NTNU
Norwegian University of
Science and Technology

Semantic User Behaviour Prediction in Online News

Applying Topic Modeling, Community Detection, and User Modeling for News Recommendation

Nina Kjekstad
Elida Karina Reknes

Master of Science in Computer Science
Submission date: June 2018
Supervisor: Jon Atle Gulla, IDI
Co-supervisor: Lemei Zhang, IDI
Peng Liu, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

In recent years, predicting user behaviour has become increasingly important within the news recommendation area. Knowledge of behavioural patterns offers valuable insight for developing efficient and user friendly services, ensuring good user experiences for users and increased revenues for companies. Though very useful, these user recommendations can be difficult to make and the news domain poses its own, unique challenges for the task. Challenges such as short life span for item recommendations, sparse connections to large item vocabularies, and volatile networks and user behaviours are prominent in news recommendation. This creates performance difficulties for traditional recommendation methods, especially due to the need to recommend unseen items.

By employing topic modeling on the rich text objects provided by the news domain, new articles introduced in the system can be compared to old articles in a meaningful way. Organizing these articles in clusters based on topic proportions, useful topic combinations can be elicited. Using these clusters to create user models based on composite interests, newly introduced articles can be recommended to users.

While a key finding is the conflict between optimizing goals for subtasks and optimal performance in the prediction task, implementing these technologies does lead to improvements of user prediction results. Different combinations of parameters provides prediction results with very different qualitative characteristics, but comparing to predictions using original labels and random recommendations, the models obtain better Mean Average Precision (MAP) and Ranking Accuracy (RA) than both. With models using both topic modeling and clustering, RA shows improvements on random, ranged from 34%-64%, for models which performed well. Other models gains improvements ranging from 18%-278 % on top 3, 5, 10 and 30 positions for MAP, but declines in performance on RA by $18\% - 28\%$. Considering the characteristics of each prediction evaluation metric, RA appears to better capture the quality of recommendations.

Sammendrag

Behovet for å forutse brukerhandlinger for å anbefale artikler har blitt tydeligere de siste årene. Forståelse av brukermønste kan gi lønnsom innsikt som forbedrer anbefalingssystemer sin evne til å øke brukere sin brukeropplevelse, og dermed inntekt. Å gi nyttige brukeranbefalinger kan være vanskelig, og nyhetsdomenet sine tendenser til uberegnelige og sammensatte brukermønste medfører egne utfordringer for oppgaven. Stort spenn i artikkeltyper og tema de omhandler kombinert med mengden artikler som publiseres hver dag, resulterer i få koblinger mellom brukere og artikler sammenlignet med størrelsen på artikkelvokabularet. I tillegg til det korte tidsrommet hver artikkel vil være interessant å anbefale, gjør dette det vanskelig å bruke tradisjonelle anberfalingsmetoder.

Ved å lage semantiske modelleringer av abstrakte temaer, som kan modellere artikler sett i lys av innhold, kan nye artikler sammenlignes med gamle. Organiseres disse artiklene i klynger som deler temakombinasjoner, kan brukere få anbefalinger som samsvarer med innhold i klynger de deler bruksmønster med.

Hovedfunnet er uoverensstemmelser mellom optimale resultater i delprosessene og optimale resultater for brukeranbefalinger. Til tross for dette, og store variasjoner i kvalitative egenskaper for forskjellige parametre, viser modellene evnen til å forbedre anbefaling av nye artikler sammenlignet med bruk av originale kategorier og tilfeldig anbefalinger. Evaluering viser at enkelte av modellene får en økning for evalueringsmetrikken RA på 34%-64%. Den andre gruppen modeller får en økning på 18%-278% for MAP i posisjonene 3, 5, 10 og 30, men redusert RA med 18%-28%. Med hensyn til karakteristikkene til evaluerignsmetrikkene, synes RA å bedre speile behov i anbefalingsscenarioet.

Preface

This thesis documents research conducted in the period of January to June 2018, as a partial fulfillment of the degree Master's of Science at Norwegian University of Technology and Science (NTNU). The submission is a part of the course TDT4900 Computer Science, and the work has been performed at the Department of Computer Science (IDI) as a part of the SmartMedia project. We would like to express our gratitude to our supervisors Professor Jon Atle Gulla, and PhD candidates Lemei Zhang and Peng Liu, for productive guidance and useful feedback on our work.

Trondheim, June 7, 2018

Nina Kjekstad Elida Karina Reknes

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

The number of people using online services daily has been increasing vastly over the years, with the amount of users accessing their news online increasing along with it [1]. News outlets no longer rely mainly on paper format news, and some have even become online-only papers after decades of publishing in order to stay in production [2]. Main sources of income are sales of subscriptions and advertisements, with the latter being traditionally essential and lately declining in revenue [3]. Even before the current level of digitalization of newspapers, advertisement income were being redirected to online services providing free space and markets for classified ads [4], and now news outlets are competing with technology giants for ad revenues online [3]. With so much of the news industry conducted online, clicks and individual article reads will generate much of the advertisement income where printed classified ads did before.

Along with this change in how users consume their news, competition has changed within the news field. While reading the paper delivered in your mailbox each morning is a natural habit, online news provides users with hundreds of potential news sources covering partly overlapping news. When users can get the same service from many different providers, increasing user satisfaction and loyalty is crucial [5]. A personalized service can achieve this while also alleviating the issue of information overload, an increasingly frequent occurrence with the vast increase in information available online [5]. A common tool to provide this personalized service which can improve user experience and loyalty is Recommender Systems (RSs).

1.1.1 Problem Outline

This opportunity to increase user satisfaction and company revenues simultaneously, has resulted in the field of user modeling and predicting user behaviour gaining an increasing interests in many domains, news recommendation among

them. While many recommender system challenges are shared across domains, some specific domain characteristics of users or items can result in amplified consequences or unique combinations of these challenges. The field of recommender systems, and consequently user modeling, has struggled with making useful recommendations on sparse data and computational efficiency on large datasets since the start and across all domains [6]. Sparse data can be a result of not having enough information about a user's activity in a large item space, or about the items that are being recommended. Both these situations can inhibit a system from providing good recommendations, and are relevant issues to address when recommending news articles to users.

The expansion of online news services and number of users has generated a great deal of data, available for user and purchase history analysis when creating better services. In the scenario of news recommendations, collected data available for item modeling may often include the articles themselves, user read history, authors and manual category tags. In terms of user activity information, the news field intuitively has a certain advantage over typical movie streaming or online store recommendation services. Users often read several news articles a day, but many users may visit an online store a few times a month. While the news area may be able to collect more user activity data than many other services, it also maintains a high pace of item addition. Hundreds of articles may be published each day for a single paper, with most users reading only a handful, and the user activity matrices will still be significantly sparse.

Though the amount of item and user data collected in the news field is often vast, utilizing it for recommendations is not always easy. Traditional methods of recommendations such as collaborative filtering, recommends items to users based on items that similar users have favoured, often even if the item was consumed or rated some time ago. In the news field, unlike many other domains, recency of content recommended is one of the most important aspects, and recommendation of older items will generally be of little interest to most users due to the commonly short expiration date of relevance for news articles [1]. Thus, the recommendations cannot benefit from the higher frequency of user activity by traditional methods. This poses the challenge of exploiting historical user activity in order to generate recommendations on items that have yet to be read by sufficiently many users to do user comparisons. Making recommendation to users or with items of which little or no information is known is referred to the "cold start" problem.

Another data collection deviation between the news domain and certain other recommendation fields, is the lack of explicit feedback [7]. Users rarely rate an article on a scale from 1 to 5, or even provide a binary positive or negative feedback after reading. Even if they were asked to do so, content and topics in news may make it difficult for users to properly reflect their preference for an article, or where it stems from. A reader can disagree with the sentiment expressed by an article, while still being interested in reading it due to interests within the topics it discusses. With the frequency at which users read articles, it would also be immensely disrupting to request feedback after each article. In recent times, with many news outlet having their own comment sections or linking to news content

on social media, some feedback could theoretically come from observing responses to mentions of news articles. While these comments could be interesting in a sentiment analysis context, mapping user comments from separate data sources to users within a system can be difficult. Additionally, the previously mentioned tendency for news readers to be interested in reading articles which deviates from their own topical interests, or sentiment about a topic, could make this work more confusing than helpful. Depending on this type of feedback is also not viable in a large-scale news recommender system, as only a marginal portion of the readers are likely to engage in social media responses, making it specialized for users with very specific behavioural patterns. Consequently, feedback is generally based on whether a user clicks on, or reads, an article or not, data which can be assumed non-discriminatory available for all users with persistent user identification. This lack of explicit feedback to indicate whether a user in fact liked read articles, combined with the volatility of expressed user interests, leads to additional challenges when eliciting user interests in the news field recommendations [7]. Especially when noting that the news domain is also prone to users reading articles that diverges from their expressed personal interests, such as breaking news or an article linked by a friend [1]. This can create additional noise or misinterpretations in user interest models.

1.1.2 Approach

To summarize, the problem at hand includes deriving user models from topical interests without much user feedback and recommending articles with little to no user reading history, due to the high turnover rate of news articles' relevance. Approaching this problem, knowledge from two developing fields in particular constitutes the main focus of this thesis, accounted for in the following.

Parallel to the developments and data volume increase in the news domain, the extraction of abstract topics from text has been a growing field over the last decades. While some news articles are published with manually added tags representing topics or categories, these tags are often few, very overarching and lacking in full topical coverage. With the news field's easy access to rich text that not only describes the item to be recommended, but is the item itself, topic modeling is a natural tool for discovering similarities and differences across articles. This can be used as a way to circumvent the lack of information about item consumption. A new published article will always offer text that can be analyzed through topic modeling, making topic modeling a consistently useful tool in the news recommendation area. By enabling the comparison of new and old items, the disadvantage brought on by a lack of data to perform user comparisons will hopefully be minimized. Topic modeling is also a tool for reducing the impact of sparsity and efficiency issues, through representing user interest as accumulative topic interest instead of relations to each read article in the item space.

At the same time, researchers have utilized and improved on community detection methods across many domains and applications. Community detection can extract useful patterns and groupings, often from networks where nodes represent users with attached features in the scenario of recommendation systems. The use-

fulness of community detection is not restricted to use within recommender systems though, and has been researched in contexts such as biology and physics as well as more social networks. Discovering the underlying community structure within a news article network, these structures can be used by leveraging the common behavioural patterns to predict a single user's behaviours or clicks. This more sophisticated manner of mapping a user's behaviour to composite topic patterns, can hopefully mitigate some of the sparsity issues faced in RSs, as well as improving the general quality of recommendations.

1.2 Research Context

This Master Thesis report is written as the final requirement to obtain a Master of Science degree at NTNU within the field of Computer Science.

1.2.1 NTNU SmartMedia Project

The project is a part of the SmartMedia Program[1], a project run by the Department of Computer Science at NTNU. SmartMedia collaborates with the Norwegian media industry and investigates topics that aid the industry, such as the use of semantics and linked data in large-scale real-time news recommendation. Predicting user behaviour is a central part in the recommendation of relevant content with topics of interests for the user, as well as in marketing, which has become critical in the media industry over the last few years.

1.3 Research Questions

This project aims to improve news article recommendations through technologies such as topic modeling and community detection. These methods, and their ability to improve user models and prediction results, are also the main focus of the research questions.

1. **What types of topic models or semantic approaches can be used to extract the temporal user-item interactions to improve the performance of news recommendation?**

   When extracting item information, it is important to generate a model which efficiently can infer comparable information for items introduced at a later time. Topic models and embeddings can provide this for textual items through detecting common patterns in training, and applying them to unseen data to label them accordingly. A key question becomes how these models can aid in discovering and recommending new and relevant documents to users without attached read history about the item, encompassing the effect the different models and varying choices of topic granularity has on the prediction results.

[1] https://www.ntnu.no/wiki/display/smartmedia/

2. **How can topic-based community detection and dimensionality reduction technology help solve the sparsity issue and improve efficiency in recommender systems?**

   Excessively sparse data is a challenge when recommending items to users, resulting in difficulties such as lack of scalability and efficiency reduction. By using topic modeling and embedding, articles will be presented in a lower dimensional space, with topically similar articles having similar representations. How will this topical dimensionality reduction impact the sparsity challenges, together with clustering employed on said article representations? Additionally, this may aid in increasing the coverage of documents being recommended, as documents with few interactions in sparse user-item matrix are traditionally rarely recommended. The research will explore how clusters found through community detection on topic modeled documents compares to the manually labeled categories, and how this may impact the quality of recommendation.

3. **How can community detection technology be used to mitigate item-side cold-start issues in news recommendation?**

   Expecting no or insufficient read history about articles due to the short time frame of relevance, how can community detection aid in circumventing the need for user-item interaction before a new item can be recommended? This includes developing content based user models by discovering optimal clusterings for the topical items and utilizing user history, in order to extract useful recommendations among cold items.

1.4 Contributions

This section will briefly present the thesis' main contributions, in answer to the aforementioned research questions. Firstly, notable gains in prediction quality is observed when performing dimensionality reduction through semantic analysis, and the two investigated generative processes for extracting topics, namely Latent Dirichlet Allocation (LDA) and TopicVec, is proven to perform approximately equal to each other on the given data set. With models using both topic modeling and clustering, the prediction metric RA shows improvements on random, ranged from 34%-64%, for models which performed well. Other models gains improvements ranging from 18%-278 % on top 3, 5, 10 and 30 positions for MAP, but declines in performance on RA by $18\% - 28\%$. Considering the characteristics of each prediction evaluation metric, RA appears to better capture the quality of recommendations.

However, the evaluation of predictions must also be seen in light of the complications related to simulating a real news domain, both in terms of language and sparsity in feedback, and that consequently lack of documented interaction does not implicate disfavour. Nevertheless, some of the models do show improvements on cold start items recommendations, compared to random and a method utilizing

original labels. Moreover, further dimensionality reduction with topical community detection by traditional clustering shows very varying performance, and results indicate that more sophisticated evaluation metrics is needed in parameter selection and optimization of sub-processes. Lastly, as mentioned, community detection yielded results of varying quality depending of parameter choice, and in addition to more sophisticated optimization methods, less conventional clustering methods in this sub-process may be fruitful to serve the main process better. With that being said, the communities found do show tendencies of not simply replicating the original labels, being evidence of the insight achievable from the process.

1.5 Report Structure

This report consist of seven chapters. This chapter, chapter 1, explains the background and motivation for the project, as well as a problem outline and research goals and questions. The rest of the report is structured as follows: Chapter 2 provides an overview of the basic theory used in the the report, followed by chapter 3 documenting a literature review of related works and comments how these are relevant for the project. In chapter 4, we describe the methodology, including the data sets, as well as introducing our experiments. Experiment sets up will be detailed in Chapter 5, while Chapter 6 presents and discusses the results. Finally, Chapter 7 concludes the research and outlines potential further work.

Chapter 2

Theoretical Background

This chapter will provide an overview of the basic theory for technologies and methods used in the the report.

2.1 Clustering

Clustering is the task of grouping data points, such that data points within a group (or cluster) are more closely related to each other than to those in other groups. The technique is used in many disciplines and can be considered as one of the core tasks in data mining, the process of discovering patterns and new information in large data sets [8].

One of many usages of clustering is community detection, often used to represent real community structures derived from graphs [9]. It is an important area of research, and can represent data sets from many different domains [10]. One such domain is social media applications, where a user's content may be partially decided by the activity of friends or communities consisting of similar users. If diligently implemented, this may open for exploration while also taking history of user's own preferences into account. However, this puts high demands on attributes and quality of the data set.

2.1.1 Clustering Algorithms

A numerous amount of clustering algorithms and variations exists, and which is the most suitable depends on a range of different features. These features include whether the use case requires many or few clusters and if this number of clusters can be predefined, if the cluster sizes are likely to be uneven, if the clusters' are of a non-flat geometrical shape, and the size of the data set in terms of scalability. Different clustering methods may yield quite different clusters, as exemplified in figure 2.1. In the news domain, a suitable clustering algorithm should be able to handle large data sets, many clusters of varying sizes and preferably also non-flat geometrical shapes. Two commonly practiced and relatively basic clustering

methods are k-Means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN), described below and presented in figure 2.1.

k-Means

k-Means is a frequently used baseline clustering algorithm, and is described in detail by Hartigan [11]. In short, the algorithm divides the data points into k clusters by minimizing the within cluster sum of squares. It requires a predefined number of clusters and is more suitable when the clusters are of even size and flat geometry, which is not optimal for the news domain. However, it is a simple and scalable algorithm, and may serve as a well-known general-purpose baseline algorithm.

DBSCAN

DBSCAN, clusters points based on density [12]. It groups points in areas of high density together based on a given value for the minimum number of surrounding data points required within a radius. Any points lying alone in areas of low density which does not meet the requirements of surrounding data points are marked as outliers. Due to this generic view, the algorithm can handle clusters of non-convex shapes, in contrast to k-Means. The algorithm is also scalable, can handle uneven cluster sizes and does not require a predefined number of clusters, making it promising for clustering in the dynamic and composite news domain.

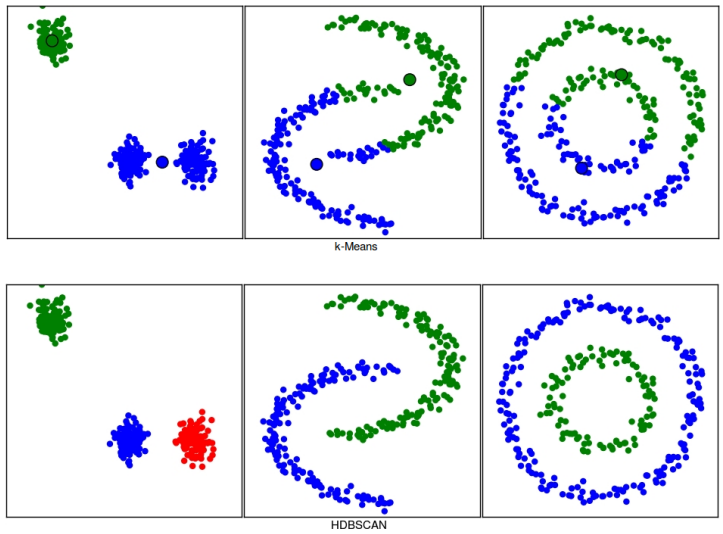

Figure 2.1: Comparison of k-Means and DBSCAN[1].

[1] http://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/_images/plot_cluster_comparison_1.png

2.1.2 Clustering Evaluation

The validity of a cluster is an assessment of the confidence level surrounding whether the extracted clusters are actually true [13]. In contrast to methods where the groupings of the data points are learned from knowledge of correctly assigned groups, or labels (supervised learning), clustering methods may often be tasked with grouping unlabeled data points, or "learning without a teacher" (unsupervised learning) [14]. Accordingly, quality of an unsupervised clustering is measured based on internal evaluation [15], solely using the data clustered, for instance calculating metrics for cohesion within clusters and separation between. Several indices exists to evaluate cluster validity, and two of these are the Silhouette Coefficient [16] and Calinski-Harabasz (C-H) Score [17] (table 2.1).

Evaluation Metric	**Equation**	**Parameters**
C-H index	$s(k) = \frac{Tr(B_k)}{Tr(W_k)} \times \frac{N-k}{k-1}$	N is number of points W_k is d(nodes to centroid) B_k is d(centroid to centroid)
Silhouette Coefficient	$s = \frac{b-a}{max(a,b)}$	a: mean(d(nodes in same cluster)) b: mean(d(nodes in next nearest cluster))

Table 2.1: Clustering evaluation metrics.

