University of Glasgow | School of Computing Science

Effective Product Recommendation using Social Signals

Ching-Han Cheng

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at The University of Glasgow

5th /September/ 2019

Abstract

This research proposes an idea to integrate social information into a recommender system to complement the data sparsity issue. A basic recommender model helps with information overload issue, and the extra social resource resolves the cold-start problem. Based on the implementation of a recommender model, it is to calculate the similarity between users' and items'. Therefore, the connection between a recommender model and social media is the item name, and a tweet mentioned the item. There are mainly five components of implementations in our product. First, the explicit and implicit baseline recommender model is built for user preference prediction. Second, in order to gather the on-time trendy of a specific product, a real-time tweets collecting function is approached by REST API. Third, in gathering a higher quality social resource, a search technique for real-time indexing and customize searching is provided by Elasticsearch. Fourth, a search query generator with entity parser by DBpedia is applied for extracting better search queries in summarizing the item names. At last, a sentiment analysis process is approached to quantify the tweets into positive, negative, or neutral. During the evaluation, we have found a sampled dataset with limit items leads to better performance on baseline recommender models. The character of each parameter and the best parameter setting is observed during model tuning. A more specific query about the item is gathered after entity parser. The query with all entities can extract more informative tweets since a relatively specific keywords list is provided. However, due to the limit time in this project, there are two ideas proposed for further implementations. First is to gather more in-deep social feature from the specific relevant tweet. Second is to build a hybrid model with explicit, implicit, and social feedbacks.

Education Use Consent

Contents

Chapter 1

Introduction

There are tens of millions of products available through online shopping every day; therefore, knowing the preference of customers and doing customize recommendation could be one of the core elements in providing better customer experience. Recommender systems are built and widely applied on electronic commerce fields such as Amazon.com, asos.com, and Netflix.com. In Amazon's online stores, people can see a list of products "they may like…" based on the preference they have informed Amazon by leaving explicit or implicit feedback on a specific product. The online platform would also show the items "other people have seen before…" based on the historical transactions of other customers who have a similar preference. The customize products recommendation helps users save the time form doing sophisticated searching by filtering a list of items they may be interested in by their historical transactions and interactions with the website. According to research from the McKinsey & Company in 2013 [16], it shows '35 percents of products sold in Amazon comes from recommendation' which raises an enormous extra revenue. Besides, the implementation of recommender system increases customer loyalty by understanding their needs. [27] For instance, with the recommender system, it could help new visitors to find the products they need much easier.

There are several types of recommender system; collaborative filtering approaches are well-known as gathering a group of users with similar preference, based on their reviews and ratings which show a positive or negative opinion about the product. The correlation between users could be used to generate product recommendation based on the preference in the specific group. In order to estimate the similarity between users, there are user-based, item-based, and model-based methodologies. User-based is defined as 'how do similar users like the item i", item-based is known as 'how does the user like an item similar to item i", model-based is to build 'a model of user-item interactions'. However, one of the problems with the recommender system is cold-start issues. When it comes to a new customer or a new product, the performance of the recommender system will decrease due to a lack of information about the specific customer or product. To mitigate the data sparsity issue in cold-start conditions is one of the aims of this project.

Social media is a platform gathering people's opinions on diversity issues from life, polities to society. Twitter is one of common social networking which ranked 11th most popular site in the world. There are around 500 million tweets posted on Twitter per day, according to Alexa's survey. People used to share their feelings and information on social media to maintain or build new connections with friends and publics. A recommender system seems more trustworthy while it provides reasons to convince users why he/ she would like the product or need the product. For example, the idea behind a product recommendation may base on a friend of the user bought the product. Furthermore, social media has a real-time nature, which makes it a widely use of event detection channel. In this perspective, tweets could be considered as a resource that provides extra information on upcoming or new products to solve the cold-start issue of recommender systems.

This project focus on building a recommender system with Amazon products review dataset and social media dataset is involved in solving the cold-start issue. To build the connection between datasets, a search engine- Elasticsearch is applied to filter valuable tweets relative to a specific product. The sentiment analysis is then be done on those tweets to decide whether to provide an extra recommendation depends on the user feels positive or negative about the product. Moreover, to extract useful features about the item may

Chapter 2

Analysis/Requirements

2.1 Literature review

Recommender systems have become a popular research topic since the first papers about collaborative filtering published in the mid-1990s [6, 22, 25]. Several ideas have been widely applied in various fields. E-Commerce is one of the leading areas. In the practical perspective of recommender systems, a well-implemented recommender model resolves the information overload issue by generating customize products recommendation and prioritize them into a list [10]. Moreover, the character of updating results in real-time helps users narrowing down the scope in searching the product they want [8]. In the case of looking for an item on amazon.com, a user may not be sure about which brand of the product they want, or how much for an average price. Once entering some keywords of the product, it will not only generate the results match the queries but also similar products the users may need for creating a comparison table with price, size, and reviews. While more information has been collected from a particular user, a more personal recommendation list can be made [23]. To the extent of presenting a better recommendation result, there are mainly two components to follow; with good coverage and serendipity. The former is to recommend a product the user like; the later is the user does not know the product before the system provides to him/ her [3].

Explicit models and implicit models are well-known as applying MF (matrix factorization) approaches which are the most popular and effective technique to characterize users and items by vectors of latent factors. The formal theory of building a model-based recommender model is to generate a set of users U={U1, U2, U3, ..., Ui, ...Um} and a set of items I={I1, I2, I3, ..., Ij, ..., In} then organize them into pairs (Ui, Ij). Each pair has a score showing the levels of preference, r(Ui, Ij) is the score of a specific user (Ui) interested in a particular product (Ij). Through training a recommender model, it could finally make predictions r on pairs of (User, Item) [2]. There are two typical types of feedbacks collected from customers, explicit and implicit feedback. The characteristic of the former feature is that we could easier know how people exactly feel about an item according to the star ratings or like versus unlike than the other. The later one faces difficulty in estimating the level of preference from users' reactions such as clicking on a product or time lasting on a page. In the early stage of the recommender system fields, researchers are more focus on explicit models [13, 21] due to considering explicit feedbacks is the most obvious way of presenting customers' preference on items. Nevertheless, an issue about lots of unobserved ratings in the matrix as missing values causes data sparsity [5]. In order to implement the algorithm more efficiency, several methodologies have been approached for dimensionality reduction on explicit models, such as SVD++ [14] and timeSVD [13]. Besides, not all contents are included in explicit feedbacks; most customers interact with items by implicit feedbacks such as views, clicks, and purchases. As a result, the hybrid feedbacks are normally approached in most cases concerning both types of variants [10].

Although the implicit feedbacks are much easier and cheaper to be collected than explicit feedback, it is difficult to identify how positive or negative the users feel about the specific item when they interact on it. For instance, while a user last in a product page for more than ten minutes in considering whether is due to the user interested in the item or he/she is away and forget to turn off the page. Moreover, one of the problems in implicit models is a lack of negative feedback, also known as the one-class problem [18]. A popular solution to deal with this obstacle is to model all missing values as negative feedbacks [7]. Nevertheless, those unobserved values are likely to be a mixture of real negative- the user is not interested in the product; missing value- the user might want to buy the product in the future. A solution for not taking all unknows as negative values are to do pairwise learning with BPR (Bayesian Personalised Ranking) approach [20]. In summarize, SVD and BPR methodologies are the solutions dealing with explicit and implicit models, respectively.

One of the limitations encounters in this research area is the cold-start issue, which is the difficulty in gathering enough information about a new visitor or an upcoming product [**?**]. An idea for compromising this situation is to apply multimedia sources [**?**, 11]; for instance, images could be an element for extracting more features of a specific item, which tend to profile the relationships between various items sold on the website. Moreover, considering more implicit feedbacks which are available from several detection methods; how long does the visitor spend on each page or how often. Each reaction on the system can also be ordered in sequence by time-stamps. There are mainly two directions of generating sequential transactions; one is by the frequency of a user buys the same product; the other is the order of deals. For instance, a customer may purchase a high utility product such as tissue once a month; or when a customer first purchases a coffee pot, he/ she may start buying coffee beans.

Besides the perspective of resolving the issue on data sparsity behinds the cold-start problem mentioned in the previous paragraph, some researches tend to focus on a more customize recommendation [4, 26]. To the extent of getting extra personal information from a user, social media is a bridge to make the connection. Social media provides a platform letting people express their opinions and build connections with the public. World of Mouth (WoM) is considered as a strategy in viral marketing [19, 15], which switches the module of company-to-customer to customer-to-customer [28, 15]. Twitter is one of a powerful platform applied for sharing opinions or leaving comments. Therefore, this research comes with an idea of integrating social media dataset with baseline recommender models. As more people willing to share their reviews after using a product or looking for reviews on social media, which also generates a social impact on potential customers. In the way of gathering social features as a part of the user feedbacks, sentiment analysis is available form the VADER resource [9] which helps with implementing emotion detection from the text, especially form social media. A positive review is likely to motivate the public to buy the product; in contrast, a negative review may cause the decision not to buy. To gather more information about customers' preference, the novel idea of this research is to add social media- Twitter as an external channel to boost the performance of original recommender models. Moreover, to complement the data sparsity of cold-start issue.

Nevertheless, a general challenge in providing social media sources is the high percentages of noisy tweets and a lot of retweets [17]. According to the statistical survey, more than 90% of tweets are noises, which are tweets without useful information. Another issue is about retweets since most retweets contain mostly similar or precisely the same contents as the original tweets. These induce farther calculations leading to waste on computing results. With regard to resolving this issue, a proper filtering process is required to remove those noises and retweets. Concerning extract the information we need, a search technique can be used to set the conditions that need to be matched. A search technique has typically two components; able to do indexing [1, 24] or crawling the data then extract the information we need through searching [12, 24]. The indexing process is applied for saving data into a particular format; for instance, tweets are encoded in JSON. The searching process then helps us to filter only the part of the data we need. In this project, Elasticsearch is approached for making a connection between Twitter and social media dataset. It is an open-source applied as a distributed search and analytics engine for various types of data, such as textual, numerical, structured, and unstructured data.

2.2 Requirements

This research aims to build a recommender system which can make the predictions on a user's preference in each specific item. In considering of building a more practical product, we focus on the well-known e-commerce as the basis of our recommender system. Moreover, to implement more personal recommendations, social media is an additional source taken into consideration. A customer review dataset is available from amazon's open-resource dataset. The electronic products reviews is chosen from various category to analysis due to not only fewer researches have been implemented in this topic before but also a more worth-consideration point of view; based on the assumption that people are more likely to share their opinions about electronics products on Twitter than other categories of products such as books, videos. The product reviews dataset is approached for building a baseline model; the social media dataset is an extra resource applies to collect customer's opinions on new products which deal with both data sparsity and also implementing a more customized recommendation.

