords are fetched in real-time before the social matching module re-ranks the recommendations, performance is affected by many factors. If all Twitter and movie data is already stored in file or database, the system performance is fast but if the system needs to fetch tweets and keywords and pre-process them the system performs much slower. The biggest bottleneck for the measured performance in online performance experiments is the tweets per time limitation that the Twitter API has for free developer accounts. Only 1500 tweets can be fetched per 15-minute window. This means that the system is waiting for the tweet limit timeout to run out most of the time measured in the online performance test.

As previously stated, the social matching is affected by multiple bottlenecks regarding the used dataset. Several potential solutions to these problems have been identified which are deemed likely to mitigate some of these issues. Firstly, the creation of a more substantial dataset with a more refined pre-processing could lead to better performance. Instead of the 100 000 ratings dataset, the complete dataset of MovieTweatings could be used and put through the same pre-processing of removing cold-start users and movies with less than 30 ratings. As the results show, the best results were obtained when the precision at k threshold is applied. Yet with the 100 000 ratings dataset, increasing the threshold for precision filtering discards a substantial part of the users. Potentially the complete MovieTweatings dataset with precision filtering would leave a much greater number of users for evaluation than the 500 remaining when using KNN. It would be of interest to see what improvements a dataset of this nature would have on the result.

Yet the pre-processing performed still leaves users tweeting in languages other than English in the dataset. This could be handled in different ways. A more refined pre-processing could include the handling of such users by fetching their timeline and perform a language detection. Users not having most tweets in English could then be discarded. Since LSI similarity does not work if the tweet and movie corpora are in different languages or even alphabets, discarding these users is one approach that would remove the need for LSI thresholding. A different approach would be to implement multi-language support. This could be achieved in several ways. An easy to implement approach would be to identify the tweet-language and perform a translation of the keywords for each movie before performing social matching. This solution would be of interest if user retention is prioritized. Another solution would be to fetch keywords from different sources that return movie specific keywords in the desired language.

Lastly, as shown in general statistics of the system, the number of keywords for movies varies. Out of the 2867 movies in the dataset, 245 got less than 10 keywords. Thus, an LSI falling below the threshold could be the result of not tweet language, but a lack of keywords for the movies contained in the user top-k. Here as well could a more refined pre-processing be applied. Prior to the removal of movies with less than 30 ratings, movies with less than 10 keywords could be removed from the dataset. This would additionally decrease the number of users falling below the LSI threshold retaining their original top-k.

7. Discussion

In the literature review section studies using Twitter in collaborative filtering, the environment was presented. Although the addressed problem in our work is deemed as a novel, the taken approach in each of these studies and their result were of interest. Alahmadi & Zeng (2015) showed that the intricate user relationships on Twitter can be harnessed to alleviate cold-start in a collaborative filtering environment. Although this approach was very different from ours, it was included to demonstrate approach variation for using Twitter in a collaborative filtering environment.

The work by Nair et al (2016) demonstrated feasibility in basing a Collaborative Filtering environment on the MovieTweets dataset. The presented result showed that they could predict the correct genre for users with an accuracy of 72 % for the larger dataset.

Additionally, although this work similarly to Alahmadi & Zeng (2015) addresses cold-start, genre prediction was achieved by a type of social matching using cosine similarity. Our result demonstrates that social matching can show positive results not only for genre prediction but for individual movies. It also further demonstrates that MovieTweets is applicable to the modeling of a Collaborative Filtering based recommender system.

The work by Bansal et al (2016) took a similar approach using the genre to produce recommendations. With a user's favorite movie as the input, they retrieved general tweets for the movie such that user timelines were not considered. Yet the study performed the matching between the input movie with genre-specific movie data extracted from IMDb using LSI and Euclidean distance. The results show a prediction accuracy of 70%. Our results are in line with these results in the sense that LSI can be successfully used for matching pre-processed tweets with keywords for movies. Our work further shows that social matching can be used to augment collaborative filtering in a top-k movie re-ranking environment with positive results. Although our work is centered on re-ranking in the movie domain, others have been identified where the proposed social matching could likely be adopted. This is further expanded on in the Future work section.

8. Conclusion

In this thesis, we design and develop an artifact through the design science process model. This is done to examine how incorporating the use of social media into a recommender system, that uses traditional collaborative filtering, can affect top-k recommendations produced by the system. The result of the design and implementation phase is a system that, through a personalized approach, re-ranks the top-k through the social matching of users tweets and movie keywords in a novel way. Evaluation of this prototype recommender system is done with regards to design science evaluation criteria and mathematical recommender system evaluation metrics NDCG and MAP. The results show a clear improvement when comparing NDCG and MAP metrics of baseline algorithms WRMF and KNN with metrics from social match re-ranked top-k.

Even greater improvement is found when comparing a randomized top-k produced by WRMF and KNN with re-ranked randomized top-k through social matching.

Analysis of tweet and movie data shows that refined pre-processing or addition of multi-language support could increase these improvements even further since many users in the dataset could not be properly social matched with the movie because of non-English tweets or inaccessible tweets and movie keywords.

9. Future Work

To further study potential effects and improvements of incorporating social media into movie recommenders, refined pre-processing and a bigger dataset as described in the analysis of the results of the prototype system could be used.