The Silhouette Coefficient is a ratio value of the mean distance values between the points in the cluster to its own points, as a measure of cohesion, and to the closest other cluster, as a measure of separation. The C-H index is a similar metric, being defined as ratio of between-clusters dispersion mean and the within-cluster dispersion, but is less computationally demanding than the Silhouette Coefficient, which might be advantageous when working with large data sets. Both metrics favor dense and well-separated clusters, and optimal values are max. The Silhouette Coefficient will always be in the range -1 to 1, which makes it easier to compare across implementations. The C-H penalizes high values of k, shown in the $\frac{N-k}{k-1}$-element. This could affect the performance of the metric in such a large dataset, as some lower values of k may get the same or better score based on this element. Both methods' performance are skewed to give higher scores to convex clusters, such as those more aptly found by k-Means. This must be taken into consideration when comparing results across clusterings algorithms.

2.2 User Modeling and User Behaviour Prediction

User modeling is the field of creating a representation of a user's preferences and behaviour, based on information collected and features generated by a model [18]. Usage of user models includes Information Retrieval (IR) [19], adapting links or content on pages [20], system guides [21] and RSs. The latter, being the focus of

this research, will be further described in this section, after briefly describing some of the basics in construction of a user model. The main building block in a user model is collected knowledge of user behaviour through direct or indirect feedback.

2.2.1 User Feedback

User feedback is commonly separated into implicit and explicit feedback [22], where explicit feedback is feedback the user is directly involved in, and aware of, providing. Examples include a user rating an item they have bought or movie they have watched, indicating a quantifiable favor of the item. Other examples could include asking users questions about demographic attributes, or to choose interesting topics from a list. These methods often have the advantage of providing correct information, as the user is able to explicitly control how they are presented. Some issues are however present. First of all, it puts task demands on the user. In the news domain for instance, users will often read several articles each day, but likely not a significant portion of the total number of articles published. Users are also not likely to take the time to, or appreciate a prompt to, rate all articles they read within a day. This can contribute to very sparse matrices when creating user models based on item rate history [22]. This is expected to be especially prominent in the news domain due to volatility in articles' relevance and individuals' behaviours. Additionally, some users may not have optimal insight into their own interests, or a way to express them in such discrete formats as named categories or a 1 to 5 rating. Users may be interested in news within a topic, without having a particular interest or favour of a certain article within the subject. Users who provide contradictory item ratings are also an issue [23], as well as potential opinion changes over time.

With news domain users usually not being willing to rate the relevance of each read article, user behaviour patterns will be captured by interpreting various implicit signals to second-guess their interests and satisfaction [7]. The goal of implicit feedback is thus to extract useful information from user behaviours that can help improve the user model [6], without requiring explicit statements from users. Information collected from implicit feedback can range from simple records of search and item purchase history, to more detailed statistics and measurements of user behaviour. The latter have included comprehensive user behaviour data collection of metrics such as time spent on page and mouse movement [24][25]. These measures are often not available datasets or expensive to collect, and user activity in the form of purchase history and the like is more likely to be of practical use. Implicit feedback can be used both as a replacement for explicit feedback, or as a supplement to get more accurate user models. It has the advantage of being non-intrusive, thus it is not dependent on users being willing to respond frequently to questions, or performance-wise skewed in favor of those users who do.

2.2.2 Recommender Systems

In a RS, a user model can be used to predict what items would be the most preferential, or what actions would be most likely for a user. It can be useful to both users and service providers, making it easier to provide personalized service. The

systems have had vast uses, and have been especially popular among e-commerce as an efficient method of simultaneously increasing user satisfaction and their own revenues [26].

As the amount of content available online and within services increases drastically, many users would struggle with a information overload when trying to use online systems if RSs were not used to filter out vast amounts of information [19]. In the news domain, this involves predicting what news and topics users prefers in order to recommend relevant items, like articles.

RSs share some features with IR systems. Both systems filters information in order to provide optimal results fitting user interests, but the goal in a RS is to recommend items that the user would be interested in at any given time. This is in contrast to IR systems, which more often have a direct idea of what the user is currently interested in form of a query requesting information on a given topic. This leads to a greater demand for recall in IR systems in order to avoid missing important information on a user specified topic. RSs systems on the other hand, are often trying to recommend a few items from a vast selection and precision may be more important than attempting to recommend all relevant items, reintroducing the information overload issue. Precision in recommendations over time, can be used to gain users trust in recommendations, as well as to encourage users to try them.

The recommendations given are often based simply on information about user history and items. Most methods for recommending items to users can generally be categorized as collaborative or content-based, or hybrid systems [6] combining these methods.

Collaborative Filtering vs. Content Based

Collaborative Filtering (CF) methods model a user based on interests which are usually inferred from collected consumption, and possibly rating, history. Items are recommended based on other users with similar historical item preferences as the user. These items are usually discovered by estimating a rating for each item that has been featured in the user's nearest neighbours of users' history [27]. The fundamental assumption of collaborative methods is that users with similar item purchase or rating history, are inherently similar in respects to interests and expected future purchases. Over the years, user-item and item-item collaborative filtering has been used as a baseline for experiments, inspiring many modified versions and hybrids that improve on the traditional method and still takes advantage of its simplicity [26][27].

Content Based (CB) methods utilize information about the items to recommend items similar to those a user has preferred previously [28]. Other methods may use demographic information about users or a specified knowledge base that provides information about items and their ability to meet known user needs. Where such information is limited, log files or user click data may be used in order to detect a sequence of actions, time spent on page etc in order to more accurately model users or calculate degree of interest in an item. However, as mentioned earlier, in many cases one cannot assume access to such a detailed level of user data.

2.2.3 Prediction Evaluation

Evaluation Metric	Equation	Parameters
Precision	$P = \frac{\text{Relevant items} \cup \text{Retrieved items}}{\text{Retrieved items}}$	-
Recall	$R = \frac{\text{Relevant items} \cup \text{Retrieved items}}{\text{Total relevant items}}$	-
F1-score	$F_1 = 2 \times \frac{P \times R}{P + R}$	-
Accuracy	$A = \frac{\text{TP+TN}}{\text{Total}}$	TP = True (correct) Positives TN = True (correct) Negatives
$Precision_i$	$P_i = \frac{\text{\# of correct items among first i}}{i}$	-
$\Delta Recall_i$	$\Delta R_i = \begin{cases} \frac{1}{n} \text{ if } i\text{th item is correct,} \\ 0 \text{ otherwise} \end{cases}$	-
Average Precision	$AP@k = \sum_{i=1}^{k} P_i \cdot \Delta R_i$	-
Mean Average Precision	$MAP == \frac{1}{\lvert Q \rvert} \sum_{q \in Q} AP@k$	Q = set of all users
Percentile Ranking	$\overline{rank} = \frac{\sum_{u,i} \alpha_{u,i} \cdot rank_{u,i}}{\sum_{u,i} \alpha_{u,i}}$	αu_i = if user u interacted with article i, 0 otherwise Further described in related works (3.5.2)
Ranking Accuracy	$RA = \frac{50\% - \overline{rank}}{50\%}$	Further described in related works (3.5.2)

Table 2.2: Prediction evaluation metrics.

The standard metrics precision and recall are convenient when evaluating classification, as in prediction of words, topics, or clusters. The harmonic average of precision and recall, F1-score, describes a balance between precision and recall. These two values often become a trade-off, as the increase in Recall often can lead to diminishing Precision and vice versa. F1-Score is thus also often used for classification, and it's basis from the number of positive results requires a ground truth to be set, as with precision and recall.

In a RS it is common to generate a ranked list of recommendations ordered by relevance. In order to see how precise the recommendations are, one can calculate precision at different positions in the ranked list. MAP considers whether the relevant items tend to get ranked highly, and takes all relevant items found into consideration, in contrast to the also common Mean Reciprocal Rank (MRR), only considering the single highest-ranked relevant item.

Average percentile ranking [29] is an efficient method to evaluate ranking based on implicit feedback [30][31], searching for relevant articles in the list, and calculating in which percentiles they are positioned. The evaluation method will be further described in 3.5.2, together with ranking accuracy.

2.3 Natural Language Processing

Natural Language Processing (NLP) is the field of applying natural language for useful purposes, through manipulation and analysis of speech or text to let computers gain an understanding of human language [32]. It spans many domains such as linguistics and Machine Learning (ML), and has been gained a lot of research interests in the later years.

2.3.1 Generative Models

Generative prediction models are named so due to their ability to generate example pairs of observable data X and class Y after its training [33]. In generative models, the joint probability distribution of $P(Y, X)$ is used to calculate the conditional probability $P(X|Y)$, the probability of seeing data X when you are given class Y. It is this calculation of this joint distribution which makes it possible to generate pairs of (x, y) data pairs, through sampling [33]. This ability is absent in discriminative models, where the goal is more directly related to a traditional classification tasks, to calculate the probability of a given class label Y given the features X, $P(Y|X)$.

Generative models often use distribution priors such as Gaussian and Dirichlet, chosen by domain knowledge to appropriately fit the data in question, with variables being drawn from these [34]. This offers a flexibility that discriminative methods have lacked, though they have in turn been preferred for many NLP tasks because of the more direct link to the classification task of labeling x with a label from Y and often superior performance on classification tasks [35]. Generative models do however tend to perform better than discriminative models in settings lacking labelled data [33], making it more suitable in cases where labelled data is difficult to come by or only applied to parts of a corpus.

2.3.2 Smoothing

Smoothing is used in lanugage models and n-grams in order to avoid overfitting and an abundance of zeros when calculating the probabilities of observing words [36]. The name derives from its tendency to "smooth out" distributions by increasing the zero-values and decreasing some of the higher values, avoiding overly sparse probability values. It has been shown to improve models and can avoid making the model overly dependent on small excerpts of text [36], hence the advantage of decreasing overfitting.

Jelinek-Mercer Smoothing

Jelinek-Mercer Smoothing interpolates a maximum likelihood probability method and a collection model method, given by the following general equation [37]:

$$P_\lambda(w \mid d) = \lambda p_{ml}(w \mid d) + (1 - \lambda)p(w \mid C). \tag{2.1}$$

Here, $p(w \mid d)$, refers to the probability of the observed word w in a current document d, and λ is a parameter coefficient deciding the influence levels of the first and last term. The first term, p_{ml}, is the maximum likelihood probability of the observed word w in document d. The latter term, $p(w \mid C)$, is commonly used in smoothing methods, and aims to increase the probability of words that are currently unseen in the document by considering its presence in the corpus C [37]. This is referred to as the collection language model. Considering the word frequencies and probability of the entire collection, not just the current document, can help avoid having many zero probabilities for currently unseen words. This allows for globally frequent words gaining a higher probability than less frequent words, even if they have yet to be observed in the current document. This again avoids zero probabilities being assigned to instances not observed in the training set at the time, which may be present in the test set.

2.4 Topic Modeling

Topic modelling is a known and tested field within both ML and NLP, and is often represented as a statistical model. It allows for discovering overall themes, or topics, from unstructured data [38], such as news articles. Having a more general representation of the articles, and consequently user history, can be useful in prediction of relevant user content.

2.4.1 Latent Dirichlet Allocation

LDA is a relatively simple and popular generative probabilistic model of a corpus, often used for topic modeling. The statistic model allows for documents to describe several topics, a natural assumption for many documents. The naming originates from the Dirichlet distribution, used to describe topics per documents in the model [38]. The distribution is able to generate relatively sparse results [39], which can support the idea that each document contains relatively few main topics. Topics are represented by a distribution of words in the vocabulary, defined by the corpus. Simply put, topics are related to specific words, and a document containing many of these words are more likely to be categorized within this topic. The model depends on pre-defined topics, and the number of topics is predetermined.

Figure 2.2 shows a standard representation of the LDA model. N represents a word-topic combination, where w is the word and z is the topic distribution of the word. M denotes the document in the collection. θ is the topic distribution

[2]Source: pp.997 [40]

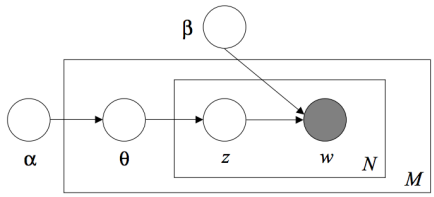

Figure 2.2: The standard model of LDA[2].

of the document, with θ_m representing the distribution for the document m. α is a Dirichlet parameter prior on the topic distribution per document, while β is a Dirichlet parameter prior on the word distribution of topics [40]. These are both corpus-level parameters, and are called priors due to representing a distribution prior to being presented with the observable data [41]. The shaded node w is the only observable factor before generation of the model [40].

The topic mixture of each document is a problem of Bayesian inference, and can be inferred calculating the following joint distribution:

$$p(\theta, \boldsymbol{z}, w \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(\boldsymbol{z}_n \mid \theta) p(w_n \mid \boldsymbol{z}_n, \beta), \tag{2.2}$$

where $p(\boldsymbol{z}_n \mid \theta)$ is θ_i for i such that $z_n^i = 1$. Integrating over θ and summing over z leaves us with the marginal distribution of a document. Using the product of the documents' marginal probabilities, we obtain the probability of the full document corups D:

$$p(D \mid \alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn} \mid \theta_d) p(\mathrm{w}_{dn} \mid \boldsymbol{z}_{dn}, \beta) \right) d\theta_d. \tag{2.3}$$

The function is intractable for exact inference (details [42][40]), but the posterior may be approximated through convexity-based variational inference, in addition to multiple other alternative inference algorithms [40].

In the dynamic news domain, the requirement of pre-defined topics is a challenge, as topics frequently might disappear or emerge. Another challenge is aligning topics as content changes over time. For instance, the term distribution of a 20 year old article about photography would most likely differ from one today where old camera technology is more or less phased out. Another example is digital currency, where credit cards and online accounts more recent years has been joined by blockchain technology. However, many improvements, extensions and adaptions of LDA has been introduced, and it appears to be a powerful and intuitive tool to incorporate as part of a topic modelling solution. Adaptions include support of topic correlations, and allowing topics to include the notion of words that are unlikely to be associated with it [38].

2.4.2 Topic Model Evaluation

The topic learning is unsupervised, and corrections and evaluation might be done by performing topic modeling on labeled corpora, comparing performance across models. However, where this is not present or topics have very coarse granularity, other metrics based on internal evaluation can be used.

Evaluation metric	**Equation**	**Parameters**
Perplexity	$\exp\left\{-\frac{\mathcal{L}(\boldsymbol{w})}{\text{count of tokens}}\right\}$	$\mathcal{L}(\boldsymbol{w})$: log-likelihood of the unseen documents $\boldsymbol{w}_d$: unseen documents
Topic Coherence	$\sum_{i<j} \log \frac{1+D(w_i,w_j)}{D(w_i)}$	$D(w_i, w_j)$: count of documents containing words w_i and w_j $D(w_i)$: count of documents containing the word w_i
Bound	$\mathbb{E}_q[\log(p(corpus))]$ - $\mathbb{E}_q[\log(q(corpus))]$	$corpus$: documents to infer variational bounds from p: prior q: posterior

Table 2.3: Topic model evaluation.

When topic modeling the aim is to extract topics aligned with human judgment, which by itself is difficult as human may disagree on semantics in a topic. The perplexity is the per-word likelihood bound for an unseen held-out corpus, where a lower score should indicate a better model. However, one should take into consideration that perplexity and human judgment are often not correlated [43]. Another metric is topic coherence which can be measured using $UMass$ measure [44], and scores how much words within a topic describes it. Since the scores are log probabilities, they are negative, and large negative values indicate words that do not co-occur often. The bound estimates the variational bound of documents from corpus and is the lower of the log probability of observations. If the approximation distribution is perfectly closed to the true posterior distribution the bound hits the log probability [45].

2.5 Deep Learning

Deep Learning (DL) is a popular sub-field of machine learning which processes data through trainable deep, weighted Neural Networks (NNs). The deep NNs open up for several layers of abstraction which can be fine-tuned through training, and have been noted for their ability to extract useful representations of data and discover structures in high-dimensional data. NNs has yielded good results on relevant tasks such as sentiment analysis, topic modeling and NLP[46].

2.5.1 Neural Networks and Training Basics

Each arrow in the fully connected NN, indicating that all nodes are connected to all nodes in the previous layer, in figure 2.3 will have a unique weight value attached to it. This weight is denoted by w_{ij} for the weight applied to information sent from node i to node j. It is these weights that are tuned when training NNs in DL. This training often aims to minimize a cost function by nudging weight values in a direction that currently lowers cost or loss, moving around in the error space. This can be done in several different ways.

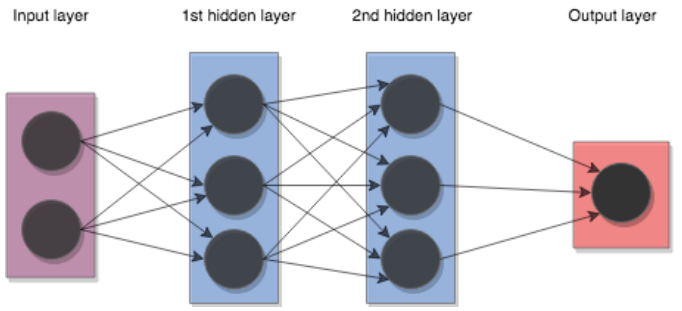

Figure 2.3: An example of a simple Nerual Network with all-to-all connections, two hidden layers and one output node.

Descent Training

For years, it has been common to use optimization algorithms which takes iterative steps descending towards the minimum of a given cost, or loss, function when training neural networks [47][48][49][50]. In each step, the algorithm tunes the weights in the neural network slightly in the direction that moves towards a minimization of the cost function value. This is usually done in batches, often called minibatches, where each weight is updated at the end of processing a set number of training cases. This makes updating the entire network for each training case unnecessary, while keeping generalization relatively high compared to running all test cases in one batch [50]. Variables in such descent training often include a step size, or a learning rate, a choice of how far to move in the chosen error-minimizing direction. The learning rate should be adapted in accordance with the loss function landscape, too small step sizes in non-smooth landscapes can lead to getting stuck in a local minima or simply very slow convergence, and too high learning rates in landscapes such as smooth hyperbolic shape can lead to oscillating between points centering around a vertex minimum. Several manners exist to determine step sizes for different methods, but no all-encompassing rules for choosing an optimal learning rate exists across networks or problems for the methods, and likely monitoring results for a given case is the best option [50]. The step size often decreases for each iteration, as you expect to start relatively far from the goal and to get closer for each step [51][50]. The choice of direction to move in, is sometimes dependant on a

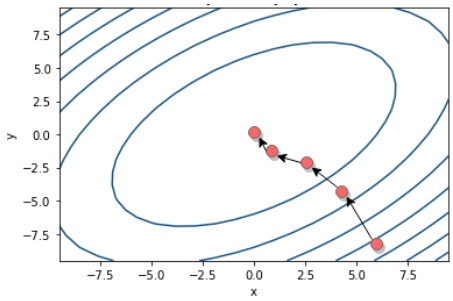

Figure 2.4: Simplified visualization of a Gradient Descent training algorithm's movement in an error space given by $f(x, y) = x^2 - xy + y^2$.