In the opinions of boosting the performance on product recommendation, we would like to link the twitter dataset amazon dataset together for gathering more personal information. Nevertheless, both datasets are provided from different channels, so the integration is not able to implement by user-based. For solving this limitation of cross-platform is to do the combination with item-based; for instance, an item sold on Amazon could be mentioned on twitter. The idea of this project is to train a baseline recommender model first to identify the relationships between items and users. In following, the item names can be extracted to filter those tweets are tweeted for discussing any item sold on amazon.com. If a tweet is collected from the previous step, it presents a piece of extra information about the user's opinions in the specific product. To the extent of understanding the content of tweets, a sentiment analysis approach is available for identification. By this progress, some features are collected from both aspects of products and users. At the end of this project, we would like to evaluate if these features extracted able to boost the baseline models built at first. If those features are able to fulfill a better performance, what is the threshold of the characters; for instance, how many positive tweets about the product could help for boosting a user's preference on the specific product, or how many negative tweets could bring a negative impact on users' preference on the item.

From the aspect of integration between baseline recommender model and tweeter features is needed. A model-based recommender system is chosen due to the idea of adding an extra feature on the original model. With regards to utilize of amazon dataset, the similarity scores between users are available in calculation base on the concept of collaborative filtering. Both explicit and implicit feedbacks are provided in generating explicit and implicit models with considering star rating versus without the consideration. BPR model is also accessed as a proposed methodology by setting particular parameters. Apart from the baseline models, the process of gathering extra features from Twitter arises a difficulty in dealing with most tweets are noises and to process a massive amount of data. A solution approaches in this research is to build a search technique with Elasticsearch. Elasticsearch is an open-source applied as a distributed search and analytics engine for various types of data, such as textual, numerical, structured, and unstructured data. This research has implemented a basic search engine by Elasticsearch to extract useful tweets form social media by building a link between two datasets. Sentiment analysis is then approached in determining the tenderness each tweet, positive, negative, or neutral. The overall results are then provided as new features for the process in model integration.

In summarize, base on the researches have been implemented in the past; recommender systems aim to provide a high accuracy personal recommendation in various fields. Therefore, in the decision of datasets, not only amazon reviews dataset but also twitter dataset is selected for providing extra information to solve the cold-start issue. In the stage of building the recommender models, both explicit and implicit models are built as the baseline. Then some extra features can be extracted from tweets through mainly three steps, data collection by REST API, filtering useful tweets with the search Elasticsearch technique, sentiment analysis by VADER sentiment analysis methodology. At last, the integration between baseline models and features are implemented by various strategies to check the varies in performance.

Requirements List

For building the product of this project, there are requirements we aim to follow in each progress. A list prioritize each requirement by MoSCow method is shown in table 2.1 below:

Scenario

In real cases of running an e-commerce business, some general challenges, and the solutions from this project are discussed in the following.

First, when a customer visits amazon.com, which product should we recommend to him/her? In this project, we could first predict his/her preference toward each item by training

	Requirements	MoSCow
Req-A	This product must be able to read the reviews and generate an item recommendation list to a specific customer.	Must
Req-B	This product must be able to rank users' preference for variety items.	Must
Req-C	This product must be able to gather tweets through TWITTER REST API.	Must
Req-D	This product must be able to retrieve tweets from a pre-generated index.	Must
Req-E	This product must be able to do sentiment analysis on tweets.	Must
Req-F	This product must be able to extract entities.	Must
Req-G	This product should be able to do the integration between baseline models and social media features.	Should
Req-H	The project will develop a mechanism for generating effective product search queries for social media data.	Should
Req-I	This product should be able to generate scores that incorporate social and review based evidence.	Should
Req-J	This product could be able to produce recommendations for users with no past history.	Could

Table 2.1: MoSCow Requirements table

a baseline recommender model with amazon reviews dataset [Req-A]. The baseline model is able to predict a list of product recommendation ranking by the most favor to less favor is generated [Req-B]. Second, what people perceive on a particular product? If there are more discussions from social media on the product, should we publish more advertisements on the specific goods while people seem interested in the upcoming item? By resolving this issue, the twitter dataset is approached for gathering more information about new products [Req-C, I]. The new products' names are provided as a query in searching relative tweets [Req-D], then the sentiment analysis process is applied for checking users' opinions on the new product [Req-E]. If there are relatively more tweets are positive, we are likely to spend more money on marketing the products, otherwise, lower cost on the advertisement. Third, how can we know those tweets we filtered are relative to the items sold on amazon? This issue is hard to be resolved due to the natural limitation in social media resource; nevertheless, to optimize the result from the filtering process, the query provided in searching is critical. In the aim of generating a more specific query fits on the product, entity parser is applied to generate a list of entities extracted from each item [Req-F, H]. Fourth, how to boost the performance of our recommender models to provide a better product recommendation? The integration between baseline models and social media features are provided in trying to implement a better performance on models [Req-G, J].

Chapter 3

Design & Implementation

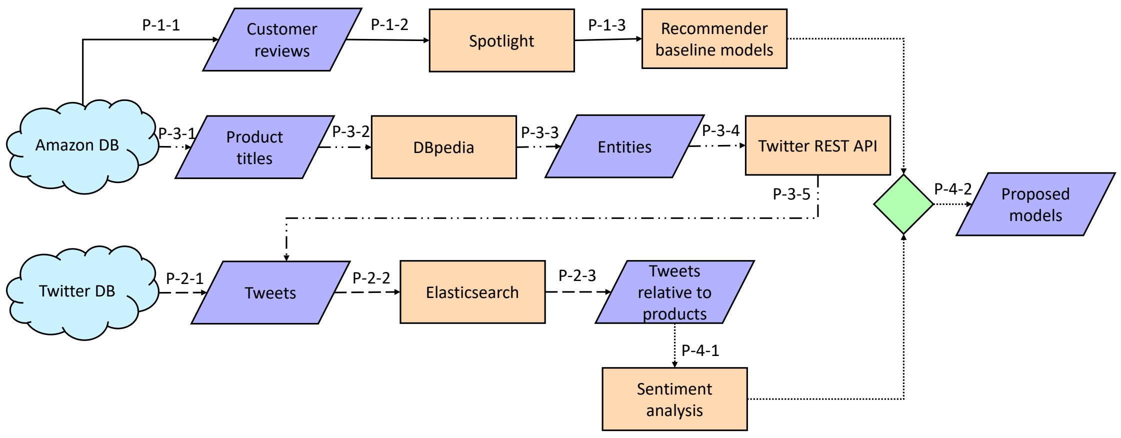


Figure 3.1: Product architecture

The figure 3.1 above shows the overall architecture of our product; there are mainly four components contained in this project. At first, the process of building baseline models by the spotlight, which implement the product recommendations to predict the preference of each user on items in Amazon dataset [Req-A, B]. Second, filtering tweets with more information relative to products sold on Amazon by Elasticsearch technique [Req-D]. Third, generating a proper query for extracting tweets through Twitter REST API by searching keywords [Req-C, F, H]. Fourth, A sentiment analysis is approached to collect features for model integration. [Req-E, G, I, J].

Before diving into each component in details, this chapter structured in four parts. At first, a dataset introduction is shown since a data sampling on amazon dataset has been approached in this research. The sampled dataset is then provided in the following process. The later three parts deploy the design and implementation of the four components mentioned above(Figure 3.1). The first component in building baseline models contained in part two. Part three describes the second and third component due to they are both involved in searching for Twitter data. Part four shows the last component of model integration.

3.1 Datasets

The two datasets apply in this project are Amazon customer review dataset- electronics category from 1995 until 2015 and Twitter dataset.

Amazon reviews dataset

The original size of amazon reviews dataset is 3,091,024 rows x 15 columns, with 185,771 unique items x 2,152,758 unique customers. The definition of variables which apply in this research is shown below.

Variables	Definition
customer_id	random identifier that can be used to aggregate reviews written by a single author.
product_id	the unique Product ID the review pertains to. In the multilingual dataset the reviews.
product_title	product name.
star_rating	the 1 to 5 star rating of the review.
review_date	the date of the review was written.

Table 3.1: Amazon dataset variables list

We first train models with the original dataset; however, there come problems on large time-consuming on training (at least four hours on each iteration) and the difficulty in getting the performance of models due to the runtime error on evaluation process (without enough memory for calculating). To the extent of improving the efficiency on building models, a smaller dataset for training is sampled from the original one. There are various strategies approached in resampling; the basic idea is to sample by interactions. This progress solves the issues of long-time training; still, the performance on models are too low to make the prediction effectively.

From the observation on the original dataset, the average number of reviews in each product is 17; the average star rating is 4.0. An idea for reducing the average mispredicted score of the explicit model is to minimus the bias on star ratings in the dataset. An implicit model aims to prioritize the higher preference items in the head of the product recommendation list; however, not every item are popular among the public. Therefore, while fewer numbers of items are considered to lead to a shorter ranking list generated, and those less important products are discarded. From the aspect of boosting the performance on both models; we choose items with more reviews which assumed to get higher preference among users, and then select an even number of reviews in each level of star rating. The following list shows the basic ideas in approaching the sampling process:

1. Remove those items with too less reviews from customers.
2. The average of star ratings on products should be 3 (scale 1 to 5).
3. Generate a sample with equal numbers of reviews in each group by star rating (group 1 to 5).

As a result, there are around 72,470 users x 100 items in the sample. A much shorter time-consuming is taken, around 15 seconds for an iteration, which is about 960 times faster. The performance on models also has a 30 percents improvement on the accuracy.

Twitter dataset

There are two different tweets resources provided in this research; one is a historical record gathered in 2014; the other is collected in real-time with the REST API searching by keywords. With regard to collect more relevant tweets, the keywords are collected by various approaches; further discussion is provided in later sections.