Other ways to further improve the artifact could be achieved by altering the data pipeline for the movie keyword corpus. Keywords describing the movie could be gathered from more sources and given a weight depending on keyword quality. This weight could be taken into consideration in the social matching process.

The focus of this thesis was to incorporate social media into movie recommendation systems, but a similar approach could be used within e-commerce. Social matching in the form of LSI similarity could potentially be used between a user's tweet corpus and an item corpus containing keywords describing properties of e-commerce items and products.

References

- Acosta, J., Lamaute, N., Luo, M., Finkelstein, E., & Cotoranu, A. (2017). Sentiment Analysis of Twitter Messages. *Proceedings of Student-Faculty Research Day*.
- Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Berlin: Springer.
- Alahmadi, D. H., & Zeng, X.-J. (2015). Twitter-Based Recommender System to Address Cold-Start: a Genetic Algorithm Based Trust Modelling and Probabilistic Sentiment Analysis. *2015 IEEE 27th International Conference on Tools with Artificial Intelligence*, 1045 - 1052.
- Angiani, G., Ferrari, L., Fontanini, T., Fornacciari, P., & Eleonora. (2016). A Comparison between Preprocessing. *Kdweb 2016*. At Cagliari.
- Bansal, S., Gupta, C., & Arora, A. (2016). User Tweets based Genre Prediction and Movie Recommendation using LSI and SVD. *Ninth International Conference on Contemporary Computing (IC3)*, 1-6.
- Bergamaschi, S., Po, L., & Sorrentino, S. (2014). Comparing topic models for a movie recommendation system. *Proceedings of the 10th International Conference on Web Information Systems and Technologies. 2*, 172-183.
- Deerwester, S., Dumais, S. T., Furnas, G., & Landauer, T. K. (1990). Indexing By Latent Semantic Analysis. *Journal of the American Society for Information Science. 41*, 391-407.
- Dooms, S., Pessemier, T. D., & Martens, L. (2013). MovieTweatings: a Movie Rating Database.
- Gantner, Z., Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2011). MyMediaLite: A Free Recommender System Library. *Proceedings of the 5th ACM Conference on Recommender Systems*, 305-308.
- Gomez-Uribe, C. A., & Hunt, N. (December 2015). The Netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst. 6, 4, Article 13 (December 2015)*, 19 pages, 19.
- Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (January 2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems Volume 22 Issue 1*, 5-53.
- Hevner, A. R. (2004). Design Science in Information Systems Research. *MIS Quarterly Vol. 28*, 75-105.
- How Many Products Does Amazon Sell Worldwide – January 2018*. (2018, January 15). Retrieved from <https://www.scrapehero.com/https://www.scrapehero.com/how-many-products-amazon-sell-worldwide-january-2018/>
- Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. *2008 Eighth IEEE International Conference on Data Mining*, 263 - 272.
- Huang, A. (2008). Similarity measures for text document clustering. *In Proceedings of the sixth new Zealand computer science research student conference*, 49-56.
- Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods, and evaluation. *Egyptian Informatics Journal*, 261-274.

- Jagdale, R. S., Deshmukh, S., & S.G.Chodhary. (2016). Semantic Analysis of Tweets using LSA and SVD. *International Journal of Emerging Trends & Technology in Computer Science*, 5, 39-42.
- Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems Volume 20 Issue 4*, 422-446.
- Koren, Y., Bells, R., & Volinsky, C. (2009). Matrix Factorization techniques for recommender systems. *IEEE Computer Society*.
- MacKenzie, I., Meyer, C., & Noble, S. (2013). *How retailers can keep up with consumers*. United States: McKinsey & Company.
- Manning, C. D. (2008). Evaluation in information retrieval. *Introduction to Information Retrieval*, 151-175.
- MediaWiki API*. (n.d.). Retrieved from https://www.mediawiki.org/wiki/API:Main_page
- Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM Vol. 38, No. 11*, 39-41.
- Nair, P., Moh, M., & Moh, T.-S. (2016). Using Social Media Presence for Alleviating Cold Start Problems in Privacy Protection. *2016 International Conference on Collaboration Technologies and Systems*, 11-17.
- Oates, B. J. (2005). *Researching Information Systems and Computing*. SAGE Publications Ltd.
- Online movie Database API*. (n.d.). Retrieved from <http://www.omdbapi.com/>
- Peffer, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design Science Research Evaluation. *Design Science Research in Information Systems*, 398-410.
- Peffer, K., Tuunanen, T., & Rothenberger, M. A. (2012). A Design Science Research Methodology. *Journal of Management Information Systems, Vol. 24, No. 3*, 45-77.
- Ricci, F., Rokach, L., & Shapira, B. (2010). *Recommender Systems Handbook*. Springer Science+Business Media, LLC.
- Salton, G., Wong, A., & Yang, C. S. (1975, November). A vector space model for automatic indexing. *Communications of the ACM Volume 18 Issue 11*, pp. 613-620.
- Terrier stop list*. (n.d.). Retrieved from <http://terrier.org/>.
- Textblob python module*. (n.d.). Retrieved from <https://textblob.readthedocs.io/en/dev/>.
- The Movie database API*. (n.d.). Retrieved from <https://www.themoviedb.org/>
- Twitter, I. (2018). *Investor Fact Sheet*. San Francisco: Twitter, Inc.
- Zhang, Y., Jin, R., & Zhou, Z.-H. (December 2010). Understanding the bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics Volume 1*, 43-52.