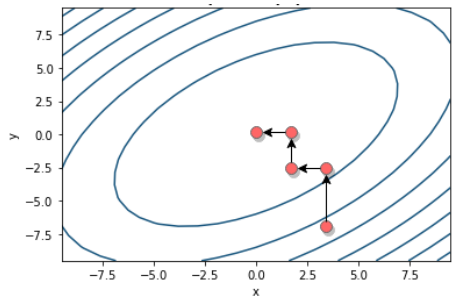

Figure 2.5: Simplified visualization of a Block Coordinate Descent training's movement in an error space given by $f(x, y) = x^2 - xy + y^2$.

derivative of the loss function in order to determine whether the cost is increasing or decreasing, meaning that not all descent training methods are a preferable option when dealing with non-differential cost functions. Non-convex problems can also be a challenge for some descent methods. A very common descent training algorithm used in training neural networks is the Gradient descent, often in the form of Stochastic Gradient Descent. As the name indicates, gradient descent moves along the steepest gradient in the error space, allowing it to change multiple variables at a time, illustrated in figure 2.4. Gradients can be slow to compute and rate of convergence is not always good, but it has been a popular training algorithm in many cases. Coordinate descent, such as Block Coordinate Descent (BCD), on the other hand, will move along one axis at a time in the error space, illustrated in figure 2.5. It will iteratively choose a single of the variables to change in order to minimize, and is built on the idea that performing the easier task of minimizing each variable individually will eventually lead to the optimal solution. It has the advantage of being applicable even in cases where the function cannot produce a derivative [52]. Still, BCD also struggle with with non-smooth cost functions due to the increased chance of getting stuck in a local minimum.

2.5.2 Representation Learning and Embeddings

Unsupervised learning has been used for the task of finding optimal representations for data for quite some time. The goal when modifying representation often vary, but generally one wants to keep as much information as possible from the original format, while representing it in a manner which is more useful to the task at hand. Common methods include reducing the dimensionality of the data, or representing it in a sparser manner to enhance likeness and differences between data entries. The latter, sparse representations, creates data representation in which most of the values are zero. Creating sparse representations is sometimes called embedding. Concepts or words can be represented as embeddings, or continuous vectors, to

model the relationships between them [53]. NLP has been proven to have great use of word embeddings learned in an unsupervised manner [54].

In word embeddings, each word starts as a one-hot vocabulary size vector before being embedded into a lower-dimensionality space. When embedding words, those that frequently share contexts or other chosen features, will be placed close to each other in the lower-dimensionality space. This means that a well-performed word embedding will provide a vector space in which small distances equals a certain semantic or contextual similarity [55]. Similarly, word pairs which share a similar semantic relationship should also share similar distances in a good word embedding, e.g. the distance between embeddings for countries and their capital cities, illustrated in a 2-dimensional space in figure 2.6. The usefulness of these embedding features has shown promise for use in fields such as translation, where one can compare each language's word embeddings to compare word use in context. They can be learned by dimensionality reduction methods such as Singular Value Decomposition (SVD), probabilistic models and neural networks [55]. [56] divides word embeddings largely into two coarse groupings of methods utilizing either low-rank Matrix Factorization (MF), often with SVD, or Neural embedding models utilizing neural networks, often with gradient batch descent optimization.

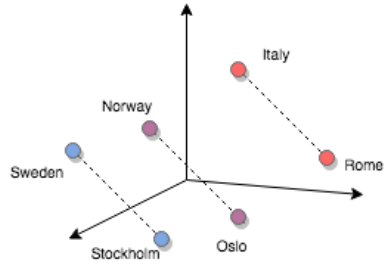

Figure 2.6: Word embedding feature of similar distances indicating similar relationships between words.

2.6 Dimensionality Reduction

Several methods described in this chapter aims at making more generalized representations, and some may be viewed as dimensionality reduction, which is the process of transforming a high dimensional data set to a data set with lesser dimensions, still capturing the original data sets information. This can be useful in regards to computational requirements, but also has the quality of extracting hidden features. It is also a valuable tool in qualtitative evaluation, enabling visualizations of high dimensional data, through techniques like t-Distributed Stochastic Neighbor Embedding (t-SNE) [57], linear discriminant analysis [58] or Principal Component Analysis (PCA) [59].

2.6.1 Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF), also called Nonnegative Matrix Approximation (NNMA), is a popular method for dimensionality reduction and data analysis, which has proven to be useful in a range of applications, like text analysis or image recognition [60], and more relevant for this paper, community detection [61][62], topic modeling [63], recommendation [64], and embedding [65]. Like the name indicates, it uses non-negativity constraints, leading to a parts-based representation where parts can be used separately to recognize items [66]. The basic idea is to derive latent features by factorizing a matrix $\boldsymbol{V}$ into (usually) two matrices $\boldsymbol{W}$ and $\boldsymbol{H}$, as displayed in figure 2.7. Here matrix $\boldsymbol{V}$ might represent purchases or interactions with items (for instance movies), and $\boldsymbol{W}$ and $\boldsymbol{H}$ represents latent features (for instance genres) for users and items, used to approximately reconstruct $\boldsymbol{V}$.

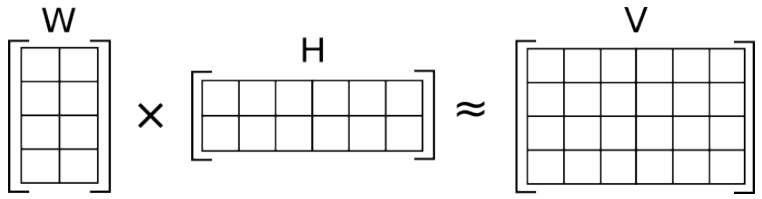

Figure 2.7: Non-negative matrix factorization[3].

[3] https://en.wikipedia.org/wiki/Non-negative_matrix_factorization#/media/File:NMF.png

Chapter 3

Related Works

The following chapter documents a literature review of related works and comments how these are relevant for this research.

3.1 Topic Clustering

Community detection as a tool for personalized recommendations has gained popularity in research the last five years. It has proven to be an efficient method when using interactions between users and rather permanent items such as books [67] or movies [68][69][70]. However, these methods deal with relatively stable item sets, making it less sustainable in the news domain.

An alternative method should therefore be able to quickly predict article relevance for a user without relying on user interactions. Sure the articles can, through manual labour, be targeted to communities, but manual labour is a demanding task susceptible to reduced objectivity and subjectivity. At the same time, community detection has potential to help deal with the sparsity of data [71], and some suggest that overlapping communities that allow for a user to be members of several community may assist in providing diversity in recommendations due to a larger relevant item space to choose from. This is also seemingly realistic way to represent users, as people's interests are nuanced and most people in reality do not exclusively belong to only one community [72].

A way to deal with this problem could be by clustering the articles by semantic topics. Li et al. [73] propose a online blog reading system providing personal ranked content based on topic clustering, resulting in generally satisfied users in a user study. However, one should note that the clustering methods are based on link structure among items (articles, blog posts etc.), where traditional news articles and blog posts might differentiate.

In [74] Wu et al. compares topic models by topic clustering Chinese web news. For a topic model of K topics each document d_i is represented by a K-dimensional topic distribution vector $\vec{V}(d_i)$. Topic distance is then defined as the distance between two document d_i and d_j measured by their topic vectors $\vec{V}(d_i)$ and $\vec{V}(d_j)$,

shown in equation 3.2. These distances are then used in topic clustering (by K-center clustering) on web page collections with different number of topics and articles, showing that topic clustering using topic distance benefits from the topic features captured by more complex topic models, such as LDA described in 2.4.1, compared to simpler models further described in paper.

$$D(d_i, d_j) = \sum_{k=1}^{K} P(\theta_k|d_i) \ln(\frac{\theta_k|d_i}{\theta_k|d_j}). \tag{3.1}$$

$$\vec{V}(d_i) = (P(\theta_0|d_i), ..., P(\theta_k|d_i)). \tag{3.2}$$

3.2 Topic Embedding

Recent efforts have seen many important improvements in word embeding methods, such as the model word2vec enabling more efficient training on larger corpora, and a multitude of other methods employing for instance regularization or bilinear regression for better performance [56]. Over the last few years, several efforts have been made to extend the theory of word embeddings, outlined in section 2.5, into useful representations of longer texts. This could provide opportunities for representing documents as fixed-length vectors derived from content and context, a useful format for many ML, recommendation, and NLP cases. Most testing has been focused on information retrieval, text classification and sentiment analysis tasks, not performance in applications such as recommender systems. In [75], topic embeddings are used to cluster news text with promising results, but this is never extended to use for prediction or recommendations. While several of the models attempt to take advantage of the additional semantic information contained in trained word embeddings when extracting topics [76], others generate both at the same time [77] and some do not make use of word embeddings [78].

Paragraph Vector is one of the earliest, and maybe the most known, document embedding methods. Authored by the researchers behind the earlier mentioned word embedding method word2vec, [79] extends work from [80][81] to generate vector representations for text of variable length. Their goal descriptive is to predict words within a paragraph, allowing semantic features to be captured and embedded into the randomly initiated vectors as training progresses. A combination of Efficient Correlated Topic Modeling, an extention of LDA, and topic embedding is created in [78], gaining solid results on classification and information retrieval tasks. It's most promising feature, is scalability.

Comparable results to these are achieved in [82] when testing on some of the same datasets. Here, Li et al. presents the topic embedding method TopicVec. It is extended from PSDVec, a word embedding built to be scalable through exploitation of sparseness, as well as to avoid corpus information loss which can be caused by combinations of MF and SVD by capturing non-linear interaction between words [56]. PSDVec is a generative model made with facilitation of later learning of latent factors, such as topics, in mind. Simply put, documents in TopicVec are presented as "bags-of-vectors" in which the elements refer to embeddings of the

words contained in the document [82]. In a document, groups of related words are expected to show up which will have shorter distances to each other in the embedding space than to other groups of words. These groups have been named semantic clusters. Each semantic cluster has a semantic centroid, which is considered to be the most representative entity for the semantic cluster. These centroids, along with surrounding words, can be used to represent the document. Due to the setup and qualities of embeddings, words surrounding the centroid should have a semantic relation to the same topics even if it was not specifically mentioned in the document.

The semantic centroids produced by this generative method is compared in representation and practical tendencies to the topics produced by LDA, as both models uses Dirichlet priors to represent the distribution relationship between words, documents and topics. This means that each topic has few high-probability words related to them, and that each document is expected to contain a few high-probability topics. Several other attempts at combining word embeddings with LDA like methods have been made earlier, but [82] notes them as too slow to function on larger corpora [83], potentially choosing improper distance relations between topics and words [84] or lacking statistical foundations and requiring a large corpus to create good topics [85]. An advantage of TopicVec is its expressive output, with topic vectors describing word distribution for topics, document vectors describing topic distributions for documents, as well as influential words related to the topics.

[77] notes the disadvantage and potential risks of pre-training word embeddings used for topic models on external data sources such as Wikipedia. The potential lack of appropriate data and semantically different tendencies are the main concerns. While this is a valid concern in many fields, commonly available large corpora such as Wikipedia should share many common features with the news field and be appropriate for pre-training in this context. Commonalities include similar language and diverse coverage within many different topics and events. Wikipedia articles also vary greatly in size, such as news articles. In [77], Xun et al. questions the choice of the many heuristic methods like PSDVec, in which word and topic embeddings are trained separately, because it can lead to loss of information learned by taking advantage of both local and global contexts during training.

3.2.1 Embeddings in Recommendation Systems

Some work has been done to incorporate embeddings in recommender systems, but domain focus appears to fall outside the parameters of direct relevance to the news domain.

[86] takes inspiration from NLP and uses word2vec, but creates embeddings on venue check-ins instead of textual information in order to recommend new venues to users. Vasile et al. creates embeddings of products based on sequences of purchases based on a model named Prod2Vec, while incorporating available textual meta data such as tags are used on a dataset music playlist and listening history, in [87].

In [88], textual information is collected from Wikipedia articles related to the items and used to create word embeddings by Word2Vec based on the gained corpus. Then items are presented as vectors based on their related words, and further

used to create user models from item purchase history. While this method has the most potential of relevance in the research setting of this thesis, a difference is that the processed texts are the items to be recommended and will not be augmented by external datasources.

None of the methods appear to have stated a strategy for addressing new topics or updating the embedding vocabulary without rerunning a process or continuing iterating through a training process which will modify several parts of the embeddings. Concerns for scalability are mainly focused on exploiting sparsity or other data features in order to improve the efficiency of the training process. Many methods use pretrained word embeddings, provided through own work or publicly available libraries such as Word2Vec[1]. These pretrained word embeddings may be of high quality, but could become suboptimal over time. With the number of new items being added being a very frequent happening in the news field [89], adding new words, which can be associated with topics, could be an important feature for maintaining model quality and alignment across time. The importance of avoiding additional training algorithm iterations become apparent when considering the consequential change of interpretation of the topics built on top of the word embeddings berfore updating. While some methods allow for adding of new word embeddings at a later time, the calculation of these are not explicitly mentioned, nor the potential attachment to topic embeddings.

3.2.2 Updating Word Embedding Vocabulary

The news domain faces challenges with respect to it's dynamicity and rapid change of relevant content, which will inevitably impact topic embeddings in terms use of word embeddings and vocabulary. Within news, the vocabulary can change as new terms and expressions emerges, and already embedded terms might change semantic meaning over time. Incorporating new terms frequently used over a longer time period could be important to include relevant content for predictions, but the dimensionality of the embedding space and demand for inference may be too large for the potential semantic changes in the terms being worth constant costly updates, despite being low-dimensional compared to traditional representations.

Though it does not appear to have been a prominent subject of interest within the field of recommender systems with content information encoded in embeddings, incremental learning has been of interest within the domain of dimensionality reduction and embeddings. The topic aims to avoid the computationally demanding task of rerunning a full algorithm when presented with new data. This has often been achieved by adapting existing models in order to handle incremental changes, rather than using more rigid data inputs in the form of data batches [90] or simply feeding all available data as input [91]. Important considerations in [90] and [91] include handling skewed and non-uniform data in order to adapt to smaller and less predictable data input, or incrementally updating all affected embeddings with the impact of the new data or removal of old data. A natural disadvantage with these methods, is their focus on incremental training, while a common use

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

case will be post-training modifications. While the models could merely run an extra training iteration, this would leave the previous embedding basis vulnerable to changes. In the case of word embeddings generated to operate as a basis for topic embeddings, this may cause alignment issues or require heavy re-computations of word embeddings which propagates into the topic embeddings, clusterings and user models.

3.3 User Models in Recommendations

The concept of modeling users was introduced decades ago, then named stereotyping [92], and has even been described as one of the earliest tools in user modeling [93][70]. However, these early user models were often more rigid and pre-modeled with attributes manually attached [93][70], generating the need for a lot of manual labour as well as risks for subjectivity and missing connections between user patterns. Consequently, more automated methods of using user feedback for user modeling has been used in later time [94][95], and is also the focus in this report, being a more realistic use case for current systems.

Both CF and CB, briefly introduced in 2.2.2, has proven to be useful techniques for representing users with recommendation purposes. As mentioned, the techniques has inspired many modified versions and hybrids, combining elements from both [6]. However, in a domain where both users and items are rapidly changing, like in the news domain, there may not exist sufficient history about items, or articles, in order to give good recommendations before the item is outdated. Nevertheless, a purely CB system may get stuck with repeatedly recommending similar items, and hence lack the ability for exploration. Building an underlying network structure, like described in section 2.1 could be a possible solution, but lack of explicit connections in relevant data sets often enforces extraction of connections based on available data, like article content. This is challenging due to implication of users within a cluster preferring similar content, while interest profiles may be rather complex composite.

In [96], the content of scientific papers are used to recommend papers to users. The method improves on the traditional method of having keywords representing content. The use of topic modeling of content appears to have become a popular method to enrich the amount and quality of content information before performing CF, often successfully [97][98]. This makes sense, as the amount of data to analyze is increasing and some data sets may have non-sufficient description of items which may be costly or inefficient to expand on manually.

3.4 User and Content Dynamicity

The challenge of data analysis in a dynamic domain with drifting concepts and preferences has been solved in different ways. In [99], three ways to address the representation of concept drifts are presented. Instance selection generally only consider the instances within a given time frame relevant, and often attribute the

same importance to all the instances within that time period. This method will generally risk providing wrong insight in cases where the concept drifts are changing over longer periods of time rather than abruptly within definable time periods, making it less relevant for the perspective of news recommendation. There is also a risk of increasing the issues of sparsity when using this method [100].

Employment of instance weighting is a technique that can be used to mitigate some of these problems. A weighting, usually given by a formula defining a decay rate, can discriminate instances based on their recency without having to have a set cutoff of which instances to consider. The last approach described in [99], is ensemble learning. Here, one keeps track of the prediction indicators which contribute to the final prediction made, weighting them by an assessment of relevance of the indication to the current time.

However, [101] notes that using these methods to merely disfavor older instances is not sufficient. Through experimentation they found that having low or no decay rate provided better prediction abilities at times, attributing this to influences such as persistent user interests which exists in addition to those interests which change over time or is merely present for a short amount of time. They did still find the tendency for user interest, as well as their rating scale, to change with time. They argue that the findings could be rooted in the importance of the persisting user interests when making predictions across users or items. This emphasizes the necessity for finding a model which correctly interprets variations and new user interests, while still maintaining the persistent user interests. If a user has shown a continuous interests in a topic for several months, it is important not to choose a decay rate which may underestimate the relevance of this topic merely because the user shows interests in many other topics in a short, more recent time span.

3.5 Sparsity in Recommender Systems

This section will present methods dealing with some of the main challenges brought on by the news domain's volatile nature leading to sparse user-item matrices.

3.5.1 Item Cold-Start Recommendation

As mentioned, one of the main challenges in RS is making meaningful predictions when a new user or item is introduced to the system, due to the cold start problem, being particularly relevant in the news domain. One attempt of dealing with this problem is the Saveski et al. proposal of a hybrid approach, Local Collective Embedding (LCE) [30], combining collaborative and content information, exploiting both the properties of the items and the similarity of the user preferences using matrix factorization. They define these matrices as follows: content matrix $\boldsymbol{X_s} \in \mathbb{R}^{n \times m}$ describes n items by representing each item as a row with m properties, and collaborative matrix $\boldsymbol{X_u} \in \mathbb{R}^{n \times u}$ describes user interests by representing a cell (i, j) as user i's interest in item j. Given a new item $\boldsymbol{q}$ with properties $\boldsymbol{q_s} \in \mathbb{R}^{1 \times m}$ and user interest $\boldsymbol{q_u} \in \mathbb{R}^{1 \times u}$, the goal is to predict $\boldsymbol{q_u}$. That is to say, predicting user-item preference $\boldsymbol{q_u}$ through document-term matrix $\boldsymbol{X_s}$

and document-user matrix $\boldsymbol{X_u}$. Both matrices are factorized to lower dimensions, where $\boldsymbol{X_s}$ discovers topics in documents and $\boldsymbol{X_u}$ discovers communities. In order to utilize this in recommendations, LCE collectively factorize $\boldsymbol{X_s}$ and $\boldsymbol{X_u}$ and enforce a low-dimensional representation in a common space so that each factor can both be described by a topic and a community. Given a new item, LCE may project it in this common space and infer likely interested user communities. The model is evaluated using data from the news domain and has proven to outperform several state-of-the-art methods for item cold-start.