3.2 Baseline Recommendation

The first component of our product is to build baseline recommender models. The progress of implementation is shown in the flow chart below:

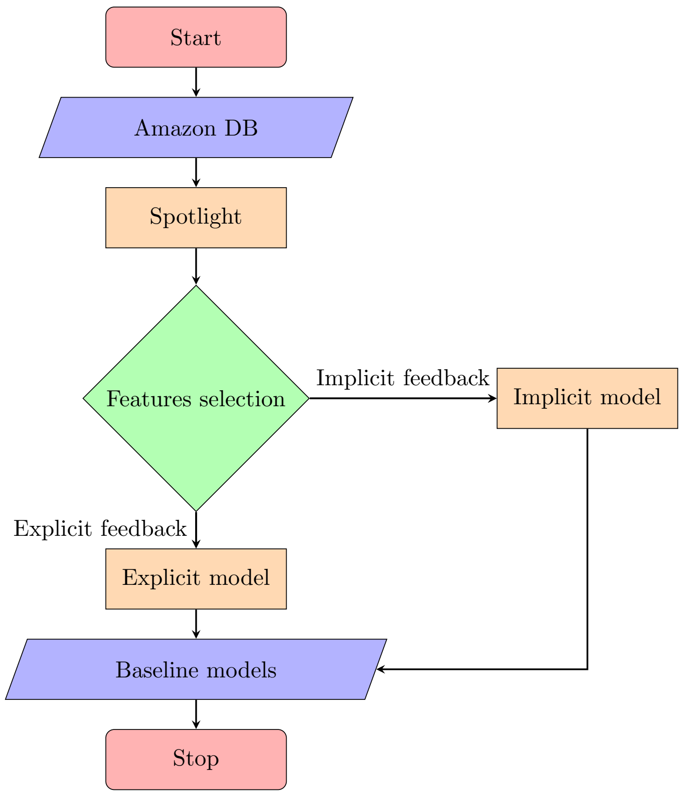


Figure 3.2: Product architecture -1

Design

Base on the concept of the collaborative filtering methodology; to calculate the similarity between users' preference from their reactions, we need transactions between users and items.

Amazon product reviews dataset provides information about users' reviews and ratings on products they have bought from amazon.com. This dataset is provided in building our baseline recommender models with customer_id, product_id, product_title, star_rating, and review_date as variables. In order to build a model-based recommender system, a resource called spotlight is chosen for building both explicit and implicit baseline models. Spotlight is a well-structured Python framework to implement both deep and shallow recommender models by PyTorch. The package contains both factorization and sequential models; the former models use the idea behind SVD decomposing the utility matrix into two latent representation of user and item matrices, the latter includes the time-series concept.

Moving to the feature selection stage, which depends on the types of models we are going to build. Spotlight provides explicit, implicit, and sequential models' sources. The different feature selected between explicit and implicit models is; only the former models regard star_ratings as a variable in modeling. Besides, to distinct sequential models from others; it requires review_date as timestamps to generate sequential interactions for training.

Therefore, as this project planning to build both explicit and BPR implicit baseline model, the explicit feedbacks and implicit feedbacks are considered respectively. The reasons for choosing them as baselines are; the explicit models based on the most obvious theory in considering star ratings as a user's preference. Nevertheless, explicit feedback is normally harder to collect than implicit feedback. Consequently, an implicit model is involved and according to resolving the challenges of one-class problem [18] mentioned in the previous chapter, the pairwise learning method on BPR models are build by with implicit models resource with specific parameters setting.

To evaluate the performance on models, the RMSE (Root Mean Square Error) is the baseline to calculate the wrong predictions of star rating in an explicit model. On the other hand, MRR (Mean Reciprocal Rank) is the baseline for all implicit, BPR and sequential models which measures how quickly a user can find a product he/ she feels interested in from the list of products recommended. The comparison between models is shown in table 3.2 below:

Implementation

To implement the baseline models, we first apply some data preparation to index the user and item IDs into the numerical type and transform ratings and time stamps into floats and integers in respectively. In the beginning, there show some nulls in data, so any rows with null are removed. Besides, the variable 'review_date' needs a python DateTime transformation by pandas source into proper timestamps. After that, some fundamental statistic summarizes on the dataset is provided, such as the average numbers of reviews on each item, average star ratings, how many unique users and items.

Due to the sampling process mentioned above overcomes the long time-consuming on training and low performance on models. A randomly sampling by interactions with spotlight.sampling is provided at first; however, this only speeds up the training process, the performance is still quite low. Therefore, in the case of reducing the numbers of items but keeping more valuable ones is to filter those items with enough numbers of reviews to limit the bias in our dataset. A minimum average number of reviews on items is set as 1,500 due

Types	Factorization model - explicit	Factorization model - implicit	Factorization model - implicit (loss = BPR)	Sequential model
Variables	user_id, item_id, star_rating	user_id, item_id	user_id, item_id	user_id, item_id, review_date, star_rating, num_users, num_items
Evaluation	RMSE(Root Mean Square Error)	MRR(Mean Reciporocal Rank), Precision, Recall		
Default parameters	loss='regression', embedding_dim=32, n_iter=10, batch_size=256, l2=0.0, learning_rate=1e-2	loss='pointwise', embedding_dim=32, n_iter=10, batch_size=256, l2=0.0, learning_rate=1e-2, num_negative _samples=5	loss='BPR', embedding_dim=32, n_iter=200, batch_size=128, l2=1e-9, learning_rate=1e-3	loss='pointwise', representation='pooling', embedding_dim=32, n_iter=10, batch_size=256, l2=0.0, learning_rate=1e-2, num_negative _samples=5

Table 3.2: Recommender models setting

to the idea of keeping around only 100 items in the dataset is proposed. There are around 165 items left after setting 1500 as the threshold. A randomly choosing 100 items then generate a new dataset is approached in the beginning. However, once a random dataset is generated, we have to retrain the model; otherwise, a different baseline model is approached. The retrain limitation mentioned could easily lead to restart progress if there is not an eternal principle provided. In preventing this issue, the random setting is discarded, a dataset match requirements in design is stored and applied for building baseline models.

Before the training process, variables are selected for different models, as shown in table 3.2. Those variables after data preparation are assigned into interactions for cross-validation process trying to mitigate the over-fitting issue. The dataset is then split into training and testing data in 80% and 20% separately by interactions.

In the training stage, the different parameters setting vary the performance on models. Due to the limitation of long time-consuming on training in the beginning, only a small iteration below ten and the lowest embedding dimension 32 are given. The default parameters are set in the beginning before approaching the sampling process, which took around 40 hours on training with ten iterations and 32 embedding dimension. Despite the long-time training, the performance on the explicit model is still too low, and the RMSE is around 2.1. That is, while a rating on a specific item is predicted as three, the actual rating might be 1 or 5, which generates a meaningless prediction. Besides, the implicit models are not able to be evaluated due to a lack of RAM in the calculation. After the re-sampling process, the MRR score is successfully calculated by the evaluation module. During the experiment on tuning the models, it shows significant improvements in both explicit and implicit models; the RMSE score turns from 2.1 to around 1.36, the MRR score is from about 0.01 to over 0.09.

After all, each model is fitted and trained by a specific module. An explicit model is able to predict the star ratings of each item from users. There are both positive rating from 1 to 5 and a negative rating from -1 to -5 are generated as the results. On the other hand, an implicit model can order the preference of items for each user. The comparison between each pair of items are approached so the result will show in a pair with (1,-1); for example, there is a pair of item (I1, I2) for comparison if a customer is more interested in I2 than I1, then the result will be (-1,1). In summarize, a list of product recommendation for a specific user is available from both models by raking the user's highest favor item to the less favorable one. [Req-A, B]

3.3 Searching Twitter Data

This section discusses the progress of collecting tweets relative to items sold on amazon.com, which is crucial in providing extra resources from the social aspect. There are mainly two stages in the process; firstly, building a search technique by Elasticsearch for both indexing and searching to remove noises from tweets. Secondly, generating query by DBpedia entity parser and then retrieving tweets through searching keywords by Twitter REST API.

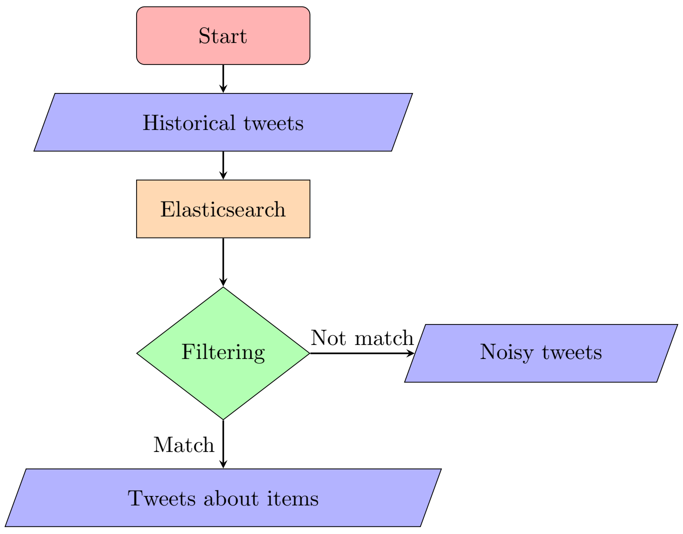


Figure 3.3: Product architecture -2

Design

The initial idea is to gather relevant tweets from a historical twitter dataset has collected in 2014. This decision based on the period between Amazon dataset(from 1995 until 2015) and Twitter dataset(2014) are matched. The historical tweets are firstly processed to the Elasticsearch stage; Elasticsearch is an open-source provides a scalable search solution and a unique type-level identifiers indexing technique. An indexing process is approached in

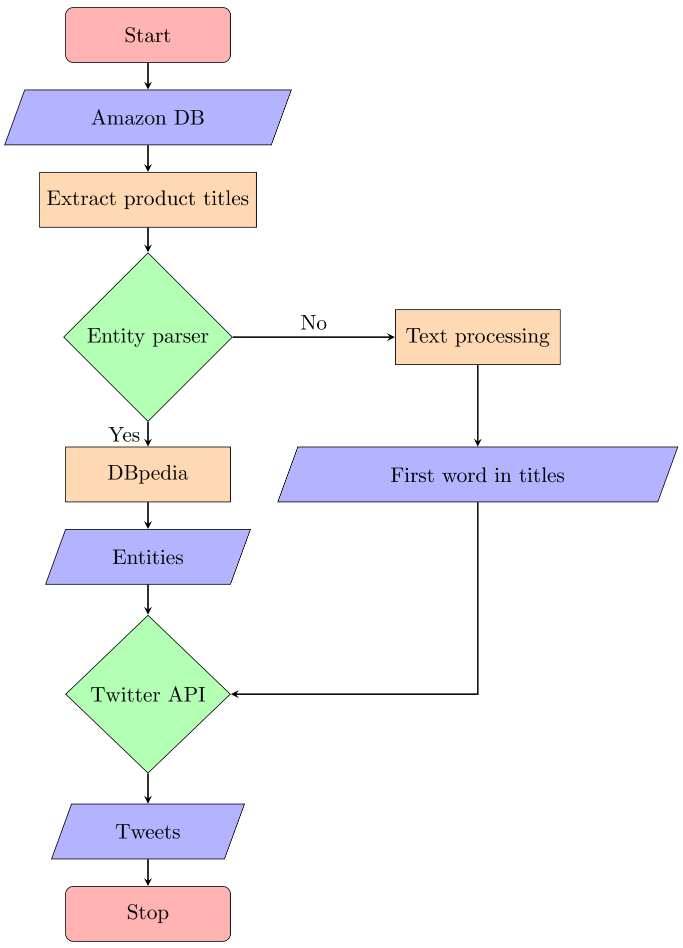


Figure 3.4: Product architecture -3

available searching; a customize module for tweets indexing is chosen in this step. After indexing, the searching process is proposed to remove irrelative tweets. In this research, there are four rules set for filtering tweets we need, which is shown in the following.