3.5.2 Implicit Feedback and Evaluation

As mentioned in section 2.2 the news domain suffers from lack of explicit feedback from users for different reasons, making implicit feedback the foundation for prediction. It is important to note that in contrast to explicit feedback, where items may be rated both positively or negatively, implicit feedback will not reflect feedback regarding undesired articles. While interaction with an article is an evidence of the user's interest in it, the absence of such is not an indication of disfavour, as this may stem from multiple different reasons. Due to this and lack of ability to receive user feedback on generated recommendations, precision based metrics are not suitable, but rather recall based metrics.

Average percentile ranking [29] is an efficient method to evaluate ranking based on implicit feedback [30][31]. Defining $rank_{u,i}$ as the percentile ranking of article i in the ordered list of recommendations for user u, $rank_{u,i} = 0\%$ if article i is predicted to be the most interesting article for user u, while if $rank_{u,i} = 100\%$ the article is predicted to be least preferred. For an article the total average percentile ranking, $\overline{rank}$, is then defined:

$$\overline{rank} = \frac{\sum_{u,i} \alpha_{u,i} \cdot rank_{u,i}}{\sum_{u,i} \alpha_{u,i}} \tag{3.3}$$

$$\alpha_{u,i} = \begin{cases} 1 & \text{if user } u \text{ interacted with article } i \\ 0 & \text{otherwise} \end{cases}. \tag{3.4}$$

, where lower $\overline{rank}$ values indicate correct predictions closer to the top of the ranked list. By calculating $\overline{rank}$ for each article, we can use mean average percentile ranking to evaluate a model.

For random predictions, the expected value of rank is 50%, and thus a model resulting in rank $> 50\%$ is no better than random. To ease illustration, Saveski et al. [30] introduces RA, calculated by converting the average percentile ranking as followed:

$$RankingAccuracy = \frac{50\% - \overline{rank}}{50\%}. \tag{3.5}$$

RA thus is optimal at value 1 when percentile ranking is 0%.

Chapter 4

Methodology

The following chapter will describe the methodology, including the data sets, as well as introducing our experiments. Experiment set up will be detailed in Chapter 5, while Chapter 6 presents and discuss the results.

4.1 Implementation

This section provides a brief, high-level description of our implementation and it's components.

4.1.1 Architecture

We aim to improve recommendations of news articles by performing predictions which mainly relies on content, in order to deal with the news domains' challenges regarding sparsity and volatility. The process utilizes topic extraction and clustering methods to build user models and embed new articles, and incorporates ideas from related works (chapter 3). Figure 4.1 shows the implementation at a conceptual level (components described below), and can be viewed as a pipeline transforming unseen articles to user recommendations, aided by a corpora of known articles and user interaction links. The filtering and cleaning of the corpora is left out of the pipeline, but methods used can be found in section 5.1.2.

The models in the pipeline are trained on the known article base and it's connections to users. When unseen articles are presented to the system, they are processed by these models and assigned to clusters, which are used when ranking each article's relevance for each user. The subset of articles considered relevant for prediction could be defined as a times slot, like articles published within the last 48 hours, or by window size, like the 800 most recent articles. It could also be decided by behaviour, such as increase in interest, specific tags, or another relevance function.

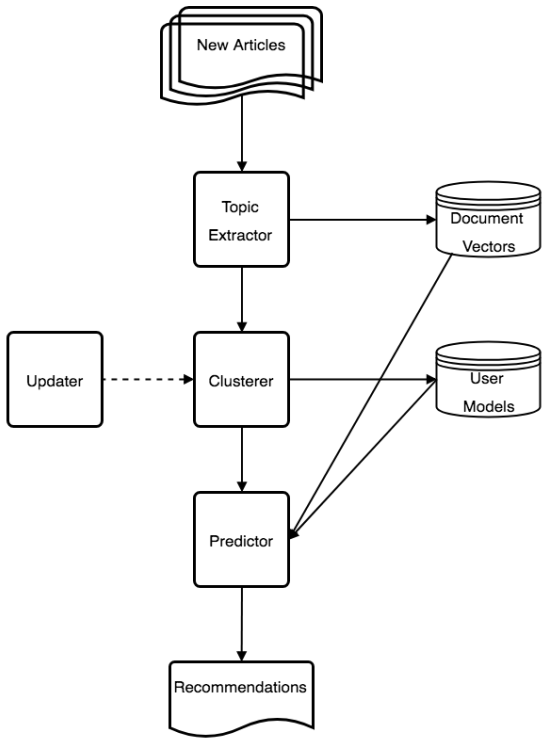

Figure 4.1: Basic architecture for article recommendation.

4.1.2 Architecture Components

Each of the components in the architecture illustrated in figure 4.1 will now be accounted for in turn.

Topic Extractor

After preprocessing, semantic analysis is used to construct models which can represent documents as proportions of abstract topics. The topic modeling step generates document labels which are more descriptive and less restrictive than labeled categories. Categories are often very high-level and overarching, and even if multiple categories are employed for a single document, connections and nuances may be lost. By employing topic modeling, the document topic labeling can be made automatic and non-subjective, with flexible granularity. This allows for discovering more complex topics, as well more composite topic document representations. The topic models will generate t topics that can be used to infer document topic proportions for an unseen document, which is consequently represented as a t-dimensional

vector describing its topical content. It is these document vectors that are used to represent the document in the system.

Clusterer

The clustering step augments the documents' topical content by associating them with the patterns of a group of similar documents. Inputting the document vectors from the topic extractor, the clusterer allocates each document to the cluster which best fits its topical content.

Due to the clustering model being fixed when the system is initialized, an updater which allows for re-initialization of clusters is added to the architecture. An updating process could make use of stored document vectors to train a new clustering, which would require updating user models accordingly. This can protect the system from performance decline caused by situations such as one cluster becoming overly dominant over time, or the originally trained model not being equipped to handle new behavioural patterns.

User Modeler

In the user modeler, each user is appointed a representation based on their historical association with the topical clusters outputtet by the clusterer, each assigned with a membership strength. When new documents are read, changes will naturally propagate through user models.

The component can easily be varied in complexity, by adding levels to the user modeling. This can include normalizing user activity to accommodate for varying behavioural patterns, or using decay rates on cluster memberships to capture interest drifts.

Predictor

The predictor combines user models and community strengths with features and cluster assignments of documents, in order to provide a ranked list of recommendations to users. This component also allows for some flexibility, in terms of complexity. Examples include the choice of treating all documents in a cluster equally, supporting exploration, or assigning documents with a cluster membership similar to with user models.

4.2 Experiment Methodology

In the following section, methods used to execute the architecture are detailed. The first two subsections presents the construction of user models through topic extraction and clustering, using LDA and TopicVec respectively. This is followed by a description of how user predictions are performed, and the evaluation of recommendations.

4.2.1 User Modeling with LDA and Clustering

Generation of user models with LDA topic modeling and clustering is done in three main steps. At first the most promising topic models generated by LDA are selected, followed by representing documents as topic vectors utilizing these models. The document vectors are then clustered, forming topical communities, before users are connected to these communities based on their interaction history.

Before running LDA to extract the topics, a dictionary is generated from the vocabulary. The dictionary contains tokens ids for each word and is used to convert the preprocessed document vectors to multiple Bag-of-Words (BoW) consisting of token ids and token counts. The BoWs are then used to train LDA models, as described in 2.4.1, for a range of predefined numbers of topics. A topic modeling with t topics and n documents, yields a $t \times n$ document topic matrix later is used to derive topical clusters.

The clustering is done based on topic distances between the documents in the document topic matrix, as defined by Wu et al. [74], explained in 3.1. Based on the calculated distances, clustering of documents is done to detect topical communities. Multiple distance based clustering methods are applicable for this task, but due to the news domain's dynamic and composite nature, it is preferable that the method is scalable and able to handle irregular shapes and sizes of clusters. Being a fast, general-purpose algorithm able to handle large portions of data, the k-Means is one of the selected algorithms. However, it does have a few drawbacks, and do not satisfy all convenient features, as described in 2.1. A complementing algorithm, tackling some of these shortages, is the DBSCAN algorithm which neither requires a predefined number of clusters or enforces convex, evenly sized clusters. The other algorithm selected, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) developed by Campello et al. [102], is an extension of DBSCAN converting the traditional algorithm into a hierarchical algorithm.

Like the regular DBSCAN, the hierarchical version also starts by finding the identifying core points and neighbours within a given radius. It produces a clustering tree containing all DBSCAN partitions in a hierarchy, and then extracts a flat clustering by maximizing the overall stability of the set of extracted clusters. This can be viewed as the following optimization problem:

$$\max_{\delta_2,\ldots,\delta_\kappa} J = \sum_{i=2}^{\kappa} \delta_i S(\boldsymbol{C}_i) \tag{4.1}$$

$$\text{subject to} \begin{cases} \delta_i \in \{0,1\}, i = 2,\ldots,\kappa \\ \sum_{j \in \boldsymbol{I}_h} \delta_j = 1, \forall h \in \boldsymbol{L} \end{cases} \tag{4.2}$$

where $S(\boldsymbol{C}_i)$ denotes the stability value of each cluster $\boldsymbol{C}_i$, except root $\boldsymbol{C}_1$, and where

$$\delta_i \begin{cases} 1 & \text{if } \boldsymbol{C}_i \text{ is included in the flat solution} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

for each cluster. $\boldsymbol{L} = \{h \mid \boldsymbol{C}_h \text{ is a leaf cluster } \}$ is the set of leaf cluster indices, and $\boldsymbol{I}_h = \{j \mid j \neq 1 \text{ and } \boldsymbol{C}_j \text{ is ascendant of } \boldsymbol{C}_h (h \text{ included})\}$ is the set of indices of all clusters on the path from $\boldsymbol{C}_h$ to $\boldsymbol{C}_1$ (root). The constraints prevent selection of nested clusters on the same path. The full, detailed process of the optimization can be viewed in [102].

After performing k-Means and HDBSCAN clustering, the best performing models are picked out based on Silhouette coefficient and C-H index (section 2.1), as well as a qualitative evaluation in combination with these metrics. For a qualitative evaluation and understanding, the clusterings are dimensionality reduced and visualized in two and three dimensions, using Linear Discriminant Analysis, t-SNE and PCA. These are then labeled with both cluster id's and categories (and for the reddit documents also subreddits) to see the overlap between them and potentially different levels of granularity.

The best clusterings are used to create user models from a set of test users. For each test user, a group of documents the user has interacted with is transformed to topic vectors with selected topic models, and then assigned to clusters from selected cluster models. Each user is then assigned topics and weights decided by how many times the user has interacted in the cluster, and the general behaviour pattern of the user and the cluster the document belongs to, as more closely described in the user modeler component in previous section (4.1.2).

4.2.2 User Modeling with Topic Embedding

The TopicVec method using PSDVec for the base word embeddings, mentioned in 3.2 and detailed in [56] and [82], is used for generating the topic embeddings. The main motivations behind this choice is the relatively high dimension of the word embeddings, the expressive results of the topic embedding through topic vectors as well as topic proportion vectors for each document, and the access to code made available by the authors themselves[1]. Due to the domain similarity between texts in Wikipedia and news mentioned earlier, the concerns voiced in [77] regarding pre-training on a different dataset than the topics are trained in are not deemed to be crucial. While [77] also criticizes the heuristic principle PSDVec is built on, the ability to separately control, and update, the word embedding is preferable in this case. Especially, the ability to add words and keep alignment of topics in [77] is unsure and more difficult to test than with PSDVec and TopicVec.

In the following sections, the process of word and topic embeddings will be detailed, as well as the updating scheme for word embeddings.

Word Embeddings

The word embeddings are generated using the method described in [56]. Mentioned in Section 3.2, PSDVec is a generative word embedding method trained through BCD, explained in 2.5.1, designed to accommodate for later use of latent factor utilization such as topic detection, discussed further in the next section. The linking function between two words in the resulting word embedding method, PDSVec,

[1] https://github.com/askerlee/topicvec

considers the linear correlation of the words, a residual nonlinear correlation and unigram priors. In the case where s_i and s_j are independent, $P(s_i, s_j) = P(s_i)P(s_j)$, but with any correlation, $P(s_i, s_j)$ is given by a scaled factor from:

$$P(s_i, s_j) = exp\{\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j}\} P(s_i)P(s_j) \tag{4.4}$$

which denotes positive or negative correlation. Here, $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_j}$ captures the linear correlation between between s_i and s_j. $a_{s_i s_j}$ is the bigram residual, in some literature named a bias term, capturing the noisy interaction between the bigram words.

By the model in [56], we apply Gaussian priors and Jelinek-Mercer Smoothing, introduced in 2.3.2, in order to reduce tendencies of overfitting. This can be important in this use case, where the word embeddings will be used for texts within all kinds of different topics, and potentially not on the same type of dataset that the original word embeddings were generated with. Instead of smoothing the probability of reading a word in a current document by a factor of the word's global presence, we smooth the probability of reading a word in the current context of another word by its global probability. This avoids many zero-probability bigrams, which may skew the model too heavily to the training data.

After this, the learning objective of the PSDVec method is maximizing the likelihood of the corpus given by $p(D, V, A)$, where the probability of a specific document is expressed in the generative manner described in 2.3.1 as:

$$p(d_i|\boldsymbol{V}, \boldsymbol{A}) \tag{4.5}$$

calculated by approximating information about the probabilities of co-occurrence information, using a variant of a BCD training algorithm. In order to deal with the scalability issues brought on by this choice, [56] works with a blockwise regression algorithm that can use the sparsity of the weight matrices to its advantage.

The results of their work is 180 000 continuous word embeddings in a 500-dimensional space which is competitive in most similarity task tests and used in further work detailed below. The researchers in [56] have also provided these results as a pretrained word embedding, which is used in their further work in [82] and this project. The decision to use this smaller vocabulary, instead of for instance Google's 3 million word word2vec trained vocabulary with 300 dimensional vectors, is due to the embedding update scheme. It is unreasonable to assume that a word embedding trained 2 years ago will contain all news-relevant words, but if using a 3 million word embedding vocabulary as a basis, the testing of addition of new embeddings would be more difficult due to it being a rarer occurrence. The advantage of the increased amount of dimensions and smaller size of embedding vocabulary compared is the added opportunity and compatibility of testing for the embedding updating scheme.

Topic Embeddings

The topic embeddings are generated by the TopicVec method summarized in 3.2. As in LDA, due to the similar expectations of word-to-topic and topic-to-document

relations, Dirichlet priors are used for regularizing the topic distribution. A learning process trained by gradient descent is used to derive embeddings for topics, which approximates underlying semantic centroids. The topics are created in the same n-dimensional space as the word embeddings. These topics can then be used to infer the topic distribution of new documents, whose topic proportion matrix will be of size t, where t is the number of topics discovered in the corpus.

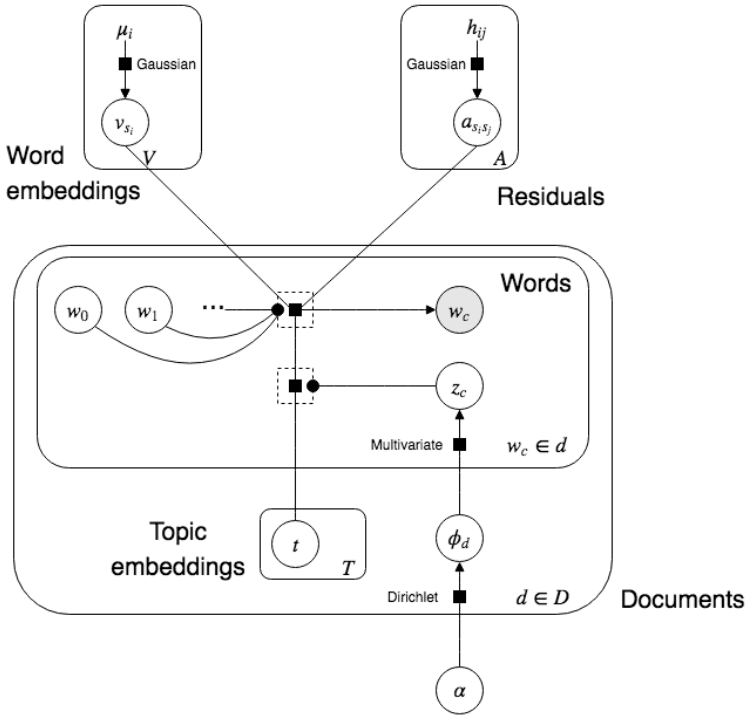

Figure 4.2: An illustration of the generative TopicVec model from [82].

A visual representation of the generative model is given in figure 4.2. The model has more components than LDA, which are summarized in table A.1, explained in the following and detailed in [56] and [82].

The shaded node of w_c is the only observable variable, and the rest are being drawn from various distributions and trained to maximize a goal. The upper containers V and A, along with the context words, represents the embeddings generated by PSDVec discussed in the last section, which is done before topic training. In our case, this work is pretrained and will not need to be run for each topic embedding instance.

The container $\boldsymbol{T}$ denotes the topics in document d, where each instance of t_k represents the kth candidate topic embeddings in the document. In the lower right corner, the Dirichlet prior is applied to the inference approximation distribution for document proportions, θ_d, used to approximate the topic proportions within a

document, ϕ_d. Each word w_c has a topic assignment z_c.

Given these variables drawn from varying distributions, training becomes a matter of maximizing the data likelihood of the corpus, similar to the goal in PSDVec but with more variables. The likelihood of a single document d_i is given by:

$$p(d_i, \mathbf{Z}_i, \boldsymbol{\phi}_i | \boldsymbol{\alpha}, \mathbf{V}, \mathbf{A}, \mathbf{T}_i) = p(\boldsymbol{\phi}|\boldsymbol{\alpha})p(\mathbf{Z}_i|\boldsymbol{\phi})p(d_i|\mathbf{V}, \mathbf{A}, \mathbf{T}_i, \mathbf{Z}_i). \tag{4.6}$$

with the variables seen in fogure 4.2. This is also a generative step, calculating the probability of seen documents, along with label proportions for documents and words, given the word embeddings and topics. For the full corpus, the likelihood becomes:

$$p(\mathbf{D}, \mathbf{A}, \mathbf{V}, \mathbf{Z}, \mathbf{T}, \boldsymbol{\phi} | \boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\mu}). \tag{4.7}$$

The training objective is to discover the word-topic and document-topic distributions, $\mathbf{Z}$ and $\boldsymbol{\phi}$, given the hyperparameters. As the method is heuristically split into two steps, the variables $\boldsymbol{V}$ and $\boldsymbol{A}$ from Equation 4.7 are fixed by generating optimal word embedding by PSDVec. This step is pre-run in our case. The second stage finds the optimal topics from the full corpus, thereby fixing the variable $\boldsymbol{T}$. What is left, is then:

$$p(\mathbf{Z}, \boldsymbol{\phi} | \boldsymbol{\alpha}, \mathbf{D}, \mathbf{A}, \mathbf{V}, \mathbf{T}). \tag{4.8}$$

This inference of document and word distributions to topics is approximated, as in many other instances includingLDA, due to the infeasibility of calculating them precisely efficiently.

Interesting output is $\boldsymbol{T}$ and $\boldsymbol{\phi}$, topics described in 500-dimensional space from the distribution of words and documents described in n-dimensional space describing the proportion within the document of the n topics discovered. It is these n-dimensional vectors describing topical content in documents which is used when clustering and prediction. Using these document representations, clustering is performed in the same manner as described in section 5.2.1.

Updating Underlying Word Embeddings

The news recommendation field has the advantage of both generating large amounts of textual data in itself, as well as often addressing the same events and topics in a similar manner to massive amounts of data available in text corpora such as Wikipedia. This provides opportunities for training solid word embeddings with a large training corpus, but the word embeddings will naturally be dependant on what is available in e.g. news corpora at the time of training. The nature of the news field, where thousands of articles are published within a multitude of topics each day and new events occur and terms are coined relatively often, is not necessarily compatible with a static word embedding vocabulary based on data from a year ago, even if it is based on a very large text base. At the same time, the news field produces articles featuring information on most conceivable topics, and

having a large word embedding vocabulary could prove important to facilitate finding topics across many domains and subjects. The need for a relatively large and diverse training corpora, to extract topical tendencies across most domains, further increases the futility of a RS which needs to recalculate word embeddings, topic embeddings and user models on an expanded corpus each time a new word of interest is incorporated. Additionally, when topics are built on top of this embedding vocabulary, they are dependent on the underlying word embedding vocabulary and its consistency. Thus we are faced with the challenge of preserving the meaning of topic embeddings through consistent word embeddings while adding new words to the word embedding vocabulary in order to better capture topical structures in the future.

As mentioned in Section 3.2.2, several efforts have been made within the field of incrementally updating word embeddings as they are trained. Though these attempts have been successful at times, the specifications of the methods does not quite meet the requirement of updating word embeddings which are intended serve as a base for topic embeddings. Most methods are created to allow for an incremental training, which is mostly useful before the word embedding is taken into use for topic embeddings in our case, as well as often adapting the entire model closer to optimalization rather that just modify the one word embedding.

The effect of adapting the methods such as [90] and [91] to update embeddings based on new words after initial training, would result in continuous changes to parts of the word embeddings. This would inherently affect any topics represented partly by any of the changed embeddings, and would require the potentially computationally heavy task of updating all affected topic embeddings, consequently including all user models based on the affected topics. The feasibility and outcome of such a process is unknown and not within the scope of this thesis, but appears to present the same challenges as merely rerunning the entire word embedding and topic extraction in terms of computational work and need for updates that will propagate across existing word embeddings, topic embeddings and user models.

In addition to this, the deletion case taken into consideration in [90] is less important in the news recommendation field. One reason for not considering removal of word embeddings after their creation, is that terms or topics declining in interest is a less pressing matter. Unused words will likely not harm the quality of a document embedding, while missing a word might result in a less descriptive embedding. Words which are rarely or never used, may also just be a temporarily uninteresting in the news domain before a topic of interest flares up again. In addition to this, removing terms from the embedding vocabulary runs the potential risk of changing the meanings of older document topic descriptions the same way that modifying the word embeddings of words used to represent the same document would. If several key words are removed from the word embedding of which a document representation was based on, a new issue of alignment is presented. Thus we want a word embedding vocabulary in which existing words retain their historical positions, while news words are simply added based on the word embeddings of their context words. By adding new interesting words in place of rerunning the entire generation process, we also facilitate the original embeddings

maintaining their position in the n-dimensional space. This allows for topics generated, and user interests based on the word embeddings, to maintain their value and meaning across time and adding of new words, making them remain valuable for recommendations.

Noting the lack of existing word embedding models that allows for sporadic addition to the vocabulary after training and is suited to support alignment of topic embeddings across word embedding updates, we have chosen to implement our own embedding vocabulary update scheme. This is a rudimentary method employed in the short-term context of this research to highlight and take into account the concern of news field vocabulary changes over time.

Having noted earlier in this thesis that some methods pertaining to word embeddings often use centroids as representative entities, we will test an extension of the principle which the original word embeddings are based on, that words are in their meaning defined by their context , to where a word is described by the meaning of its context words' embeddings. This essentially means giving a new word an embedding value which in some manner averages the embeddings of its context words, weighted by the number of co-occurrences or another appropriate metric. It stands to reason that two semantically similar words with many identical or similar context words will still be assigned similar embeddings by this method. It is important to note that other word embedding qualities, such as the distance between words representing similar relationships, may suffer in this process. We make a crude assumption of these word embedding attributes being more important in sentiment analysis and translation use cases than here.

The ability to add new words into the embedding space without updating or modifying the original embeddings, is inherently dependant on the fact that embeddings in an embedding space are so far apart that adding an element will not excessively disrupt its neighbourhood. In a binary 500-dimensional space, 2^{500}, approximately 3.27×10^{150}, potential embeddings exist. Taking into account the continuous nature of the embeddings used here, this number would be substantially higher and the distances between word embeddings is expected to generally be fairly large. If a new word embedding is based on sufficiently many different words, it should thus not be placed so close to another word in the embedding space that it will generate too much confusion or noise in terms of interpretation. Another important factor is that new word embeddings should share similar statistical tendencies to the original word embeddings, in order to avoid shifting or running an unnatural interference with any topical inferences done using the embeddings. Considering the purpose of the embeddings, describing topic distribution of an article which presumably consists of a fair amount of words, one word should within these limits not inhibit the process. The use of the topic proportions to discover similarities between continuous vector representations of documents should also dampen the consequence of one misplaces word in the embedding space, as it may be more fault tolerant than for instance a binary classification tasks.

The conditions and parameters for adding new word embeddings have to be considered and tweaked. Parameters will at the bare minimum include a threshold of the number of word occurrences to be considered significant enough to be

embedded, and a window size deciding the radius of surrounding words affecting the new word embedding. We do not want the embeddings to be overly dependant on a few words, especially in this context research where we are looking at a smaller amount of documents and time frame, so all relevant context words will contribute to the final embedding. An alternative is choosing some top N words of co-occurrence after collecting word mentions in documents, or having a similarity or count threshold as in the original training of PSDVec. Options for the occurrence threshold parameter can include being mentioned in a specified amount of articles and across a certain time frame and, or simply a total occurrence count. Some parameters may also be more lenient when the text corpus is known to be of a consistently high quality, decreasing the possibility of noise mentions and known wrongful additions (such as usernames of very active users in forums).

In a more long-term case, the pretrained base word embedding vocabulary for this method would have to be sufficiently large, so that the amount of words added post-training, is small compared to the amount of embeddings created in the original data processing. In a live RS, this post-training data would be expected to be a based on a relatively constant stream of new documents for topic generation in a real-life application, and its properties would be expected to differ from some of the characteristics in this research experiment.

Maintaining a reasonable word mention threshold for addition to the embedding vocabulary is important in both cases, ensuring that there isn't a multitude of word embeddings used only a handful of times before they merely take up space in the word embedding space while increasing the ratio between embeddings added post-training and originally trained embeddings. At the same time, though words may no longer be present in new documents, the information they contain about user history may still be useful. Given a sufficiently large embedding space and vocabulary, the embeddings should remain useful and keep their original characteristics despite some updates.

When words are added after initial training, they will also need to be assigned to an appropriate topic from the topic embeddings, in order for the word to be able to contribute to future inference of document topic proportions. This may also impact the required threshold, possibly demanding a higher threshold to ensure enough documents for topical information, though intuitively one could expect that sufficient context information to create a word embedding for the word should be sufficient to discern some topical belongings as well.

For research purposes, and use within a limited time scope for training, this threshold is less important, due to the limited amount of documents we will process compared to a live RS running year-round, and we focus on the vocabulary update in a shorter time space to discern feasibility of using such a method for this purpose at all. Thus we will likely keep a lower threshold in experiments than what would be natural in a system environment running over time.

4.2.3 User Prediction

After generating affinity user models for the test users based on the topical communities, a set of hold out documents are predicted and ranked for these users.

The unseen documents are transformed to topic vectors and assigned clusters with the selected models, before being scored and ranked for each user. Using the trained cluster model, a user model is generated by classifying each document in the user's collection of documents. That is - by classifying all unique documents the user either has published or commented on. The users memberships to each topical community is then decided by number of interactions within each community. More precisely membership strength, ω, is given by

$$\omega_{u,c} = \log(count_{u,c} + 1) \tag{4.9}$$

where $count_{u,c}$ is the number of unique document user u has interacted with in cluster c (log normalized to avoid particularly active user accumulating too high scores).

Each user-cluster membership can be seen in context of both the user's behaviour and cluster tendencies. For instance, CF data typically exhibit large user and item biases, systematically showing some users tending to give higher ratings than other, and some items to overall receive higher ratings [101]. Translating this to the news domain mainly relying on implicit feedback, it is reasonable to assume that some users tend to be more active overall, and that some clusters tend to have more user activity connected to its documents than others. A cluster membership for a user with low overall activity, but showing special interest in this cluster, and thus deviating from own habits, should then be strengthened by a higher rate than a user tending to be very active in a large amount of clusters. Likewise with clusters, where high activity is not very common. This also enables ordering of documents within a cluster, as they initially are considered to be of same relevance.

To encapsulate these effects, inspired by [101], we denote the parameters δ_u and δ_c as deviations of average for user u and average for cluster c from user's initial preference $\omega_{u,c}$. The score $s_{u,i}$ is then:

$$s_{u,i} = \omega_{u,c} + \delta_u + \delta_c \tag{4.10}$$

, where document i belongs to cluster c, and $\omega_{u,c}$ (equation 4.9) is initial membership weight of user u in cluster c.

In addition to the user models described in the experiments, some naive models were generated for comparison. For instance, a topical centroid model is tested by creating user models from averaging all the topical document vectors the user has interacted with, both for LDA and topic embedding. These unseen documents are then ranked by calculating distances from the document's topic vector to the user's topical centroid. Another simplistic model shows how the predictors compares to the original manually chosen labels, by naively recommending an ordered list of documents based on the original category distribution of the user's document collection. At last, a predictor is added, that simply predicts a randomly ranked list, with the same ratio between expected and not expected documents.

In the experiment setup, users with a set number of known relevant documents are chosen for testing to ensure that relevant recommendations can be made.

Adding these documents to the list of documents to score for each test users, guarantees a consistent potential for true positive hits. When selecting m additional documents to predict for a user, each category is given a ratio ς and the user's category distribution should then be reflected in the predictions by predicting $\varsigma \cdot m$ most recent documents in each of the categories, ordered randomly.

4.2.4 Evaluation

The lists of predictions are then evaluated using MAP and RA, as described in 2.2.3 and 3.5.2. The MAP values are calculated at the full ranked list, as well as at different positions, denoted by @K. As RA is a metric based off of average Percentile Ranking (PR), it is difficult to make a universally functioning calculation at different positions. Cutting the list at top K and calculating average PR the standard way will punish finding a group of relevant documents among top K more than only finding the first in this group, and leaving the rest at lower ranks outside of top K. This is not a problem in the full average PR where each document's position is guaranteed to be found and taken into account.

Several alternative ways are explored to address this issue, in order to obtain metrics comparable to MAP. One alternative is calculating the average PR for the first K relevant documents found. This is a solution which can be used with some precautions in mind - if the first K-1 relevant elements are found at good ranks, but the next Kth element is found at the bottom, the metric will not be able to capture that this is better than K elements being found at mediocre positions. Another alternative is cutting the list at top, but modifying the average PR calculation by including the non-relevant elements to avoid punishing larger groups of correct predictions. However, setting the PR of the non-relevant items requires a well defined function, as intuitive functions such as setting PR= 1 or PR= $1/K$ yields very unstable results. For these reasons, only the RA calculated from the average PR of the full ranked list is used.

In order to see how the span of users' interests affects the prediction, Gini index is calculated for each user vector. This is originally a criterion to minimize the probability of misclassification, but may also serve as a good indicator of the diversity in a users interests as it considers how skewed or even a distribution is across classes. Entropy is another impurity metric considered, but Gini is chosen as it is less computationally demanding due to not using log function. The Gini index is calculated by:

$$Gini = 1 - \sum_j p_j^2 \tag{4.11}$$

where p_j is the probability of class j and a perfectly classified Gini index is zero.

By calculating this index and ordering the users by their Average Precision (AP) and RA, one can see if trends related to interest range are present, like whether users with a broad range of interests tend to be harder subjects for user predictions, or whether the models are affected by this. The results are then compared to each

other and to other applicable predictors.

In cases where user click history is absent, other indicators of interests may need to be used as replacement indicators, such as comments. When using these as stand-in representations of reads, a binary value denoting whether an article was read or not, based on whether a user commented it, might be the most appropriate choice to mimic a real user-article history information collection. Evaluations need to be seen in light of this, as users generally tend not to comment each article read, and the subset of true positives and false negatives may increase compared to a proper user read or click dataset.

Chapter 5

Experiments

In this chapter the setup for each experiment described in the previous chapter will be described. It will first present the general prerequisites, including data, preprocessing, tools and setup, as well as evaluation, presenting parameters and details in each experiment.

5.1 Prerequisites

The following sections provides a description of the experiments done in the preparation of the described solution. This section will first introduce the data sets and criteria for the data used in the experiments, as well as tools, setup and running environments, before next section describes each experiment in detail.

5.1.1 Data

As observed, content and behaviour in the news domain differentiate from other comparable domains on critical aspects, and recommendation is often more complex than in many other RSs for a number of reasons [1]. As a result of this, this research encompasses a combination of multiple data sets, supporting different requirements, which will be presented in this section. The data sets should fulfill the following requirements to reflect the domain:

- The data should be **unstructured**, as data in the news domain often is inconsistent or incomplete, and hard to interpret because of this.
- Items should be **text documents comparable to news articles**, spanning over a wide range of topics, and include id's and time stamps.
- The data should contain history about **user preferences**, along with persistent user ids and time stamps.
- User consumption history should **refer to ids in item set**.

- As predicting specific documents yields a very strict evaluation and qualitative evaluation needs to be taken into account, the data set should be **categorized** to support evaluation.
- If multiple data sets are used, they must **match with respect to time and content**.
- The data must be **available and interpretable**. Even how obvious it may be, this is naturally crucial and despite data sets fulfilling all previous requirements exist, it is not a matter of course that they are public or complete due too privacy or pay walls.

In addition it could be interesting with more context about the users, such as geographical area, age, gender and such. However, data sets for recommender systems are few and often inadequate [7], and obtaining these features along with persistent user ids beyond session history challenges privacy concerns and are often transformed or held back.

Data Description

Due to lack of access to a data set alone fulfilling all the requirements, two data sets were mainly used, complementing each other. One was used to cover user interactions, and one for the categorized article documents. It should be noted that they are both publicly collected and thus do not contain further background information about sessions and interactions, than publicly visible comments and posts. Each of them will now be presented briefly, and compared with other data sets within the recommendation domain, displayed in table 5.1. Further details and statistics will then be accounted for in chapter 6.

Comparison with Relevant Data Sets					
Dataset	**Users**	**Items**	**Edges**	**Density (%)**	**Rating Scale**
News Aggregator[a]	NA	422 419	NA	NA	NA
20 Newsgroups[b]	NA	$\sim$ 20000	NA	NA	NA
Reuters-21578[c]	NA	21 578	NA	NA	NA
reddit[d]	33 079	390 831	2 478 733	0.02	Comment Counts
Last.fm[f]	1,892	17,632	92,834	0.28	Play Counts
MovieLens 20M[g]	138,493	27,278	20,000,263	0.52	[0.5-5]

[a] https://www.kaggle.com/uciml/news-aggregator-dataset
[b] http://qwone.com/~jason/20Newsgroups/
[c] http://www.nltk.org/book/ch02.html
[d] https://files.pushshift.io/reddit/
[e] March-August 2014
[f] https://grouplens.org/datasets/hetrec-2011/
[g] https://grouplens.org/datasets/movielens/

Table 5.1: Comparison of data sets within recommendation systems.

reddit.com
reddit.com is a popular social content aggregation and discussion website where users can submit content to topic centered forums called “subreddits”. The topics,

namely subreddits, ranges across a wide variety, from very general, like "world news", to more specific, like specific versions of Python programming language. The posted content may be in the form of text posts, images or links. Being a dynamic, massive, content rich content platform constantly evolving, with a wide range of users, gives reddit a good potential for analysis, but also leads to challenges due to volatility and sparsity in the data set, as well as challenges in handling the data considering the size. By aligning selected time slots for user interactions on reddit.com and published news articles, the data will hopefully be as comparable as possible, as same topics might be discussed. In table 5.1, one can see that the data set is considerably more sparse than benchmarks data sets used for music (Last.fm) and movie (MovieLens) recommendations, which is representative for the news domain.

News Aggregator
Requiring a categorized news data set containing general categories, the well known 20 Newsgroups data set serves as a good candidate. However, due to reddit.com's relatively young age, a data set of more recent content is desired. This also applies to the benchmark data set Reuters-21578. The News Aggregator data set [103] contains categorized headlines for 422 937 news stories collected in a period of five months during 2014 (March 10th - August 10th), making it possible to retrieve rich subset from reddit.com data set from the same months with corresponding categories (business, science and technology, entertainment, and health), which mapping is displayed in table 6.3. In addition to convenience regarding comparison, it does not rule out the opportunity to attempt to connect the data sets at a later point.

5.1.2 Preprocessing

The used data sets have all been preprocessed in the same way, using basic methods for data cleaning. This includes removing stopwords and punctations, performing lemmatazation, and adding bigrams. Stopwords were removed using the widely used Onix[1] stopword list and more web and forum specific Galago[2][3] stopword list. In addition non-latin letters were removed and remaining were normalized by removing umlauts and accents. One letter terms and terms solely consisting of numbers were also removed. Infrequent words appearing in less than five documents are also removed.

5.1.3 Tools and Setup

The programming language used in the project is Python, as this supports a vast amount of libraries and tools for data analysis. To explore and transform the data

[1] http://www.lextek.com/manuals/onix/stopwords1.html
[2] https://sourceforge.net/p/lemur/galago/ci/default/tree/core/src/main/resources/stopwords/rmstop
[3] https://sourceforge.net/p/lemur/galago/ci/default/tree/core/src/main/resources/stopwords/forumstop

we have used Pandas, which is a free Python library making the data easier to work with by providing data structures and analysis tools of high performance[4]. In addition Jupyter Notebook[5] web application has served as a great resource in exploring and wrangling the data. Visualisation has been done using Matplotlib[6] and Seaborn[7]. Topic modeling was done with help from Gensim[8] and clustering using scikit-learn[9] and scikit-learn-contrib[10]. The experiments was all run in operation systems Ubuntu 16.04.* and macOS High Sierra, running on four different Intel servers with 2x2 to 4x10 cores. Details can be found in table A.2 in appendix.

5.1.4 Evaluation

As the document base is very sparse, test users are not sampled completely at random, but have to meet two requirements. An interaction $\gamma(u, i)$ is present when a user u has commented on or submitted post i, and the test users are required to have interacted with at least n unique documents that can be trained on, as well as m unique documents one can attempt to predict. Hence, each user to perform user predictions for initially needs to have interacted with at least $n + m$ unique posts. Assuming daily recommendations and approximately 230 published articles a day[11], 230 is found to be a fair number of articles to consider relevant for prediction. Using $n = 20$ and $m = 30$ as thresholds, 200 articles is then added to the test set in addition to the 30, while 20 is being used to generate the user model. Evaluation of metrics used in the processes will be presented next in description of methods for each experiment.

5.2 Experiments

5.2.1 User Modeling with LDA and Clustering

The following experiment is executed using three different data sets. This is the News Aggregator data set, as well as two versions of the reddit.com dataset to supplement user interactions, as described (5.1.1). Two versions of the reddit.com data set are used, where one considers only the posts' titles as documents, and the other one considers the posts' titles and content together as documents. The data sets will from now be referred to as Reddit Titles and Reddit Posts. The described process is fully run for all three data sets, except for the prediction step for the News Aggregator dataset, where support from Reddit data set's interactions is needed.