1. The tweet must at least match 3 terms in product titles.
2. Only filter Tweets In English.
3. The tweet must not match ‘http ://’ & https://’.
4. The tweet must not match ’RT @’

The first condition is set in the aim of looking for a higher similarity between product titles and the content in tweets. The assumption decides the threshold three on each tweet is structured with mainly three components; brand name, product name, and item model, so there must be at least three terms matched in titles. Still, multiple situations are set as the threshold, such as at least 50% match or if there are less than five terms then must match two terms; otherwise, must match 50% of terms. At the end of these experiments, the threshold at three gets a comparably reasonable result which seems to have more relative tweets in result, which leads an around 90% of tweets removal. The second rule is obvious since the product titles are in English, in order to get a less bias result is to filter only tweets in English. The third and fourth conditions proposed in solving the problem of noises in advertisements and retweets are gathered based on the previous two rules. In the direction of collecting customers' opinions in a particular item, advertisements are considered as noises since enterprises or sellers typically post those tweets. From the observation on those advertisements, most of them contain an URL link while trying to sell the product. On the other hand, a retweet considered as a duplicate as it has the same content as the original tweet. As a result, the searching process filters the tweets relative to the product and removes most noises in advertisements and the retweets by Elasticsearch. The process architecture shows in figure 3.3.

However, based on an overall requirement of generating customers' preference for products in real-time, an on-time gathering, searching, and filtering in tweets should be available. A nearly-real-time tweets collection is approached with the Twitter REST API. REST API enables developers to access information and resources using a simple HTTP invocation provided by Twitter. To the extent of collecting more relevant tweets, the query we generated for searching is crucial to describe a product in specific. In this case, the most characteristic features of items are product titles. While people are sharing their opinions about specific goods, most of them will inform the public which item is it by providing a short description of the particular product. Therefore, the product titles are first provided as a feature for gathering tweets. In order to filter only keywords which represent the products, some text processing progress is first required. There are mainly three steps provided in the text processing; which are normalization, tokenization, and removal. The details are provided below:

1. Normalisation: transform words in lower cases.
2. Tokenisation: split the sentence in terms by a space.
3. Remove stopwords, punctuation and duplicates.

Apart from how many terms in common between titles and tweets, does the tweet show a customer's opinion on discussing the specific item sold on amazon.com. From the observation, most tweets filtered by the terms which are not relative to titles; for instance, ['composite', 'to', 'hdmi', 'converter'] has the terms 'to' and 'composite' which are likely to gather tweets not about the product, on the other hand, the terms 'HDMI' and 'converter' should be combined into 'HDMI converter'. In resolving this issue, an entity parser is involved by DBpedia to extract only keywords about the particular item. DBpedia Spotlight is a tool for automatically annotating mentions of DBpedia resources in text, linking the unstructured information sources to the Linked Open Data cloud through Wikipedia resource. The detection is different from the common NER(Named Entity Recognition) parser due to the dataset provided is not in a standard format with organization, location, and person. In contrast, the Linked Open Data cloud contains the concept closer to the Web of Documents. As a result, there is a significant improvement on extracting meaningful terms from product titles with DBpedia source.

By the DBpedia process, there are entities extracted from almost each item title; some titles without entities are considered as noises to be discarded. The removal also leads to a change in the data processing stage; those items without entities are removed. On the other hand, due to the numbers of tweets gathered from REST API per iteration is limited, there are three strategies provided in searching. We first search by the first word in titles after text processing, and then each entity and all entities. The difference between the two DBpedia query setting is the former one(each entity) aims of gathering more resources from Twitter, and the later one(all entities) provides more information in gathering more specific tweets. For instance, while a title has an entity list ['sony', 'cabinet', 'clock radio'], in searching by each entity case, the REST API will search tweets mentioned in each 'sony', 'cabinet' and 'clock radio'. Each searching has a maximum 1,000 tweets as feedback the maximum tweets we could gather is 3,000. Still, most tweets gathered from single entities are meaningless due to the REST API can only call back part of data. The more information provided in searching in all entities searching can lead to receiving the more valuable part of the dataset. As a result, there are three different types of the query provided in extracting tweets by REST API, the process flow presented in figure 3.4.

After approaching the DBpedia process, there is only a rule changed in the Elasticsearch engine. The change is based on DBpedia technique only filters entities essential in describing the item, so there is a significant cut back on the length of the query from an original product title. Therefore, the threshold of at least match-three terms is changed to at least match 50% of entities. The final results are assumed as including tweets about items; most noises are discarded through the searching process.

Implementation

To implement tweets collection based on product titles, some text processing on titles are first approached. With regard to the three text processing steps mentioned above; normalization, tokenization, and removal, the nltk source is provided. In the tokenization stage, there are ways to trim the sentence; such as by punctuation, space for customizing setting. Split by space is selected since there are terms represented model series such as 'PC-1000' which is not allowed to be separated into 'PC' and '1000'. Moreover, there is no stemming process since we still need to use those specific terms for searching and doing filtering. For instance, if a word 'headphone' is stemmed to 'headphon', it may cause difficulty in searching a tweet mentioned the specific product. After the text processing, a duplicate removal and indexes resetting are displayed.

Another way to implement title processing is with an entity parser by DBpedia which provides a source link 'http://model.dbpedia-spotlight.org/en/annotate' for entity implementation. There are confidence and support values available in setting with the spotlight.annotate function. The results are shown in the format [URI, offset, percentageOfSecondRank, similarityScore, support, surfaceForm]. In this research, the confidence rate and support is set as 0.4 and 20, and 'surface form' and 'similarity score' are stored, which are the entity names with their similarity scores.

In order to gather tweets through the Twitter REST API, the twitter access key and token are required in the beginning. With the access codes, searching is available by tweepy source implemented in python. There is a tweepy.Cursor, which is a function to implement iterate searching by the API allocated. Search API is decided so the queries with keywords can be given for tweets collection requiring a match on queries. There are various keywords set in the function, the first term in titles after text processing and each entity or all entities generated by DBpedia. These results are then converted into JSON type to get indexing by Elasticsearch in particular document type - tweet.

A more complicated implementation is to implement the combination between REST API searching and Elasticsearch since there are multiple conditions considered. Tweets collected from REST API is first indexing into Elasticsearch, and then the rule-based filtering process is set. There are mainly 'must', 'must not' and 'filter' in the first setting layer, and then the condition 'match_bool_prefix' is provided for more scalable decisions on matching in terms. A minimum_should_match is set as the threshold on boosting the relevance in tweets and items. The main difficulties face on this progress is the ordering among processes (searching, indexing, filtering), should we implement by each tweet or each group or the entire dataset. At last, we implement the progress by groups, that is, a product title as a group of searching and then indexing all the tweets into Elasticsearch before filtering by the specific product terms.

In conclusion, we implement the searching, indexing, and filtering on tweets by items [Req- C, D]. Moreover, the DBpedia entity parser is approached in gathering more informative tweets [Req- F, H]. These tweets are processed to the next stage for further features exploration as extra information.

3.4 Proposed models integration

In the final stage, we aim for the features collection in social media to boost the performance of our baseline models. With regards to figuring out the opinions in tweets from users, sentiment analysis is the idea to detect how positive or negative those tweets are. By the end of the experiment, there is a various way to assess how social source brings an impact on customer's preference on a specific product? Will a positive or negative perspective from tweets draw a boosting or decreasing on the baseline recommender models?

Design

After receiving tweets from the previous stage, a sentiment analysis process is approached with VADER Sentiment Analysis. VADER (Valence Aware Dictionary and sentiment Reasoner) 'is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media, and works well on texts from other domains'. There are mainly three perspectives in sentiments, positive, neutral, and negative. While there is more than one tweet about a particular item, we could calculate the average score of tweets or summarize how many positive tweets, negative tweets, and neutral tweets are there in the group of a specific item.

This stage aims to boost the performance on baseline models and observe any changes that happen after the model integration. There are multiple trails approached in this stage, the fundamental idea is to filter only the items with a social impact, which are considered as having relevant tweets to the specific item and the average sentiment score is not zero. The average compound score of tweets mentioned a specific product on behalf of an extra perspective from social media. Therefore if it is positive, we assume there brings to be

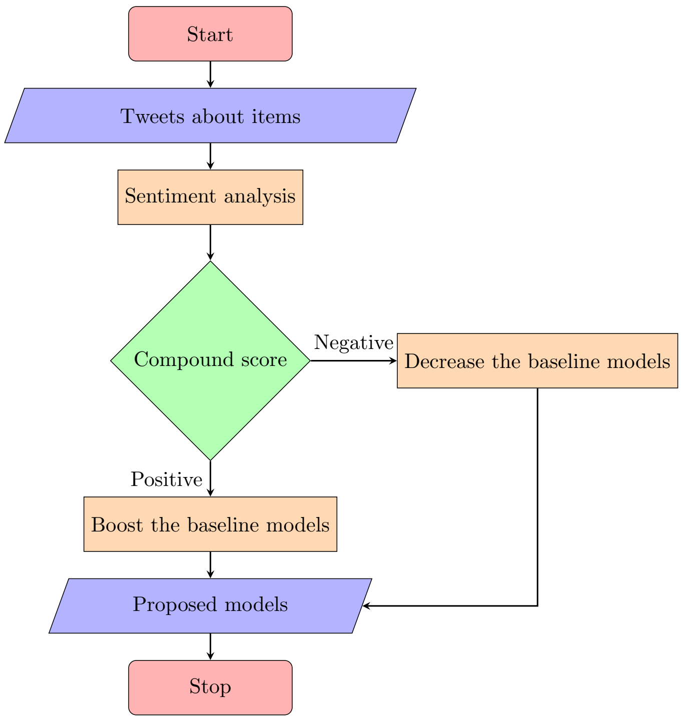


Figure 3.5: Product architecture -4

a positive impact on the particular item; an increment on a predicted score is added. In contrast, if it is negative, there could be an opposing influence on the item; a cutback in the prediction is fed.

Our proposed model implements an integration between social impacts and the baseline. The scoring formulas are shown below. That is, a compound score is calculated by the positive sentiment score minus the negative sentiment score. If the compound score lager than zero, we consider it as a positive impact; otherwise, a negative impact. The threshold 0.2 is set for boosting or decline the model's item weighting, and the threshold is decided based on the average item weighting is about 0.2.

$$
\begin{aligned}
compoundscore = positive_sentimentscore - negative_sentimentscore \\
itemweighting + 0.2, if compoundscore > 0 \\
itemweighting - 0.2, if compoundscore < 0
\end{aligned}
$$

Implementation

In tackling the sentiment analysis progress on tweets, some text processing on tweets is provided in the beginning. With some basic tokenization and normalization in texts, the vaderSentiment source is then approached. After a sentence passing through the analysis, a score list with [positive, neutral, negative, compound] is generated automatically.

While approaching the integration between models and social media features, we head back to check the formula on calculating the prediction score of the baseline models. The algorithm in an explicit model is to generate a user embedding matrix and an item embedding matrix in separately first. Then apply the inner product between these two matrices to perform a new matrix. These new matrices are summarized before adding some user biases and item biases. However, there is a difficulty that comes into the integration between baseline models and social impact progress by the uncertainty in weighting adjustment on models. A reason for the uncertainty is the diversity in emotion scores and the number of tweets by each item.

The following formula presents the way of predicting rating by an explicit model, that is, how a user [i] will score the item [j].

$$\begin{aligned}(user_embeddings_weight[i] * item_embeddings_weight[j]).sum(0) \\ +user_biases(torch.tensor([i])) \\ +item_biases(torch.tensor([j]))\end{aligned}$$

Chapter 4

Evaluation & Results

This chapter provides information about the evaluation process set up based on the research questions shown in the following table 4.1. The previous two research questions aim to attain a reasonable baseline effectively. The following third to fifth is about retrieving social information. The last is about integrating baseline and social information.

	Research questions
RQ-1	Do the recommender models become more effective after using a sampled dataset?
RQ-2	Can we get a better prediction on baseline models by changing parameters rather than using the default parameter setting?
RQ-3	Does Elasticsearch help with removing noises in tweets?
RQ-4	Does entity parser by DBpedia help with generating a more specific search query about an item in searching?
RQ-5	Which query gathers a more relevant result in tweets; search by the first word in titles or all entities or each entity?
RQ-6	Do social features boost or undermine the performance on baseline models after integration?

Table 4.1: Research question list

RQ-1: Do the recommender models become more effective after using a sampled dataset?

Due to the input of a recommender system provides the information on generating matrices between users and items, the decision made on choosing a dataset is crucial in generating better results. Therefore in this research, we have approached three different datasets; an original one and two sampled dataset by distinct strategies. In comparing a more effective model mainly depends on two components; the time-consuming on model processing and the performance on making a prediction. By evaluating the performance on models; a lower RMSE score is better for an explicit model, and a higher MRR score is better for an implicit model. The principle of the RMSE score is to calculate errors between predicted ratings and the actual ratings from users; the MRR score based on estimating how accurate the orders between predicted user preference and actual user preference on items.

By running the evaluation process, we provided a consistent dataset set-up and the default parameters in each explicit and implicit model. In the process of resampling, we had first sampled an around ten times smaller dataset by interactions; from an original dataset with 3091,024 interactions to a sample dataset with 300,000 interactions. Another sampling based on the idea of retrieving fewer items, we have implemented an extra data processing (shown in chapter 3.1) which is to choose only items with more reviews, and their product titles must have at least one entity detected from the entity parser. At a result, the numbers of items drop from 185,771 to only around 114 items left in the sampled dataset.

In an explicit model, if the sampled dataset achieves a shorter time-consuming on training and a lower RMSE score, then the sampling process is effective. On the other hand, besides a shorter time consuming, a higher MRR score is acquired in an implicit model. Table 5.1 and 5.2 show the result of the explicit model and implicit model in separately. Both tables present the changes in the time-consuming and the performance on models with approaching three different datasets. Each row represents the result in different sizes of a dataset from the original one, sampled by interactions, and limit the numbers of items.

Dataset size	Time consuming on training/ per iteration	RMSE score
2152758 users x 185771 items x 3091024 interactions	˜ 4 hours	Train RMSE = 2.1, Test RMSE = 1.91
279,789 users x 61,286 items x 300,000 interactions	˜ 10 second	Train RMSE = 0.198, Test RMSE = 1.350
73,526 users x 115 items x 75,000 interactions	˜ 5 seconds	Train RMSE = 0.866, test RMSE = 1.353

Table 4.2: Explicit model

Dataset size	Time consuming on training/ per iteration	MRR score
2152758 users x 185771 items x 3091024 interactions	˜ 4.5 hours	Runtime error
279,789 users x 61,286 items x 300,000 interactions	˜ 20 second	MRR = 0.013
73,526 users x 114 items x 75,000 interactions	˜ 8 seconds	MRR = 0.092

Table 4.3: Implicit model

In summarize, the average time-consuming on training shows a significant reduction by decreasing the numbers of interactions. The result shows an around 40 times more efficiency while there is an around ten times decline in interactions for both models. Moreover, the performance on each baseline model changed based on different conditions. The RMSE score has an influential diminish at once a decreasing in the numbers of interactions, but there is no further improvement after lesser items are provided. On the other hand, the MRR score shows a run-time error before doing the first sampling by interaction. In considering an assumption on implicit models, a shorter item list makes sure most items left are more prevalent in users, and a higher MRR score is then generated. After an about 600 times diminish in numbers of items, an almost ten times improvement on MRR score is presented. As a result, the two different approaches on sampling bring two different enhancements on models; both arise a much shorter time consuming on model training. Explicit models get better performance with fewer interactions; an implicit model performs better while fewer items involved in the dataset. With regard to getting an overall well-performed baseline model, the final sample dataset applied in the following implementations is with 73,526 users x 114 items x 75,000 interactions.

RQ-2: Can we get a better prediction on baseline models by changing parameters rather than using the default parameter setting?

A further methodology to qualify our baseline is by varying the settings on parameters. Therefore, in this research question, we tried to tune the models with a variety of parameters set to check if the accuracy in models is improved. The idea of choosing the specific parameters in each experiment based on the character of each parameter. For instance, how is it works, and is it essential to the model? Furthermore, several testings are approached for measuring which combination in parameters setting leads to better performance. At last, the best parameter setting is generated and provided for each baseline in further progress.

The experiment set-up is based on the fewer items sample dataset and the various combinations in parameter setting for building models. There are five parameters chosen, which are n_iter, embedding_dim, batch_size, l2, and learning_rate. The definition of 'n_iter' is the number of iterations in training. A larger number of iterations lead to more in-depth training and able to get a smaller loss in fitting the training dataset. The 'embedding_dim' is the numbers of embedding dimensions to use for representing items; in this case, a higher dimension is regarded to fit on a dataset with more items. The 'batch_size' is the mini-batch size, representing how large the dataset splits each element for training. Moreover, 'l2' is the value of the loss penalty, which helps to alleviate the issue of over-fitting. Last of all, 'learning_rate' is the initial rate of learning a higher rate may provide a reduction in the training time. The following two sections provide the result generated by different parameters setting in an explicit and implicit model separately.

According to the experiments progress and result shown in table 4.4, the best parameter setting is to change numbers of iteration and the l2 penalty score. There are some characters we have found in parameters tuning. First, a higher 'embedding_dim' consumes a longer time on training and lower performance. Second. More iterations lead to a better performance at first; however, it causes over-fitting once the 'n_iter' getting too large(over than 100). Third, a smaller 'batch_size' takes a long time on training, but it provides an improvement in prediction with a smaller 'n_iter. ' In contrast, while the numbers of iteration increased a larger 'batch_size' is fit. Fourth, the default learning rate at 1e-2 and the penalty score at 1e-9 are the best thresholds because any increment or decline lowers the performance. In summarize, the best parameter setting is shown in table 4.4, that is with changes in n_iter from 10 to 100 and 'l2' from 0 to 1e-9. This criterion is provided for the baseline model.

Based on the experiments result in table 4.5, most parameters are changed from the original implicit models default to the BPR models default. The 'loss' changed to BPR at the beginning, which reduces the time-consuming on training; however, a decline in the MRR score draws. By decreasing the 'batch_size' from 256 to 128 rise an extended training time but better performance is returned. Then the increment on 'embedding_dim' leads to a further advancement on a prediction but also training time. Finally, The 'l2' penalty score

	Parameter setting	Time-consuming	Performance
Default setting	loss='regression', embedding_dim=32, n_iter=10, batch_size=256, l2=0.0, learning_rate=1e-2	11.2 seconds/ iteration	Train RMSE=0.324, Test RMSE=1.572
Best performance	loss='regression', embedding_dim=32, **n_iter=100**, batch_size=256, **l2=1e-9**, learning_rate=1e-2	8.08 seconds/ iteration	**Train RMSE=0.176, Test RMSE=1.350**

Table 4.4: Explicit model setting

helps with both time saving and better MRR score. In summarize, the best combination of parameters is similar to the BPR default but with ten times less on the numbers of iteration, which provided in the implicit baseline.

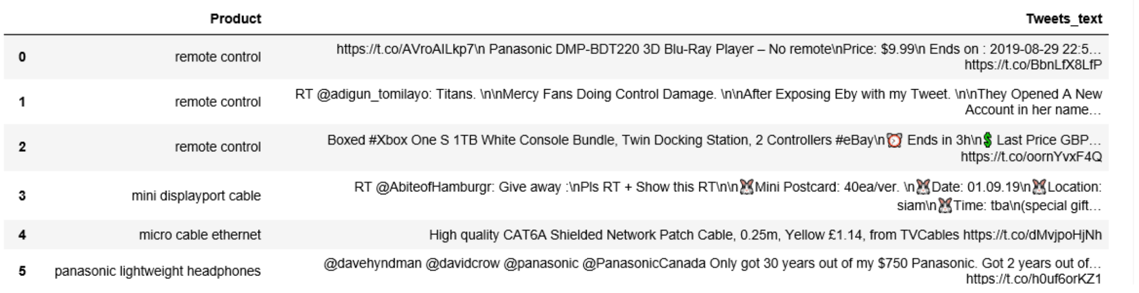
	Parameter setting	Time-consuming	Performance
Default setting	loss='pointwise', embedding_dim=32, n_iter=10, batch_size=256, l2=0.0, learning_rate=1e-2,	13.9 seconds/ iteration	Ave MRR=0.095
Best performance	**loss='BPR', batch_size=128, embedding_dim=64, l2=1e-9**, learning_rate=1e-2	40.3 seconds/ iteration	Ave MRR=1.0

Table 4.5: Explicit model setting

RQ-3: Does Elasticsearch help with removing noises in tweets?