[4] https://pandas.pydata.org/
[5] http://jupyter.org/
[6] https://matplotlib.org/
[7] https://seaborn.pydata.org/index.html
[8] https://radimrehurek.com/gensim/
[9] http://scikit-learn.org
[10] http://contrib.scikit-learn.org
[11] https://www.theatlantic.com/technology/archive/2016/05/how-many-stories-do-newspapers-publish-per-day/483845/

LDA is run grid searching parameter t (number of topics), in the range $[0, 230]$ with step size 10, in addition to t-values 4 and 5 to compare to the original category partitioning. Each model is then evaluated on a held-out validation set by measuring perplexity, bounds and coherence (section 2.4), from which a representative selection of models is chosen.

The clustering is done using k-Means and HDBSCAN. k-Means is run both with Kullback–Leibler (KL) divergence as distance metric, in addition to the more common Euclidian distances, as each document is represented by a topic distribution generated by LDA and the former hence may be more suitable. The grid of parameter values for k is $[0, 100]$ with step size 10, and also here 4 and 5 to reflect original partitioning of the News Aggregator dataset. Models are also trained for $k = t$ to observe if the clustering simply reproduced the topics and thus are redundant. HDBSCAN's primary parameter to effect the resulting clustering is the minimum cluster size ρ, deciding the smallest size grouping to consider a cluster. In addition minimum amount of samples, ϱ, can be selected indicating how strict the algorithm is when classifying noise. The grid of parameter values for both ρ and ϱ is $\{5, 10, 15\}$.

The best parameters are chosen based on Silhouette coefficient, C-H index (section 2.1) and a qualitative evaluation, and the best clustering are used to create user models. In addition, the clusterings are visualized in two and three dimensions using the dimensionality reduction methods Linear Discriminant Analysis, t-SNE and PCA. These are labeled with both cluster id and category (and for the Reddit documents also subreddit) to see the overlap between them and potentially different levels of granularity. The most informative visualizations is then presented in the report.

5.2.2 Updating Word Embeddings

Some precursory experiments are run on top of the initial embedding of 180 000 words generated in [56]. In order to coarsely test the validity of adding new words into an existing word embedding, we will look at a few qualitative and statistically quantitative examples with words or terms which suddenly became a matter of public interest after the snapshot of Wikipedia used to train word embeddings was collected, and thus would be subject to the embedding adding process we have mentioned. Estimating the times at which these topics, or search terms, became of public interest online can be achieved by accessing the Google Trends engine[12], a search engine providing graphs of the interest levels for terms on the widely used Google search engine[13]. This provides us with a relevant timeframe of Reddit posts to process in order to discover relevant ones. In addition to this, the updating scheme will be tested on the training corpus, by looking at changes in topic and cluster quality before and after updating.

Due to the computational requirements of rerunning a full word embedding process to see where a new word would have been placed relative to others, we opt

[12] https://trends.google.com/trends/
[13] https://www.google.com

for comparing similarity statistics of the 180 000 words already embedded to those of the new word added for the quantitative validation. This way, we can observe if the new word embedding behaves somewhat similar to the original embeddings, as well as giving us an option to compare its behaviour across model variables such as window size. It is pivotal that the new embeddings does not appear to statistically diverge from those trained traditionally. If new word embeddings appear to be outliers compared to the original embeddings, not having many or any words in its close neighbourhood, they may skew topics containing them in a direction which removes topical coherence from its representation. Similarly, if most new word embeddings tend to be too close to many of the words in the embedding vocabulary, they may not provide any useful distinctions to topics and could simply be a source of noise.

In addition to considering their statistical characteristics, we will visualize the new embeddings in a low-dimensional space along with its closest words. This allows us to quickly perform a sanity check of how relevant the embeddings' closest neighbours are. The variable parameters include a window size of which words contained within it will contribute to the averaging word embedding value. A pure quantitative consideration of what window size is suitable, and does not generate excess data for processing compared to topical gain from added contextual data, is not plausible. There is no ground truth of how the words embeddings should look for optimal model performance (must be seen in context of its task), semantics are often more of a qualitatively measurable element and running a new word embedding for each time is not a possible solution. We will therefore aim to run a few examples with qualitative evaluations based on concepts/terms that were not commonly used at the time of the Wikipedia Snapshot used to create the word embedding (March 2015).

When looking at different parameters, we consider the distance between the resulting word embedding for the new word and words that are expected to be close, as well as looking at the closest neighbours of the new word's embedding. We want the list of closest words to seem related to the new word embedding, as well as the distance to related words to be small compared to an average distance to all words. This behaviour is expected if enough of the context words influencing the new word embedding are in fact related, and not overly divergent among themselves. Duplicitous meanings/context is an issue in traditional word embedding training as well.

The distance metric used for calculating embedding distances is cosine similarity, giving us values between -1 and 1, where 1 indicates that two word embeddings are identical with distance 0 and -1 indicates that two embeddings are complete opposites.

5.2.3 User Modeling with Topic Embedding

Experiments are run using the TopicVec algorithm by adapting code made available by the authors. TopicVec is run on top of the continuous word embeddings for 180 000 words in 500 dimensions on a Wikipedia snapshot dataset from March 2015, pretrained in TopcVec's predecessor [82]. With Wikipedia generating articles about

places, people, events, science and everything inbetween in a relatively structured manner, it seems a reasonable base for word embeddings being used for discussions of real life events and information. The question of correctness of information on Wikipedia is considered a non-issue in regards to the word embeddings based on these characteristics, as the important factor is the context itself when generating word embeddings.

In TopicVec, preprocessing includes removing words not present in the word embedding vocabulary in addition to those who are classified as stopwords, due to these words being considered to be very rare or to provide little to no discriminatory information. Though this may be preferable in the case of their experiment, when recommending articles or items, identifying the difference and likeness between items is important and may require larger word embedding vocabulary than in this case. As mentioned earlier, we would also have to assume that new words may appear in many news documents that are being topic modeled, which can be important enough to warrant being added to the word embedding in order to improve future topic modeling.

The topic embedding user modeling is performed by running kmeans and HDBSCAN on the Reddit Posts data set described by varying numbers of topics. In addition to this, some topic modeling and clustering is tested on the Newsaggregator dataset, in order to compare tendencies. As with the LDA user models, users are then represented as k-dimensional vectors, where k is the number of clusters and each value represents the strength of cluster membership. When deciding which trained models to use for user prediction, both the clustering metrics and a qualitative assessment of the topics are taken into consideration.

5.2.4 User Prediction

As mentioned in 5.1.4, 230 documents are attempted predicted for each user, where 30 of these are known to be in the user's interaction history. The ranked list is MAP evaluated at position 3, 5, and 10, in addition to the full list and 30, being the number of articles we wish to recommend. A ranked list sorted by RA is also generated. The results from the different models are compared, as described in the experiment methods. Documents classified as noise are not predicted.

Chapter 6

Results and Discussion

In the following chapter the research results of the thesis will be presented and discussed. We will first, in section 6.1, have a look at some characteristics of the preprocessed data utilized, before presenting and discussing the results of each of the experiments building models in section 6.2 and 6.3, before presenting their performance in prediction in section 6.4, and finally, summing up the results in section 6.5.

6.1 Data Characteristics

Deeming it necessary to use alternative data sets to traditional news data sets due to lack of sufficient persistent user history found in the real news data sets, it was desirable that the alternative could mimic the behaviour and characteristics of a traditional one. The following section will present and discuss characteristics found in the used Reddit data set, comparing it to data sets of traditional news.

Some of the comparable key statistics are listed in table 6.1, and as can be seen, the News Aggregator data set does not seem to deviate to much from the Reddit data set. The most notable differences between the two, is the vocabulary size and number of documents with less than five words. This may naturally affect predictions, providing larger contexts and more specified items to train on. Considering the relatively moderate increase in word count in Reddit compared to News Aggregator compared to the large increase in vocabulary size, there is a chance that many words in the Reddit data set are rarely used due to for instance rare words or misspellings. Looking at 6.2, it becomes clear that all the data sets have high percentages of words in their vocabulary which are used very rarely. Across all three data sets, 60% of the words or more appear 10 times or less. This trend of rarely mentioned words is especially strong in the Reddit data set, which potenitally may cause noise.

Spec.	Reddit Posts	Reddit Titles	News Aggregator
Document count	279 504	277 069	335 891
Avg. doc. length	13.99	6.85	7.70
Median doc. length	7	6	8
Min doc. length	2	2	2
Max doc. length	2 362	78	2 381
Total word count	3 910 261	1 896 976	2 584 533
Total vocab size	137 502	58 033	60 614
Documents below 5	35 275	47 305	14 721
Documents below 10	162 663	207 358	166 123

Table 6.1: Data set statistics for training set.

Training corpus	Reddit Posts	Reddit Titles	News Aggregator
20th	4.0	4.0	5.0
40th	5.0	6.0	6.0
50th	6.0	7.0	8.0
60th	7.0	9.0	10.0
80th	14.0	19.0	22.0
90th	29.0	45.0	57.00

Table 6.2: Word count percentiles for all words in vocabulary in each data set.

Table 6.3 shows the distribution of documents in the Reddit and News Aggregator data sets. Here, one can also observe the mapping between the labeling in the data sets. Comparing the data sets, both sets have a distinctly larger portion of the documents in the entertainment category, and smaller in the health and fitness category. The business category is the second largest in the News Aggregator data set, while being the third largest in the Reddit data set. Correspondingly, this order is flipped for the science and technology data sets. This can stem from a numerous different reasons, one being that business often may refer to technology and vice versa, and another that there are simply more articles being published about business than science and technology, and that science and technology being a more popular discussion topic. Other potential reasons include a difference in behaviour patterns in discussions within business related forums and in technology and science related forums.

As can be seen, the ratios between the train and test sets are not consistent. This is due to the size of the test set being decided by a requirement for a number of users satisfying the activity threshold in terms of interaction history. It is desired

to obtain the largest possible subsets for users to be trained and tested on, and the ratio is therefore not fixed. In this case, there is not known any notable causes of this, but a ratio which do affect the models however, is the ratio between categories, which will be discussed in the experiment results.

Reddit Posts			News Aggregator		
Subreddit (Category)	**Train set**	**Test set**	**Category**	**Train set**	**Test set**
Business	28 538	1 699	**Business**	92 216	23 086
Finance	5 146	212			
Economics	4 520	1 013			
Businessnews	152	52			
Total Splits	**38 356**	**2 976**			
Total Sum	**41 332**			**115 302**	
Music	47 649	832	**Entertainment**	121 295	30 237
Movies	34 670	2 483			
Television	10 192	810			
Entertainment	5 119	459			
Music news	674	51			
Celebrities	651	51			
Total Splits	**98 955**	**4 686**			
Total Sum	**103 641**			**151 532**	
Technology	38 015	3 792	**Science and Technology**	86 269	21 472
Science	14 292	1 331			
Everything science	2 386	310			
Tech	1 828	278			
Tech news	889	164			
Total Splits	**57 410**	**5 875**			
Total Sum	**63 285**			**107 741**	
Fitness	17 819	71	**Health and Fitness**	36 111	9178
Health	13 205	956			
Health care	2 083	105			
Nutrition	1 040	0			
Public health	202	22			
Global health	98	20			
Total Splits	**34 447**	**1 174**			
Total Sum	**35 621**			**45 289**	

Table 6.3: Data set category distributions.

6.2 User Modeling with LDA and Clustering

The following section presents results from experiments described in section 5.2.1. Where not provided, detailed metrics and results can be found in appendix. The visualizations are done using Linear Discriminant Analysis and PCA only, as t-SNE yielded very poor visualizations for these experiments (an example can be viewed in figure A.1 in appendix). As mentioned, the experiments were run on the Reddit Titles and Reddit Posts dataset.

6.2.1 Topic Modeling with LDA

The first part in the process of generating user models through LDA and clustering, is representing documents through topics by selecting a proper topic model.

Selection of t-Parameter

Figure 6.1 shows a scatter plot of how number of topic affects the scores for the Reddit Titles data set, and the same trend was observed for the two other data sets. Note that the values are standardized and shifted to ease illustration, as the three metrics are all positioned in different ranges. Initial perplexity values were positive, while coherence and bound initially were negative. Striving for a positive coherence and bound close to zero, these scores are better for lower numbers of topics, while perplexity, striving for a lower score, yields better results for higher numbers of topics.

A representative selection of models were then chosen to cluster. The selected were models with 5, 15, 40, 80 and 100 topics, in addition to 4, being the number of original categories. This covers each of the intervals between the intersections until 100. More than 100 topics were not regarded as bounds and coherence scores keep falling with increasing topic number, and perplexity does not improve either. Full results can be viwed inn table A.3 in appendix.

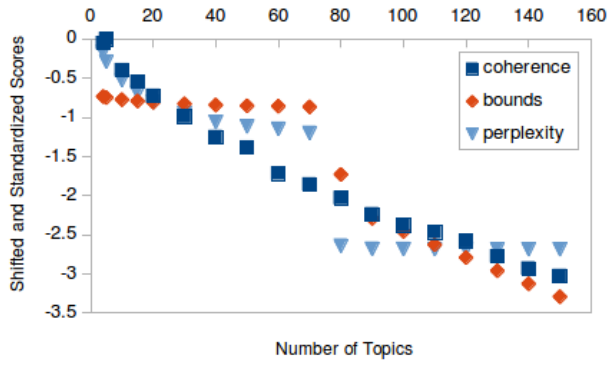

Figure 6.1: Topic modeling on Reddit Titles data set.

Table 6.4 shows top 10 most probable terms per topic generated by LDA models with four topics. If we assume that word co-occurence indicates semantic relatedness, well defined categories should be represented in the four topics, being **business, entertainment, science and technology, and health and fitness**. To avoid repetition in the table, terms of displayed bi-grams are removed if it has appeared as a sub-term in a bi-gram shown earlier in the list, such as the terms "hip", "hop", and "hip hop". Some places only the sub-term is be present because this has been a more probable term, like when people are mentioned more often by last name. Experiments were also performed where bi-gram sub-terms were removed from docs with discovered bi-grams in advance of the topic modeling, but did not result in more meaningful topics (table A.4 in appendix), and a reason may be the loss of connections between topic defining sub-terms, like losing the connection between a document containing only the significant artist first name "kanye" and a document containing his full name "kanye west" once "kanye west" is defined as bi-gram.

Data Set	Topic 1	Topic 2	Topic 3	Topic 4
Reddit Posts	business	song	watch	weight
	market	rock	movie	start
	health	music	online	time
	company	movie	service	lb
	check	film	free	eat
	internet	video	trailer	workout
	help	love	stream	lift
	home	album	tv	help
	app	band	google	gym
	technology	indie	hd	try
Reddit Title	business	movie	rock	hip hop
	google	watch	song	weight
	service	online	music	help
	world	indie	video	star
	buy	help	album	change
	company	music	cover	rap
	android	rock	pop	workout
	mobile	film	loan	time
	top	free	alternative	war
	tv	time	live	future
News Aggregator	market	kim kardashian	apple	amazon
	rate	video	price	season
	star	gm	microsoft	ceo
	china	wed	sale	video
	data	ebola	report	bank
	rise	cancer	game_throne	fire
	profit	samsung galaxy	office	time
	war	beyonce	gas	pay
	fall	miley cyrus	buy	deal
	share	jay	court	star

Table 6.4: Top 10 terms for LDA topic models with four topics.

From the extracted topics displayed in table 6.4, one can observe that the

trained LDA models for mostly manage to extract meaningful topics, yet none of the models succeeds to capture all of the four original categories. Taking into consideration that the News Aggregator data set initially is divided into four categories and consists of article headlines, which one could expect to have a higher demand of being informative than titles of internet forum posts, it is surprising that LDA does not extract more meaningful topics from this data set.

The reason for why LDA struggles extracting the four categories may stem from different reasons. It could for instance be caused by too conservative or lenient preprocessing, or it could simply be lack of meaningful, describing titles. Another reason might be that the number of categories does not fully reflect the content in terms of granularity. Hence, some articles in different categories might be partially associated to same topics, like business and technology, and making them blend. In addition one can see that some larger categories seem to have been divided into finer, more precise partitions, like entertainment in music and movies. This is to a greater extent also the tendency for 15 topics, which can be viewed in appendix (table A.5, A.6, and A.7).

Table 6.5 shows the fifteen most significant topics from the LDA topic model run on the Reddit Posts data set with 80 topics (results from the other data sets in A.8 and A.9 in appendix). The topics are sorted by significance (read horizontally, then vertically), with topic 69 as most significant and topic 45 as the 15th most significant. Again, some topics appear as vague and blended, but as mentioned in section 2.4, human judgement of the usefulness of a topic is a difficult task as a consequence of bias and subjectivity. For instance, topic 26 and topic 11, being the 13th and 14th most significant topics, comes out as clearly related to health and fitness, whereas topic 69, being the most significant, appears as less obvious.

Reddit Posts				
Topic 69	**Topic 43**	**Topic 48**	**Topic 10**	**Topic 76**
build	boy	gt	electronic	album
apps	eye	blood	repair	release
answer	fall	low	door	metal
wed	rise	event	window	god
person	sex	brain	garage	instrumental
fee	forget	stand	jazz	share
hide	wait	shoe	microsoft	son
volume	australia	foot	freeze	heavy
ca	believe	floor	dubstep	jay
pump	sea	budget	itunes	classical
Topic 73	**Topic 38**	**Topic 27**	**Topic 60**	**Topic 15**
la	movie	amaze	name	report
en	stream	title	girl	view
return	hd	spider	kid	risk
final	free	festival	guide	develop
commercial	scene	van	surgery	research
shop	stop	empire	choice	claim
map	download	imdb	catch	access
class	watch movie	pack	expert	funny
plus	action	sister	drink	bird
el	hill	tiny	wish	label
Topic 77	**Topic 26**	**Topic 11**	**Topic 6**	**Topic 45**
brand	lose	eat	website	app
email	weight	protein	web	mobile
touch	lb	diet	design	phone
lady	calorie	food	source	android
solo	eat	healthy	content	launch
south	gain	meal	boost	street
gun	cut	drink	oil	device
board	pound	weight loss	block	io
auto	start	shake	field	comedy
discount	weigh	chicken	acid	robot

Table 6.5: Top 10 terms for the fifteen most significant topics extracted with LDA topic model run on Reddit Posts data set with 80 topics.

Document-Topic Matrix

After selecting topic models, document topic matrices were generated for each models. The following tables shows examples from the result on the Reddit Posts data set with $t = 15$. Table 6.6 displays five documents in the document-topic matrix generated using the LDA model, and the user-cluster matrix in table 6.7 shows five user models generated by clustering k-Means ($k = 4$ and distance metric $= KL$) done on the document-topic matrix. In addition table 6.7 shows the category and subreddit distribution for each user's document interactions in the test set which the user models are based on.

docid	Topic 1	Topic 2	. . .	Topic 14	Topic 15
D1	0.02	0.36	. . .	0.02	0.02
D2	0.00	0.00	. . .	0.00	0.08
D3	0.70	0.00	. . .	0.00	0.00
D4	0.01	0.01	. . .	0.45	0.01
D5	0.02	0.02	. . .	0.77	0.02

Table 6.6: Document-Topic matrix for $t = 15$.