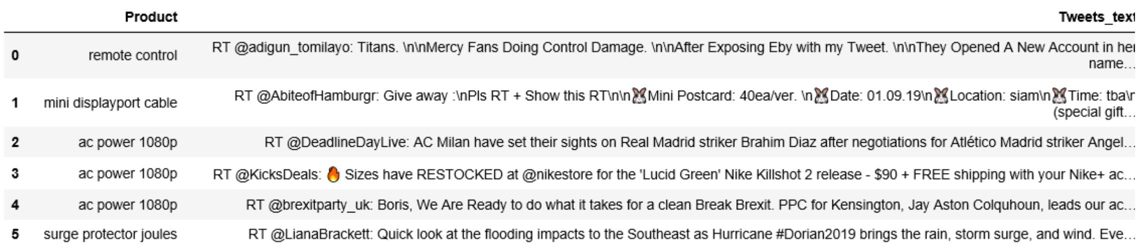
In the opinion of boosting baseline models with a social resource, it is crucial to gather only those tweets with informative content which can provide extra features in customers' opinions. Those noises in tweets not only rise an extend processing time but also a bias in model integration. The definition of noise is a tweet with the content irrelevant to the specific item sold on amazon.com. In this research, the methodology to remove noises is to process tweets through the Elasticsearch engine. There are mainly two functions in the search technique; indexing and searching. In the searching process, there are multiple filters set to gather only the tweets we need.

In this experiment, the inputs are the product titles from amazon dataset and the tweets gathered by Twitter REST API. The tweets are first indexed into the Elasticsearch engine, and then start the searching process based on the item titles. There are four experimental variables provided in the searching process. The previous two are generated based on filtering tweets relative to the item sold on amazon.com, which are the tweets filtered are in English, and there must be at least a 50% match between entities extracted from a specific item title and the content of a tweet. Then the other two rules are provided for removing advertisements and retweets. In removing advertisements, the condition based on a tweet must not contain 'http://' or an 'https://'. For instance, a tweet about 'New With Tags Under Armour Hustle UA Storm 3.0 Backpack Laptop School Bag Retweet



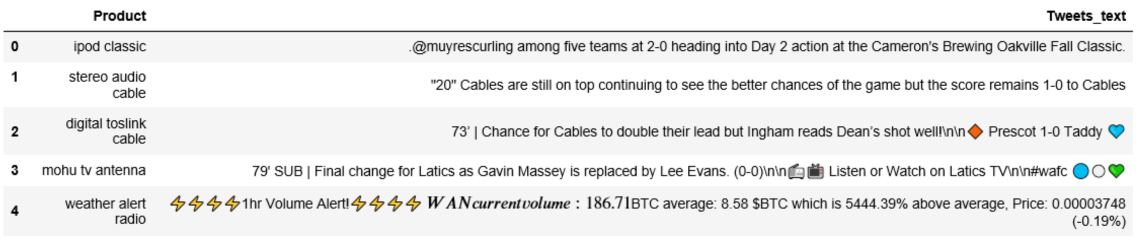
	Product	Tweets_text
0	remote control	https://t.co/AVroAILkp7\n Panasonic DMP-BDT220 3D Blu-Ray Player – No remote\nPrice: $9.99\n Ends on : 2019-08-29 22:5… https://t.co/BbnLfX8LfP
1	remote control	RT @adigun_tomilayo: Titans. \n\nMercy Fans Doing Control Damage. \n\nAfter Exposing Eby with my Tweet. \n\nThey Opened A New Account in her name…
2	remote control	Boxed #Xbox One S 1TB White Console Bundle, Twin Docking Station, 2 Controllers #eBay\n⏰ Ends in 3h\n💲 Last Price GBP… https://t.co/oornYvxF4Q
3	mini displayport cable	RT @AbiteofHamburgr: Give away :\nPls RT + Show this RT\n\n🐰Mini Postcard: 40ea/ver. \n🐰Date: 01.09.19\n🐰Location: siam\n🐰Time: tba\n(special gift…
4	micro cable ethernet	High quality CAT6A Shielded Network Patch Cable, 0.25m, Yellow £1.14, from TVCables https://t.co/dMvjpoHjNh
5	panasonic lightweight headphones	@davehyndman @davidcrow @panasonic @PanasonicCanada Only got 30 years out of my $750 Panasonic. Got 2 years out of… https://t.co/h0uf6orKZ1

Figure 4.1: Filter tweets in English with 50% of entities match



	Product	Tweets_text
0	remote control	RT @adigun_tomilayo: Titans. \n\nMercy Fans Doing Control Damage. \n\nAfter Exposing Eby with my Tweet. \n\nThey Opened A New Account in her name…
1	mini displayport cable	RT @AbiteofHamburgr: Give away :\nPls RT + Show this RT\n\n🐰Mini Postcard: 40ea/ver. \n🐰Date: 01.09.19\n🐰Location: siam\n🐰Time: tba\n(special gift…
2	ac power 1080p	RT @DeadlineDayLive: AC Milan have set their sights on Real Madrid striker Brahim Diaz after negotiations for Atlético Madrid striker Angel…
3	ac power 1080p	RT @KicksDeals: 🔥 Sizes have RESTOCKED at @nikestore for the 'Lucid Green' Nike Killshot 2 release - $90 + FREE shipping with your Nike+ ac…
4	ac power 1080p	RT @brexitparty_uk: Boris, We Are Ready to do what it takes for a clean Break Brexit. PPC for Kensington, Jay Aston Colquhoun, leads our ac…
5	surge protector joules	RT @LianaBrackett: Quick look at the flooding impacts to the Southeast as Hurricane #Dorian2019 brings the rain, storm surge, and wind. Eve…

Figure 4.2: Remove tweets contain URL



	Product	Tweets_text
0	ipod classic	.@muyrescurling among five teams at 2-0 heading into Day 2 action at the Cameron's Brewing Oakville Fall Classic.
1	stereo audio cable	"20" Cables are still on top continuing to see the better chances of the game but the score remains 1-0 to Cables
2	digital toslink cable	73’ | Chance for Cables to double their lead but Ingham reads Dean’s shot well!\n\n🔶 Prescot 1-0 Taddy 💙
3	mohu tv antenna	79' SUB | Final change for Latics as Gavin Massey is replaced by Lee Evans. (0-0)\n\n📻📺 Listen or Watch on Latics TV\n\n#wafc 🔵⚪💚
4	weather alert radio	⚡⚡⚡⚡1hr Volume Alert!⚡⚡⚡⚡ $WANcurrentvolume: 186.71$BTC average: 8.58 $BTC which is 5444.39% above average, Price: 0.00003748 (-0.19%)

Figure 4.3: Remove retweets

$19.99 https://t.co/75wNpj1kag’ is removed due to ’https’ shows in the content. On the other hand, a retweet is removed by ’RT @’ must not be included. For example, ’RT @Myrnastwit: Amazon will pay $0 taxes on $11,200,000,000 profit in 2018. So, the server at McDonalds will pay more in taxes than Amazon. . . ’ is removed due to ’RT @’ in the tweet.

Based on the conditions in Elasticsearch, a tweet can be stored while it contains at least 50% of entities in contents, without any URL link, the contents are in English, and it is not a retweet. However, due to the difficulty in proving whether the search engine is effective, there are two directions of results provided. Table 4.6 shows the changes in quantity, and figure 4.1 to 4.1 display the content of tweets left after layers of filters. In summarize, despite there are still tweets irrelevant gathered in results, the Elasticsearch technique helps with removing most noises in tweets.

Removal process	Number of results	Results
Filter tweets in English with 50% of entities match	82	Figure 4.1
Remove tweets contain URL	46	Figure 4.2
Remove retweets	5	Figure 4.3

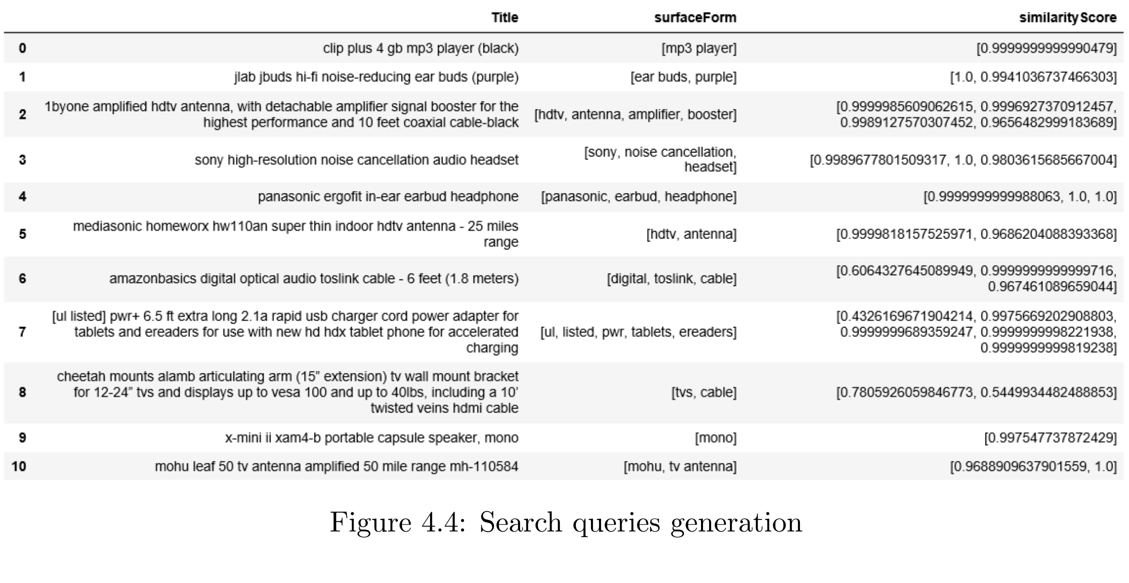
Table 4.6: Elasticsearch filtering

RQ-4: Does entity parser by DBpedia help with generating a more specific search query about an item in searching?

To the extent of searching more useful tweets, it is crucial to provide an appropriate query relevant to the particular item. Despite a text processing has been approached to the original product titles, there are still some titles contain meaningless terms in the result. Furthermore, most product titles are too long; the average length of terms of a specific product after tokenization, normalization, and terms removal are about 88 which is hard to summarize the meaning of the titles. For instance, if a product title is 'videosecu tv wall mount articulating arm monitor bracket for most 12"-24", some up to 27" lcd led plasma flat panel screen tv with vesa 100/75mm ml10b 1e9' it could be hard to summarize what exactly the product is. While those terms provided for gathering tweets are not specific enough, many noises are collected. Therefore, a proposed entity parser is provided to improve the quality of queries.

The definition of a more specific query in this project is to generate a list with fewer terms than the original title. Moreover, each term should have a meaning about the item. DBpedia is a resource implementing the entity parser through searching the keywords from an unstructured text with the Wikipedia source, and those keywords are collected as entities. Each product title is input as an unstructured text, and since almost every keyword on Wikipedia has a specific definition, most meaningless terms are likely to be removed.

In processing the DBpedia resource, the inputs are the product titles, and the outputs are entities retrieved through the entity parser. A better result depends on a shorter length of the query, and the query aims to recognized meaningful. In the comparison of the length in terms, there is a significant decline after the entity parser. During the experiment, data processing is first applied, and then the DBpedia process, the average numbers of entities detected is 3, which is an around 30 times cut back form only access basic data processing (88 terms). The result shows in figure 4.4, taking row 2 as an example, the title was '1byone amplified hdtv antenna, with detachable amplifier signal booster for the highest performance and 10 feet coaxial cable-black' with some useless terms such as ['with', 'for the highest performance']. The result generated after DBpedia is [hdtv, antenna, amplifier, booster], which is much more informative and specific.



	Title	surfaceForm	similarityScore
0	clip plus 4 gb mp3 player (black)	[mp3 player]	[0.9999999999990479]
1	jlab jbuds hi-fi noise-reducing ear buds (purple)	[ear buds, purple]	[1.0, 0.9941036737466303]
2	1byone amplified hdtv antenna, with detachable amplifier signal booster for the highest performance and 10 feet coaxial cable-black	[hdtv, antenna, amplifier, booster]	[0.9999985609062615, 0.9996927370912457, 0.9989127570307452, 0.9656482999183689]
3	sony high-resolution noise cancellation audio headset	[sony, noise cancellation, headset]	[0.9989677801509317, 1.0, 0.9803615685667004]
4	panasonic ergofit in-ear earbud headphone	[panasonic, earbud, headphone]	[0.9999999999988063, 1.0, 1.0]
5	mediasonic homeworx hw110an super thin indoor hdtv antenna - 25 miles range	[hdtv, antenna]	[0.9999818157525971, 0.9686204088393368]
6	amazonbasics digital optical audio toslink cable - 6 feet (1.8 meters)	[digital, toslink, cable]	[0.6064327645089949, 0.9999999999999716, 0.967461089659044]
7	[ul listed] pwr+ 6.5 ft extra long 2.1a rapid usb charger cord power adapter for tablets and ereaders for use with new hd hdx tablet phone for accelerated charging	[ul, listed, pwr, tablets, ereaders]	[0.4326169671904214, 0.9975669202908803, 0.9999999689359247, 0.9999999998221938, 0.9999999999819238]
8	cheetah mounts alamb articulating arm (15" extension) tv wall mount bracket for 12-24" tvs and displays up to vesa 100 and up to 40lbs, including a 10' twisted veins hdmi cable	[tvs, cable]	[0.7805926059846773, 0.5449934482488853]
9	x-mini ii xam4-b portable capsule speaker, mono	[mono]	[0.997547737872429]
10	mohu leaf 50 tv antenna amplified 50 mile range mh-110584	[mohu, tv antenna]	[0.9688909637901559, 1.0]

Figure 4.4: Search queries generation

RQ-5: Which query gathers a more relevant result in tweets; search by the first word in titles or all entities or each entity?

While a more relevant tweet is collected, it provides a more impressive feature in a specific item. In the implementation of gathering tweets by REST API, there are three types of query approached, and we aim to find relatively useful results. The definition of a more informative tweet in this research is a tweet with an user-based opinion, and the sentiment score of the text is assumed to identify this feature. While the emotion compound score of a tweet is in neutral, this may not be considered as informative content from the user. Apart from this character, it is crucial to remove noises in tweets; therefore, we aim to gather a tweet with a non-zero sentiment score and not a noise.

The three different structures of a query are the inputs, which are the first word in the title after the text processing, each entity and all entities generated by the entity parser. These queries are then provided into the standard search API as the keywords. Moreover, all three cases have a consistent setup, the maximum objects gathered from each query is set as 1,000, and the maximum number of tweets to return per page is set as 100. The returns will then go through the Elasticsearch process in filtering tweets as the final outputs.

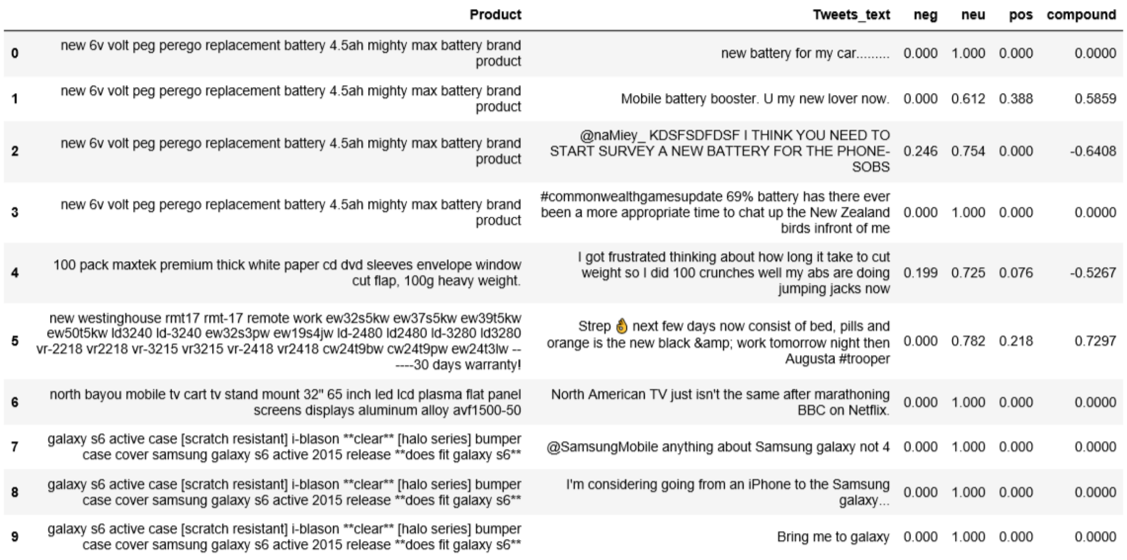
The results of searching by three different queries are shown in the figure 4.5-7, the ranking of the higher percentages of zero compound sentiment is searching by ['each entity', 'first word', 'all entities']. Although the tweets gathered seems not entirely relevant to the particular item they have matched, 'all entities' is considered as the most suitable query for searching more effective tweets.

RQ-6: Do social features boost or undermine the performance on baseline models after integration?

One of the aims of this research is trying to boost the baseline recommender models by extra information gathered from the social channel. Therefore, we interested in exploring whether a social impact draws influence on the user's perspective in the item. In the experiment, we focus on those items with a social feature generated from the relevant tweets. With each specific item, there are positive and negative impacts provided form the users. While the former reaction is detected we assume the rating score predicted by the model is underestimated, a boosting is proposed to the prediction value; in contrast, an undermine is suggested while the latter sentiment is presented. By testing the idea, the average error between predicted star ratings and actual star ratings is calculated by adding all the predicted value minus actual value together and then do averaging. From the perspective of this formula, a positive error means an overestimate, otherwise underestimate for a negative error. In summarize, our aim is to minimus the overall absolute error in predictions. There are two baseline models approached in this experiment, the explicit model with the best parameters setting and with a default setting. The former model presents a 2 of 3 accuracy rate based on the assumption for model integration. However, the overall RMSE increased from (Train RMSE 0.182, test RMSE 1.356) to (Train RMSE 0.346, test RMSE 1.357) after integration. On the later model, there is 5 of 6 accuracy rate with a slight improvement from (Train RMSE 0.333, test RMSE 1.584) to (Train RMSE 0.332, test RMSE 1.584). The results are shown in table 4.7.

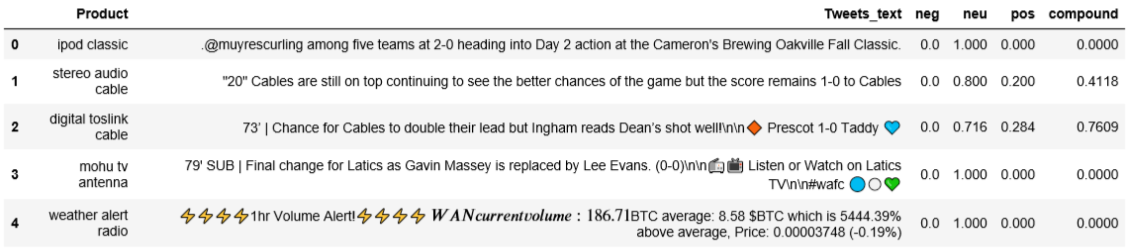
Product	Compound values in social feature	Model error Best parameters/ Default	Predicted—Actual Best parameters/ Default
mp3 player	1.4369	0.0937/ -0.8982	UE— OE/UE
ipod touch	0.1376	-0.1968/ -0.6711	UE— UE/UE
sony clock radio	0.6369	-0.2535/ -0.7384	UE— UE/UE
weather alert radio	-0.2732	-0.3322/ 0.2858	OE— UE/UE
digital toslink cable	0.7609	-0.2493/ -0.6640	UE— UE/UE
stereo audio cable	0.4118	-0.0653/ -1.5271	UE— UE/UE

Table 4.7: Model integration



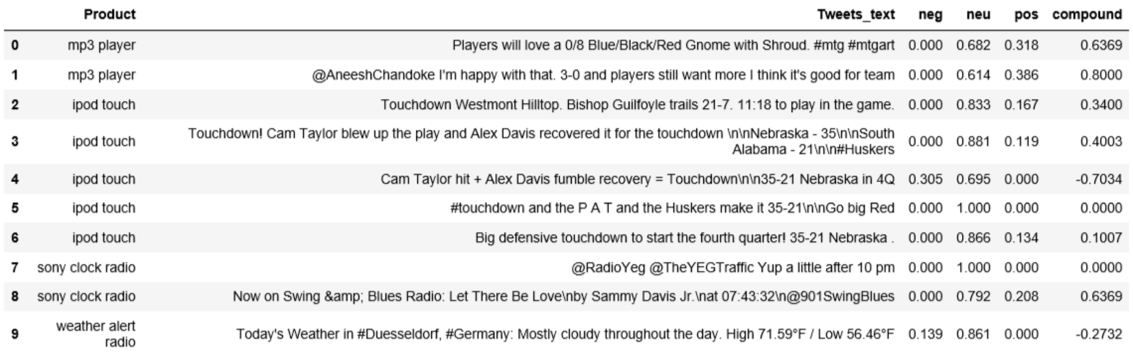
	Product	Tweets_text	neg	neu	pos	compound
0	new 6v volt peg perego replacement battery 4.5ah mighty max battery brand product	new battery for my car.........	0.000	1.000	0.000	0.0000
1	new 6v volt peg perego replacement battery 4.5ah mighty max battery brand product	Mobile battery booster. U my new lover now.	0.000	0.612	0.388	0.5859
2	new 6v volt peg perego replacement battery 4.5ah mighty max battery brand product	@naMiey_ KDSFSDFDSF I THINK YOU NEED TO START SURVEY A NEW BATTERY FOR THE PHONE-SOBS	0.246	0.754	0.000	-0.6408
3	new 6v volt peg perego replacement battery 4.5ah mighty max battery brand product	#commonwealthgamesupdate 69% battery has there ever been a more appropriate time to chat up the New Zealand birds infront of me	0.000	1.000	0.000	0.0000
4	100 pack maxtek premium thick white paper cd dvd sleeves envelope window cut flap, 100g heavy weight.	I got frustrated thinking about how long it take to cut weight so I did 100 crunches well my abs are doing jumping jacks now	0.199	0.725	0.076	-0.5267
5	new westinghouse rmt17 rmt-17 remote work ew32s5kw ew37s5kw ew39t5kw ew50t5kw ld3240 ld-3240 ew32s3pw ew19s4jw ld-2480 ld2480 ld-3280 ld3280 vr-2218 vr2218 vr-3215 vr3215 vr-2418 vr2418 cw24t9bw cw24t9pw ew24t3lw ------30 days warranty!	Strep 👌 next few days now consist of bed, pills and orange is the new black &amp; work tomorrow night then Augusta #trooper	0.000	0.782	0.218	0.7297
6	north bayou mobile tv cart tv stand mount 32" 65 inch led lcd plasma flat panel screens displays aluminum alloy avf1500-50	North American TV just isn't the same after marathoning BBC on Netflix.	0.000	1.000	0.000	0.0000
7	galaxy s6 active case [scratch resistant] i-blason **clear** [halo series] bumper case cover samsung galaxy s6 active 2015 release **does fit galaxy s6**	@SamsungMobile anything about Samsung galaxy not 4	0.000	1.000	0.000	0.0000
8	galaxy s6 active case [scratch resistant] i-blason **clear** [halo series] bumper case cover samsung galaxy s6 active 2015 release **does fit galaxy s6**	I'm considering going from an iPhone to the Samsung galaxy...	0.000	1.000	0.000	0.0000
9	galaxy s6 active case [scratch resistant] i-blason **clear** [halo series] bumper case cover samsung galaxy s6 active 2015 release **does fit galaxy s6**	Bring me to galaxy	0.000	1.000	0.000	0.0000