User ID	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Categories	Subreddits
U1	1.79	1.61	2.30	0.69	55 x ScienceTech	54 x technology 1 x tech
U2	0	1.79	1.1	1.39	12 x Entertainment 9 x ScienceTech	10 x television 9 x technology 2 x movies
U3	0	2.08	0.69	0	22 x ScienceTech	19 x science 3 x EverythingScience
U4	0.69	0	0	2.56	33 x Health 4 x Business	33 x Fitness 2 x finance 2 x Economics
U5	0.69	0	0.69	2.48	33 x Health	33 x Fitness

Table 6.7: User-Cluster matrix after LDA Topic modeling ($t = 15$) and k-Means clustering ($k = 4$).

From the interactions in table 6.7 it seems reasonable that the users with user id's $U3$ and $U4$ are connected to different clusters, and that $U4$ also is connected to same cluster as $U5$, while $U3$ is connected to $U1$ and $U2$. One can also observe topics capturing a finer granularity of interests than the low granularity original news categories from the News Aggregator data set. For instance, by looking at the mapped subreddits for user $U1$ and $U3$, seemingly sharing the same news preferences based on category, but yet having different user levels, one can observe that at a lower level $U1$ is more interested in technology and $U3$ in science.

6.2.2 Clustering

We will now provide a closer description of the experiments performed to optimize the clustering step producing user models, as presented previously.

k-Means

figure 6.2 (execution table A.10 and A.11) shows how the C-H index and Silhouette coefficient are affected by number of topics and clusters. For k-Means with euclidian distances the Silhouette coefficient performs marginally worse on higher k-values, while for k-Means with KL divergence this difference is more significant. For both, the C-H index is very unstable, and t does not influence as much as k, apart from k-Means with euclidian distances, where the combination of low t's and high k's yields worse Silhouette scores.

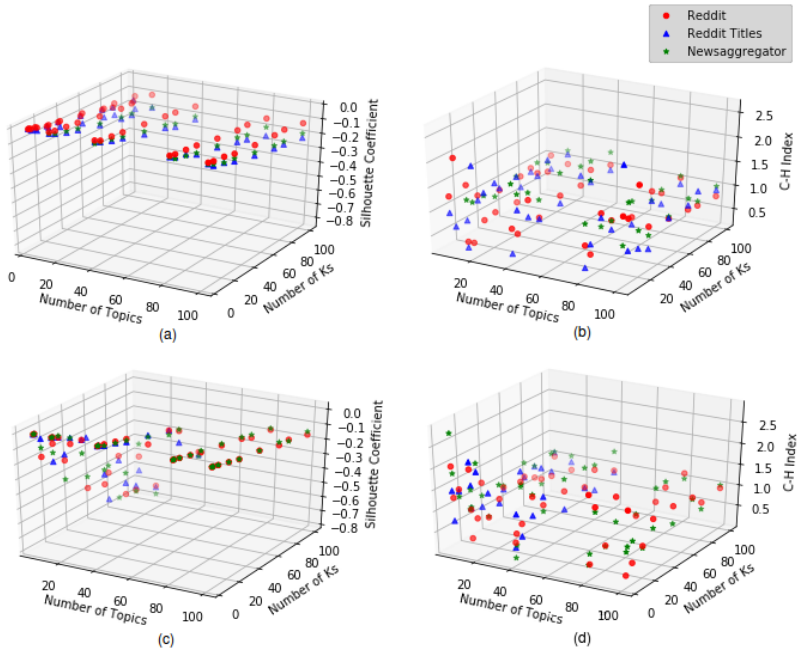

Figure 6.2: Silhouette score and C-H index for k-Means (a)(b): Euclidian (c)(d): Kullback-Leibler.

Picking the best models for k-Means is done by selecting the three models with the best scores for C-H index and Silhouette coefficient, in addition to qualitative evaluation with support from metrics and visualizations presented later. Where equally good scores, the model is chosen by using the other metric as secondary sorting. The chosen models can be viewed in table 6.8 and 6.9.

Data Set	Distance	t	k	Silhouette	C-H
News Aggregator	KL	5	4	**-0.01**	2.79
Reddit Title	KL	15	4	**-0.01**	2.21
Reddit	Euclidian	4	4	**-0.01**	2.05

Table 6.8: Top three k-Means models based on Silhouette Coefficient.

Data Set	Distance	t	k	Silhouette	C-H
News Aggregator	KL	5	4	-0.01	**2.79**
Reddit Title	Euclidian	100	5	-0.03	**2.58**
Reddit Title	KL	15	4	-0.01	**2.21**

Table 6.9: Top three k-Means models based on C-H index.

In the tables one can see that the model trained on the News Aggregator data set, using LDA topic modeling with $t = 5$ and clustering using k-Means with $k = 4$, performs the best both when it comes to C-H index, having the single highest score, and Silhouette coefficient, sharing the best score. Note that this model can not be used for user modeling due to lack of user interactions with documents, but as one of the reasons for including the Reddit Posts data sets is their relatedness and comparability to the news domain, it is valuable to observe how the same methods performs on this data set as well.

As one can observe, neither scores are remarkably high, and the best Silhouettes scores are even negative and close to zero, indicating overlapping clusters. This can also be observed in figure 6.3 (reduced by Linear Discriminant Analysis), next to figure 6.4, where top 10 words for each cluster may be observed in a word cloud, where words are sized by frequency and colors and labels corresponds to cluster. The clusters do have some coherence, but some words are obvious intruders and the clusters fail to capture the health and fitness category.

A reason for this performance may the too coarse granularity and that higher values of t or k should have been chosen. This is also supported by the C-H index' tendency to penalize high values of k, as seen in 2.1. Another cause may be that the k-Means is more suitable for clusters of even size, which may not be the ground truth. Taking the data sets' distributions (table 6.3) into account, it may be that the larger subreddits, music, movies, and to some extent technology, dominates the clustering. The same trend follows for clusterings on the Reddit data sets, which can be viewed more detailed in appendix (A.2 and A.3) for the second best model based on Silhouette score (also having the third highest C-H index), being run on 15 topics with 4 clusters. One could manipulate the data set and aid training by balancing the sizes of the categories, but as the original news data set (News Aggregator) also consists of unevenly sized categories and the goal is to see how the method may aid in news recommendation, it is left unbalanced. The heavily intertainment weighted clusters are not as evident in the second best model based on C-H index (appendix A.4 and A.5), with 100 topics and the number of clusters remaining low, at $k = 5$, but the model still seems to struggle finding k well-defined topical clusters, which is also seen in the Silhouette coefficient.

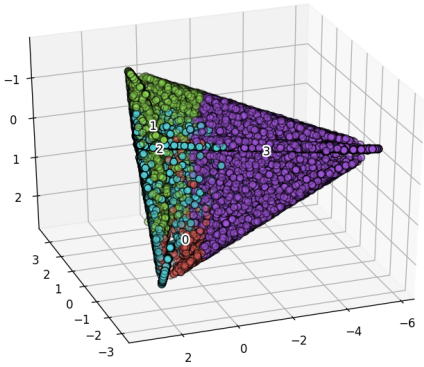

Figure 6.3: k-Means document clustering run with k=4 and KL distances on documents with 5 topics from the News Aggregator data set.

Figure 6.4: Word clouds for each cluster in model from figure 6.3.

The last model from the selected top models, is the only one run on the Reddit Posts data set where post content is included in the documents. It is run with eucilidian distances, and like the rest, it favors lower number of clusters. Having 4 topics and 4 clusters it is interesting to see whether this reproduces the original clusters, making this process redundant. Labeling the documents with their mapped categories and subreddits in figure 6.7 and 6.8, neither seems to overlap with discovered clusters observed in figure 6.5, indicating a potential of new insights from topical clustering.

Even though $k = 4$ is among the best parameters overall, the last case seems to be applicable to the previous models as well, as none reproduces the four categories. They do however form other categories, which is positive, as the goal is to extract information spanning further than the information provided by the manually labeled categories. The word cloud for the last model, illustrated in figure 6.6, shows four seemingly well defined topical clusters about movies (0), tech/business (1), music (2), and fitness (3).

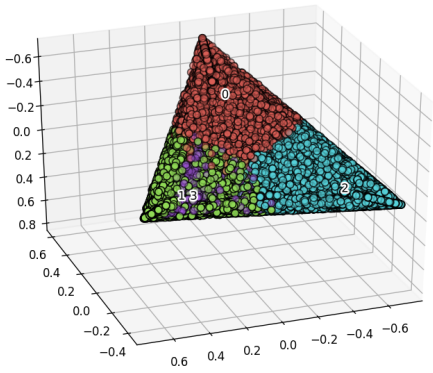

Figure 6.5: k-Means clustering run with 4 clusters and euclidian distances on documents with 4 topics from the Reddit Posts data set.

Figure 6.6: Word clouds for each cluster in model from figure 6.5.

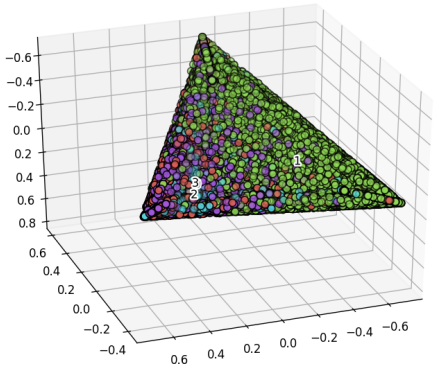

Figure 6.7: k-Means clustering (category labeled) run with 4 clusters and euclidian distances on documents with 4 topics from the Reddit Posts data set.

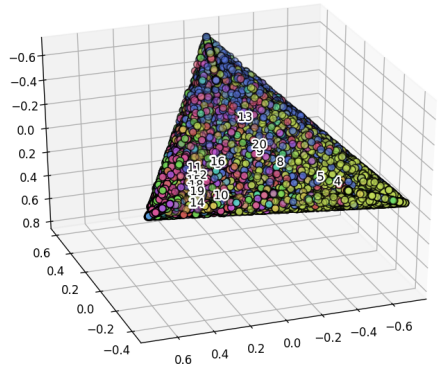

Figure 6.8: k-Means clustering (subreddit labeled) run with 4 clusters and euclidian distances on documents with 4 topics from the Reddit Posts data set.

While observing several of the top models seemingly being able to extract understandable topics, and sometimes at a finer granularity level, they are often of varying quality, and quantitative evaluation metrics is not an optimal evaluation alone. As observed, the lower numbers of clusters tend to be favored, failing to represent all clusters in one single model. To complement the models selected by metrics, an additional model is therefore selected, based on a combination of well (though not best) performing metrics, in addition to qualitative evaluation from visualizations. The selected model is trained as follows:

Data Set	**Distance**	**t**	**k**	**Silhouette**	**C-H**
Reddit	Euclidian	5	10	-0.02	2.01

Table 6.10: Qualitatively chosen k-Means model.

Though not being among the best three given either C-H index or Silhouette coefficient, the clusters quality does not appear reduced from visualizations, displayed with different labelings in figure 6.9, 6.10, and 6.11. As previously seen, large parts of the categories and subreddits are mixed in the clustering, but a significant part of the documents displayed in the upper right part of the figures appears to be clustered together with documents from the original category or subreddit. This is most clear in figure 6.10, where category 1, entertainment, in green have been clustered together. This is not surprising considering the results from the other models and category ratio in the data set. In the subreddit labeled clustering in figure 6.11, label 4 and 5 corresponds to the subreddits Music and MusicNews, while 13, 8, 9 and 20 (closer to the more vague blue area) represents the subreddits movies, television, celebrities and entertainment. From this it may seem like the clustering is able to retrieve a finer granularity of the news categories.

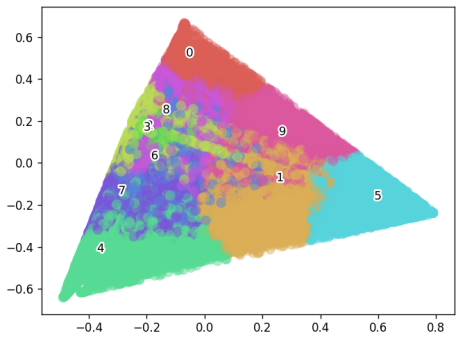

Figure 6.9: k-Means clustering run with 10 clusters and euclidian distances on documents with 5 topics from the Reddit Posts data set, labeled with clusters.

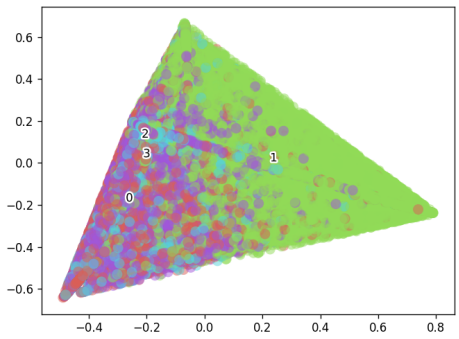

Figure 6.10: k-Means clustering run with 10 clusters and euclidian distances on documents with 5 topics from the Reddit Posts data set, labeled with categories.

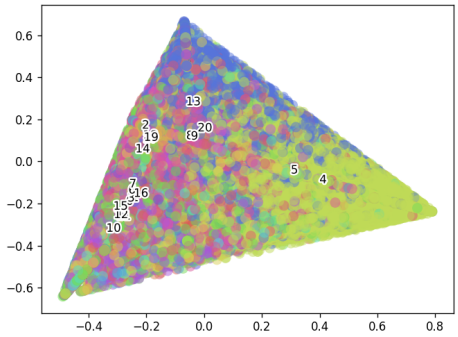

Figure 6.11: k-Means clustering run with 10 clusters and euclidian distances on documents with 5 topics from the Reddit Posts data set, labeled with subreddits.

Table 6.11 shows the five most frequent terms in the training documents in each of the clusters in figure 6.9). From these terms cluster 0, 9, 5, 8, and 1 seems to be entertainment related, aligning well with the labels in the other two figures. 4 is a well defined business cluster, blending over to tech, which has been observed to share traits earlier. 6 is not very well defined by its top five terms, but cluster 2 and 3 seem to be related to technology and fitness, though being very closely positioned in the diagram. To conclude, the qualitatively selected model seem to perform well, despite lower metric scores, and manages to capture more useful topics in one single model.

Cluster 0	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**
movie	music	google	weight	business
watch	song	apple	start	service
online	rock	mobile	time	health
free	live	android	lift	loan
video	cover	free	workout	market
Cluster 5	**Cluster 6**	**Cluster 7**	**Cluster 8**	**Cluster 9**
song	help	business	movie	movie
music	movie	service	film	rock
rock	time	technology	tv	film
album	fitness	google	trailer	watch
pop	people	market	watch	song

Table 6.11: Top 5 terms in the training data of k-Means model with 10 clusters and 5 topics.

HDBSCAN

Picking the best models for HDBSCAN was done in same manner as for k-Means, selecting the three models with the best scores for C-H index and Silhouette coefficient, in addition to a qualitative evaluation. The three best models given C-H index and Silhouette coefficient can be viewed in table 6.12 and 6.13. (All results in table A.12 in appendix.)

Data Set	**t**	**s**	**c**	**Silhouette**	**C-H**
News Aggregator	15	5	5	**-0.64**	1.01
Reddit	15	5	5	**-0.74**	1
News Aggregator	15	5	10	**-0.75**	1.02

Table 6.12: Top three HDBSCAN models based on Silhouette Coefficient.

Data Set	t	s	c	Silhouette	C-H
Reddit Title	100	10	15	-0.98	**6.19**
Reddit Title	100	10	10	-0.97	**4.39**
Reddit Title	100	10	5	-0.96	**3.82**

Table 6.13: Top three HDBSCAN models based on C-H index.

In figure 6.12 results produced by the best model given C-H index is illustrated, showing a sparse clustering with 1613 clusters. The sparse clustering is caused by a seemingly conservative noise classification, classifying as much as 90.41% of the documents as noise. Using the Gini index (experiment 4.2.3 in chapter 4) to calculate the impurity in the user models give an average index of 0.35 for the predicted documents and 0.73 for the noise classified documents, which may indicate that the algorithm struggles to create clusters of documents with more topics.

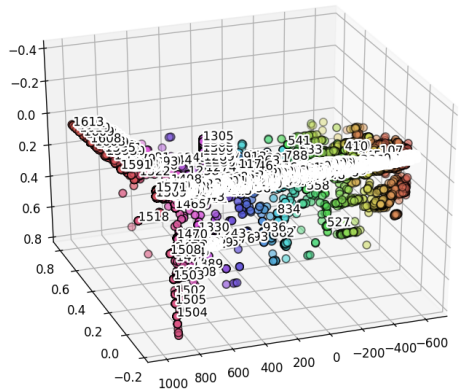

Figure 6.12: HDBSCAN model yielding the best C-H index, run on the Reddit Titles data set with 100 topics, minimum sample size 10, and minimum cluster size 15.

The best model given Silhouette coefficient uses only 15 topics and also lower values for s and c, which should make it less conservative regarding noise. The ratio noise classified documents is slightly lower, but yet high at 72%. The lower threshold for establishing clusters gives a pronounced increase in number of clusters, resulting in as much as 9902 clusters. As a result the labeling is even denser than in previous figure (6.12), and the figure is therefore not displayed. For the noise classified documents, the average Gini index is 0.63, while being 0.54 for the rest. However, despite apparently not being as affected by impurity in documents' topic mixtures as the previous one, the model still classifies too many documents as noise, and not being able to recommend 72% is overly high.

As mentioned in 2.1, both methods' performance are skewed to give higher scores to k-Means, observable in the low Silhouette coefficient. For this reason

also a qualitative evaluation of the clusterings was done visually. However, none of the clusterings appear more well organised or gainful, and the two top scored clusterings therefore remains chosen for later comparison.

6.3 User Modeling with TopicVec and Clustering

The experiments focusing on extracting user models through clustering of topic embedded documents, is divided into two parts. In the first part, a novel method for updating the underlying word embedding vocabulary is tested. Later, documents are clustered together based on their topic embedding features, derived from TopicVec.

6.3.1 Updating Word Embeddings

As mentioned in section 5.2.2, the initial word embedding experiment tests the general characteristics of word embeddings created by a centroid of their context words in the embedding space, as a sanity check for further experiments. This is done by extracting corpora from Reddit which is known to contain two words whose origin or rise in popularity can be pinpointed to a specific time by search engine statistics, "Brexit" and "Ransomware". Various values for word count thresholds and window sizes are tested for. While [104] suggests that 5 may be a more appropriate window size and perform better in a case similar to ours, though intended for a classification task, [105] determines that this window size will produce word embeddings that encapsulate more topical meanings. This may be redundant due to the intention of eliciting latent topics from the embeddings at a later time. We look at window sizes for both these cases, as well as some very high values. This is in order to ascertain the effect larger window sizes will have, as this is a very different training process than originally.

Statistics and Characteristics of Added Words

An example of statistical features is shown in 6.15. This table, along with Appendix tables A.13 and A.14, shows the statistics for the newly added embeddings of "Brexit" and "Ransomware". They show that the statistics for added word embeddings appear stable across several choices of window sizes and word count requirements, sometimes staying nearly unchanged across threshold changes, as well as staying within the statistical values featured in the original vocabulary, represented by minimum, maximum, average and median values for the statistical features across 100 000 of the original embeddings in 6.14.

Measure	Min	Max	Average	Median
Average Similarity	-0.16790	0.16229	0.06189	0.07255
Minimum Similarity	-0.45796	-0.11666	-0.21738	-0.21197
Maximum Similarity	0.27516	0.98912	0.60351	0.59362
# Positive Similarities	14 531.00	166 491.00	130 262.55	146 981.00
# Negative Similarities	13 508.00	165 468.00	49 736.45	33 018.00
25th Percentile	-0.24209	0.09119	0.00650	0.01623
Median Similarity	-0.18813	0.16672	0.05585	0.06629
75th Percentile	-0.11329	0.24849	0.11026	0.12042

Table 6.14: Word embeddings statistics for a sample of 100 000 words in the original word embedding vocabulary.