Figure 4.5: Search by the first word



	Product	Tweets_text	neg	neu	pos	compound
0	ipod classic	.@muyrescurling among five teams at 2-0 heading into Day 2 action at the Cameron's Brewing Oakville Fall Classic.	0.0	1.000	0.000	0.0000
1	stereo audio cable	"20" Cables are still on top continuing to see the better chances of the game but the score remains 1-0 to Cables	0.0	0.800	0.200	0.4118
2	digital toslink cable	73' | Chance for Cables to double their lead but Ingham reads Dean's shot well!\n\n🔶 Prescot 1-0 Taddy 💙	0.0	0.716	0.284	0.7609
3	mohu tv antenna	79' SUB | Final change for Latics as Gavin Massey is replaced by Lee Evans. (0-0)\n\n📻📺 Listen or Watch on Latics TV\n\n#wafc 🔵⚪💚	0.0	1.000	0.000	0.0000
4	weather alert radio	⚡⚡⚡⚡1hr Volume Alert!⚡⚡⚡⚡ $WANcurrentvolume : 186.71$BTC average: 8.58 $BTC which is 5444.39% above average, Price: 0.00003748 (-0.19%)	0.0	1.000	0.000	0.0000

Figure 4.6: Search by each entity



	Product	Tweets_text	neg	neu	pos	compound
0	mp3 player	Players will love a 0/8 Blue/Black/Red Gnome with Shroud. #mtg #mtgart	0.000	0.682	0.318	0.6369
1	mp3 player	@AneeshChandoke I'm happy with that. 3-0 and players still want more I think it's good for team	0.000	0.614	0.386	0.8000
2	ipod touch	Touchdown Westmont Hilltop. Bishop Guilfoyle trails 21-7. 11:18 to play in the game.	0.000	0.833	0.167	0.3400
3	ipod touch	Touchdown! Cam Taylor blew up the play and Alex Davis recovered it for the touchdown \n\nNebraska - 35\n\nSouth Alabama - 21\n\n#Huskers	0.000	0.881	0.119	0.4003
4	ipod touch	Cam Taylor hit + Alex Davis fumble recovery = Touchdown\n\n35-21 Nebraska in 4Q	0.305	0.695	0.000	-0.7034
5	ipod touch	#touchdown and the P A T and the Huskers make it 35-21\n\nGo big Red	0.000	1.000	0.000	0.0000
6	ipod touch	Big defensive touchdown to start the fourth quarter! 35-21 Nebraska .	0.000	0.866	0.134	0.1007
7	sony clock radio	@RadioYeg @TheYEGTraffic Yup a little after 10 pm	0.000	1.000	0.000	0.0000
8	sony clock radio	Now on Swing &amp; Blues Radio: Let There Be Love\nby Sammy Davis Jr.\nat 07:43:32\n@901SwingBlues	0.000	0.792	0.208	0.6369
9	weather alert radio	Today's Weather in #Duesseldorf, #Germany: Mostly cloudy throughout the day. High 71.59°F / Low 56.46°F	0.139	0.861	0.000	-0.2732

Figure 4.7: Search by all entities

Chapter 5

Conclusion

This research aimed to identify effective features gathered from the social channel to boost the performance on the recommender models. One of the significant problems causes the low performance on making the prediction is the cold-start issue due to a lack of information from a new customer or a new item. By solving this issue, there comes an idea in considering social media as an extra resource for retrieving more user's opinions on a new item. In processing the social feature extraction, we focus on collecting higher-quality information from the social platform; therefore, the implementation of searching, indexing, and creating query have been done. Moreover, the resource retrieved form social media are quantified by sentiment analysis to provide into the model integration. At last, some strategies are achieved in observing the relationship between the original recommender system and the social features for further recommender model extension.

In this research, the basic recommender system is first implemented by approaching an open resource, spotlight. After building both explicit and implicit model, an optimization in model performance boosting and training time reduction is provided through dataset sampling and parameter tuning process. With regards to filter tweets with more information contented, the Elasticsearch technique is approached in processing both indexing and searching function on removing noises in tweets. Moreover, the real-time social data collection by REST API is implemented through different types of search queries. Therefore, the optimization of generating a more specific search query with a DBpedia entity parser is implemented. The further discussion in choosing a better query to extract more valuable tweets is displayed. By processing the sentiment analysis with VADER, the social features are retrieved to complement the data sparsity issue in amazon-based recommender models.

Despite the methodologies and strategies approached in this research intends to retrieve more useful tweets, only a small amount of tweets is gathered. This induces a limitation in model integration due to a lack of samples in social features leads to the difficulty in identifying the threshold to set in boosting or decreasing the predicted rating score to get it closer to the actual score. A proposed idea for further implementation is to consider more social interactions from a tweet such as a review left from a friend. This helps with gathering more information from the specific, meaningful tweet and also involving more people's opinions on a particular product. Furthermore, due to the limit amount of tweets is gathered by the REST API in each iteration, there might be some information lost from the single searching. Therefore, while there seem to be extra tweets have not gathered in a searching, more iterations of searching by the specific query could be achieved. On the other hand, due to the time limitation in completing the final model integration process, the implicit model is not involved in the final integration. Based on the idea of a hybrid feedback model, an advanced approach is to merge the explicit model, implicit model, and the social features together.

This product can estimate a customer's preference on any specific product sold on the amazon by the recommender model trained by the historical amazon dataset. If the user is a new visitor to the website page, then some generally popular products are recommended. On the other hand, to complement the lack of information on a new item, there is a real-time search function for gathering tweets from Twitter. We could gather extra information on any particular item from social media; for instance, if there is a new product [I1]. During the process, the product title will first be generated into a query for the instant tweets retrieval, and then an indexing process is run before the searching process. All conditions for noises removal are set in the function to extract only relevant tweets. As a result, this product can provide a summarize of users' opinions on the new goods [I1] which are the overall sentiment distribution, such as how positive or negative people feel about the item.

In summarize the implementation of this project, the connection between the basic recommender system and the social resource is built. Based on the functions provided in this product, the future work can be done to involve more categories of social features which helps with identifying more feedbacks from customers. Moreover, the integration between the explicit model, implicit model, and social features are expected to be further approached.

Bibliography

[1] Daniel Cea, Jordi Nin, Rubén Tous, Jordi Torres, and Eduard Ayguadé. Towards the cloudification of the social networks analytics. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 192–203. Springer, 2014.

[2] Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2015.

[3] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 257–260. ACM, 2010.

[4] Rishab Aiyer Ghosh and Lun Ted Cui. System and method for customizing analytics based on users media affiliation status, September 27 2016. US Patent 9,454,586.

[5] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558. ACM, 2016.

[6] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[7] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[8] Yanxiang Huang, Bin Cui, Wenyu Zhang, Jie Jiang, and Ying Xu. Tencentrec: Real-time stream recommendation in practice. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 227–238. ACM, 2015.

[9] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[10] FO Isinkaye, YO Folajimi, and BA Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015.

[11] Shuhui Jiang, Xueming Qian, Tao Mei, and Yun Fu. Personalized travel sequence recommendation on multi-source big social media. *IEEE Transactions on Big Data*, 2(1):43–56, 2016.

[12] Oleksii Kononenko, Olga Baysal, Reid Holmes, and Michael W Godfrey. Mining modern repositories with elasticsearch. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 328–331. ACM, 2014.

[13] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456. ACM, 2009.

[14] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[15] Feng Li and Timon C Du. Who is talking? an ontology-based opinion leader identification framework for word-of-mouth marketing in online social blogs. *Decision support systems*, 51(1):190–197, 2011.

[16] Ian MacKenzie, Chris Meyer, and Steve Noble. How retailers can keep up with consumers. *McKinsey & Company*, 2013.

[17] Andrew J McMinn and Joemon M Jose. Real-time entity-based event detection for twitter. In *International conference of the cross-language evaluation forum for european languages*, pages 65–77. Springer, 2015.

[18] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE, 2008.

[19] Joseph E Phelps, Regina Lewis, Lynne Mobilio, David Perry, and Niranjan Raman. Viral marketing or electronic word-of-mouth advertising: Examining consumer responses and motivations to pass along email. *Journal of advertising research*, 44(4):333–348, 2004.

[20] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[21] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM, 2008.

[22] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[23] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[24] Neel Shah, Darryl Willick, and Vijay Mago. A framework for social media data analytics using elasticsearch and kibana. *Wireless Networks*, pages 1–9, 2018.

[25] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating" word of mouth". In *Chi*, volume 95, pages 210–217. Citeseer, 1995.

[26] Gabriel Vigliensoni and Ichiro Fujinaga. Automatic music recommendation systems: Do demographic, profiling, and contextual features improve their performance?. In *ISMIR*, pages 94–100, 2016.

[27] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. Representing and recommending shopping baskets with complementarity, compatibility and loyalty. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1133–1142. ACM, 2018.

[28] Robert A Westbrook. Product/consumption-based affective responses and postpurchase processes. *Journal of marketing research*, 24(3):258–270, 1987.