Window Size	3	6	12	24
Average Similarity	0.11686	0.12267	0.11868	0.11540
Minimum Similarity	-0.37818	-0.39351	-0.38232	-0.38207
Maximum Similarity	0.59561	0.60494	0.58683	0.58607
# Positive Similarities	152 624.00	154 805.00	155 499.00	154 663.00
# Negative Similarities	27 376.00	25 195.00	24 551.00	25 337.00
25th Percentile	0.03684	0.04339	0.04307	0.04048
Median Similarity	0.10915	0.11609	0.11243	0.10867
75th Percentile	0.19104	0.19822	0.19098	0.18678

Table 6.15: Brexit embedding statistics across window sizes with count threshold=400.

As the embedding statistics appear quite stable across count thresholds, one could potentially set threshold relatively low while still getting relevant results. However, we note that when running the embedding on the "Brexit" excerpt corpus with a mention threshold of 50 mentions, specific user names of Reddit users are embedded. This is highly disadvantageous and emphasizes the importance

of sensible parameter values. While low thresholds may yield statistically similar results, it also stands to reason that having slightly higher thresholds may give better or more nuanced embeddings, especially for words with higher ambiguity than "Brexit" and "Ransomware".

When embedding ransomware, additional words such as "Cyberattack" and "Wannacry" are also embedded at some of the thresholds. Coinciding so often in a subset of documents specified to contain "Ransomware" that they achieve most of the thresholds tested for, these words naturally achieve a very high similarity to the "Ransomware" embedding. This is however natural, as "Ransomware" is considered a cyberattack and Wannacry is an instance of a cyberattack of the ransomware type. Consequently, they would likely be embedded close to each other even if trained in the original process, and we do not restrict the experiment to only embed the chosen words, as this would create an unrealistic environment. This is also the reason why the maximum similarity for "Ransomware" varies from 0.63 to 0.96, sometimes Wannacry is added to the vocabulary before "Ransomware", and consequently takes part in deciding position of the word, and sometimes "Ransomware" is embedded first. This level of randomness is acceptable in this experiment, and allowing a word to be embedded as early as possible to contribute to other embeddings by adding them to the vocabulary is convenient here. If it is problematic to do so in other cases, batch updates would be equally simple to implement.

Any further quantitative evaluation is not sensible, as one word may be more fitting to achieve a high maximum similarity, if a synonym is present in the embedding vocabulary, and another may be more fitting to achieve a low maximum similarity, if it has few related words in the vocabulary. These types of examples can be made for each of the statistical features. Noting the intuitive quality of some of the most important features for word embeddings, we thus choose to visualize the embedding for the newly added word in a reduced dimensional space, to consider the neighbouring words in relation to the embedded word.

In figure 6.13, the embedding space is reduced to 2 dimensions so that we can visualize the 30 closest words to the new embedding "Ransomware" in the embedding space, when updating with window size 3 and count threshold 500. We see that many of its nearest neighbours are indeed words we would expect to be related to a virus attack, such as the hacking methods "phishing" and "ddos", and the virus software related "malware". We also see many acronyms within the fields of network protocols, computer science and information security mentioned. This is not unexpected, and appears to replicate the original word embedding characteristic of similar words, where words with similar meanings are very close and the area surrounding a word will be related to similar subjects. Looking at A.6, the increase from threshold = 500 to threshold = 1000 appears to have little impract.

In figure 6.14, we can see the same dimensionality reduced presentation of the 30 closest words to the new word embedding of "Brexit", with window size 3 and count threshold 500. The surrounding words are expected to be related to themes such as politics, the UK, immigration and voting. Though some of the words observed in

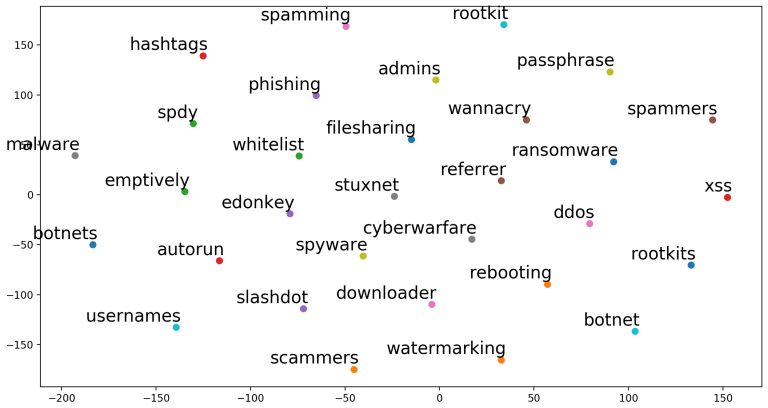

Figure 6.13: Visualisation of 30 closest words to Ransomware after dimension reduction, with parameters window size = 3 and threshold = 500.

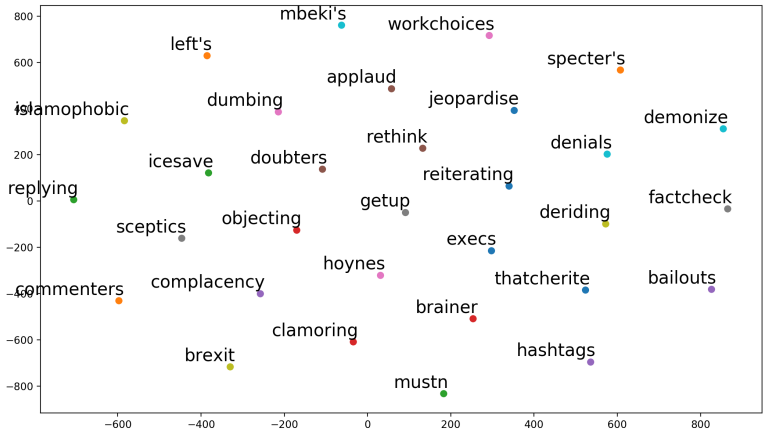

Figure 6.14: Vizualisation of 30 closest words to Brexit after dimension reduction, with parameters window size = 3 and threshold = 500.

the figure are easy to see fit into these themes, many more than with "Ransomware" are less apparent. Some of the words are also neither obvious nor unnatural to see in this context, such as "sceptics" and "islamophobic", as the debate around "Brexit" was polarized and touched on topics such as racism. Thus, the placement may be appropriate, but is clearly less obvious than for "Ransomware". This does not have

to be an indication of deviating behaviour, as many words will vary in quality and ambiguity in the original training process as well, but does warrant caution.

A window size of 3 is chosen for further experiments, to capture more direct context since the word embedding is aiming to discover the word meaning itself, and later topic embeddings will cover the discovery of topical connections. This also coincides with the window size run on the original PSDVec, which is an advantage when building on this.

In the original word embedding by PSDVec, choosing the 180 000 most frequent words from the chosen Wikipedia Corpus of ~ 2 billion words leads to a word count cutoff just above 7000. When running these experiments, a significantly lower threshold for being added is chosen, both due to a smaller corpus and because we want to make sure words are added so we can see if there is any effects.

Word Embedding Updates on Experiment Data Sets

In table 6.16, we see that a high number of unique words are not included within the vocabulary, which is not too surprising after observing the word count percentiles in table 6.1. This can be caused by a combination of reasons, such as frequent use of slang in online forums, which can be underrepresented in the vocabulary, and misspellings. For instance, one word with count equal to 1 in the Reddit vocabulary is "weight" misspelled as "wieght". The percentage of words with a substantial occurrence count is also quite low. This too can indicate tendencies such as one-off spelling mistakes, or just very specific terms. Many words which are not contained in the original vocabulary with high counts in the data set, are bigrams of words that do exist in the vocabulary, and may lead to little information gain, a mere consolidation of current word embeddings, or noise in the vocabulary. Some bigrams could however provide new information, through adding new meaning or relations between words within the vocabulary.

The News Aggregator data set has a larger corpus than Reddit data set, as well as higher word counts. At a first glance, it appears that many of the high count support words in both Reddit and Newsaggregrator are composite bigrams consisting of combinations of words which are in the vocabulary, such as references to the movie "Captain America" represented as "captainamerica".

Having observed the stability of statistical properties across thresholds and window sizes, we set the threshold to 500 word counts before being added to the embedding, in order to ensure that words are added. In addition to this threshold, we require at least 1000 of the context words to be in the vocabulary. This additional threshold is necessary due to the frequency at which words are not present in the vocabulary, leading to the risk of embedding a word without sufficient context to make a semantic meaningful embedding. The reason for choosing this method instead of removing words out of vocabulary and force a word window from only the relevant words, is that this would counteract the semantic consequences of the chosen window size, by introducing words that in true context would be further away from the focus word. With these parameters, the number of words added is listed in 6.17

The 30 words added when running this updating scheme, along with their top

Training Corpus	Reddit Posts	Reddit Titles	News Aggregator
Word count out of vocab	1 120 422	96 995	611 235
Unique words out of vocab	110 498	5 521	45 688
Word count percentiles for words not in Vocabulary			
20th	4	4	4
40th	5	5	6
60th	6	7	8
80th	11	20	13
90th	17	33	22
95th	28	92	37
99th	81	144	122
99.5th	125	235	200
99.8th	221	396	358
99.9th	330	453	479
99.95th	520	857	654
99.99th	952	1 098	1789

Table 6.16: Support count percentiles for words outside of original vocabulary.

5 closest words, can be found in A.15. A qualitative evaluation indicates that words are embedded in a contextually sensible manner. The 57 words added when running on the Newsaggregation data set is listed in A.16.

Data set	# Words added
Reddit	30
Newsaggregator	57

Table 6.17: Number of words added when updating embeddings with word mention threshold = 500 and context word threshold = 1000.

Most words added are bigrams in both cases, many composite of words which are already in the vocabulary, such as "Kardashian" and "Kanye" which may only consolidate their position and connection to each other. Some additions however, such as "climate change", are adding meaning beyond their separate words which are likely not to be excessively correlated in the original embedding. There are also non-bigram words which are added, such as "heartbleed", a reference to a security vulnerability in a cryptography library[1]. The statistics for the words added in both experiment data sets can be found in A.17. We can see that the statistics are still within expected values.

The following section presents some experiments on the effect of these updated on the topic embeddings, after introducing the model and initial runs.

[1] http://heartbleed.com/

6.3.2 Derived Topics

While TopicVec supports category-wise training, where a set number of topics can be elicited from each defined category in the data set when spesicific categories are pre-defined, we opt for the non-category based training to better imitate the realistic setting of news recommendation. Category labels may be erroneous or too generalizing, especially in online forum environments such as reddit.com, and the most important goal of the topic embeddings is not to perform well in the classification task in our case. Also, with the uneven distribution of categories in the training set, forcing TopicVec to discover the same amount of topics in each may lead to illogical or non-informative topics. Finally, as we are not aiming to use the model generated for classification tasks, but rather discover common topics across the labeled categories, attempting to train within them seems counter intuitive. Embedding methods often do qualitative evaluation, or test for sentiment and classification tasks, but this is not our main focus. Choices for the number of topics, t, are chosen to be comparable to LDA and provide a sufficient span to observe trends. This is because of the topic embedding setup being less specialized for tasks producing metrics such as perplexity. All TopicVec runs use the parameters described in 6.18.

Parameter	**Value**
Dimensions N	500
Max iterations	300
Dirichlet parameter for topics α_1	0.1
Initial learning rate δ	0.1

Table 6.18: Parameters for TopicVec.

Looking at the output topics and percentages of words connected to them from TopicVec with t=5 in 6.19, 4 fairly large topics can be seen. Though this matches with the number of predefined Reddit categories in the data set, we see that both Topic 2 and Topic 4 would intuitively belong in an "Entertainment" category, while it seems that "Business" and "Technology" from the original labels have been merged into one topic. This is not too surprising, as the entertainment category is shown to encompass over 40% of the training set documents in 6.3 and the model will likely deem modeling these topics well important in order to maximize the corpus likelihood training goal. The last topic, Topic 1, is less clear. It contains several geographic references, and may be related to geography. Topic 1 has significantly less words represented in the corpus than the others. While the two Entertainment topics combined have $\sim$ 57% of the vocabulary words associated with them, the health and fitness topic, Topic 5, has $\sim$ 31% and the merged business and technology topic has $\sim$ 21%, Topic 1 has less than 1% of the words associated with it.

The top 10 words for the 10 largest topics for t's 15, 40, 80 is described in

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
0.9%	23.4%	21.00%	23.8%	30.9%
th	movie	business	song	lift
jersey	watch	google	rock	workout
national	film	service	music	gym
world	trailer	market	band	muscle
south	star	online	pop	squat
north	tv	company	indie	body
century	review	app	hiphop	eat
cup	episode	loan	hop	exercise
st	scene	technology	video	start
city	character	mobile	cover	fat

Table 6.19: Top 10 terms and total percentage of words associated with topics found by TopicVec with t=5 on Reddit Posts.

A.18, A.19 and A.20. As the number of topics found increases, the percentage of words associated with each topic naturally decreases. Some topics still appear quite clear for these runs as well, but we can see that more topic mixes show up and that several topics become more specific. Doing a qualitative evaluation of the extracted topics, $t = 40$ appears to provide the clearest topics, with $t = 15$ having some strange mixtures in for instance Topic 8 and $t = 80$ generating confusing mixtures in for instance Topic 64.

With t=80, many topics have a very low percentage of words which holds alloted to them, down to 0-0.1%. This doesn't have to mean they're poor topics, but they are often very specific and may have less support in the corpus due to rarer occurrences. An example of such a topic, which is clearly about ancient Egypt and customs, is shown in 6.20. This appears to be a well-defined topic, but will be present in the corpus very infrequently compared to for instance music-related topics. While these smaller topics derived when we have higher values for t may be good topics, the skewed size of word-topic percentages in the corpus may affect the clusters created. With a smaller number of clusters or high thresholds for cluster sizes, documents with high proportions of small support topics may be forced into clusters which they have little in common with in kmeans or become noise points in HDBSCAN. If the number of clusters is large however, it becomes more likely that documents with high proportions of these very specific topics may generate very small clusters together. This is not necessarily as problematic, since the intention is not to create rich clusters from which we can elicit many items to recommend, but to recommend new articles that are being connected to the cluster at a later time.

Considering the fact that the training set is derived from categorized data within a few topics, common words and topics related to these four main categories are likely to dominate the topic modeling both in amount and when it comes to the word-topic allocation. This skewedness of words connected with large topics can mean that documents which contain mainly small, or rare, topics will struggle in

Topic 36
0.1%
egyptian
ancient
pazar
burroughs
matteo
pyramid
famous
tomb
orthodox
ambani

Table 6.20: An example of a topic with low word allocation count.

this specific training environment. There is also a risk of some common words being associated strongly with the most common topics, leading documents which are not related to the specific topic to get a slight topic proportion. This will likely mostly be an issue in very short documents, where other words do not have the opportunity to counteract this tendency. While very short documents is a frequent issue in this testing environment, it is likely less of an issue with full-length articles, though still present.

In 6.21, we can see that performance on the categorization task TopicVec was originally trained on, improves with higher values for t. The macro average will be most important in this case, because of a few larger categories dominating the data set. Macro average does not aggregate all evaluation results equally, but calculates performance within each category in order to avoid deceiving classification results which favours a dominating cluster. We see that the macro average generally achieves slightly lower values, which can indicate that some categories have poorer classification performance than others. This is likely the smaller category groups.

A finer granularity of topics can be better for the classification task, but for each increase in topic numbers, a dimension increase incurs for all user models. Training and inference will also be more demanding when increasing the number of topics. Thus, the optimalization of the classification task cannot be our main goal. Especially for the Reddit data set, where users may easily post unmoderated content which is not directly connected to the given category, leading to mislabeling. This leads to difficulties in evaluating the topic embeddings for the purpose set for this thesis.

Topics Derived from Updated Word Embeddings

With the limited time frame and corpus in our testing environment, as well as the set thresholds and known word count percentiles, changes in word embeddings were relative minor and any significant variations in topic quality derived from the

Topic #	5	15	40	80
Iterations for best run	**64**	**109**	**213**	**234**
Train precision micro average	Recall: 0.693 F1: 0.693 Accuracy: 0.693	Recall: 0.734 F1:0.734 Accuracy:0.734	Recall: 0.790 F1: 0.790 Accuracy:	Recall: 0.804 F1: 0.804 Accuracy:0.804
Train Precision macro average	Recall: 0.555 F1: 0.536 Accuracy:0.693	Recall: 0.620 F1: 0.619 Accuracy:0.734	Recall: 0.718 F1: 0.737 Accuracy: 0.790	Recall: 0.739 F1: 0.757 Accuracy:0.804
Test Precision micro average	Recall: 0.742 F1: 0.742 Accuracy:0.742	Recall: 0.790 F1:0.790 Accuracy: 0.790	Recall: 0.839 F1: 0.839 Accuracy:0.839	Recall: 0.847 F1: 0.847 Accuracy: 0.847
Test Precision macro average	Recall: 0.612 F1: 0.592 Accuracy: 0.742	Recall: 0.683 F1: 0.685 Accuracy:0.734	Recall: 0.786 F1: 0.797 Accuracy:0.839	Recall: 0.793 F1: 0.804 Accuracy:0.847

Table 6.21: Performance on classification task for topic embeddings across different numbers of topics on original word embeddings.

updated vocabulary is not expected. Especially since many of the words added are combinations of words which already did contribute to topic embeddings.

Looking at table A.21, the results are very similary with some minor improvement in performance on the classification task when running on the updated embeddings.The number of iterations performed in training, varies between fewer and more than the original results. With the non-deterministic nature of the TopicVec algorithm, it is difficult to reason whether these minor improvements are a result of the updated word embeddings or just a different initiation and inference process. Even if this is the case, with the previously mentioned tendency for new words added to be combinations of previously existing words, the addition may only have aided in consolidating already existing information instead of providing new insights.

Topics Derived from News Aggregator

Comparing the topics derived from the News Aggregator data set for five topics in 6.22 to the Reddit data set in 6.19, it appears like both have split Entertainment into two seperate topics. For Reddit, this becomes music and movies, while News Aggregator appears to have divided it into celebreties, and movie and films. A surprising result is the similarity of the smallest topic, Topic 1 in both tables. This may indicate that while the topic qualitatively seems strange, it is useful in the context of both data sets. An important difference is the News Aggregator data set managing to find two seperate topics for business and technology, namely Topic 4 and Topic 5. The business topic, Topic 4, appears related to economy and stocks, but also contains some words related to health. This could be a reflection of the News Aggregator data set containing more overlapping terms and interests between

these categories, such as pharmaceutical companies' stock price development, and that the behavioural patterns and types of subtopics discussed in the casual forum setting deviates too much from the characteristics of formal articles written on the topic to compare the two.

Also for larger values of t, the topics discovered from News Aggregator appear more well-defined than those derived from Reddit, based on a qualitative consideration, as can be seen in Appendix A.22 and A.23. This can be an indication of less noisy data in the News Aggregator data set.

Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**
0.9%	21.3%	20.1%	32.0%	25.7%
county	kardashian	movie	rate	google
church	kim	game	stock	apple
th	kanye	throne	rise	samsung
north	miley	film	ebola	microsoft
school	cyrus	star	recall	galaxy
city	bieber	season	gas	facebook
south	beyonce	trailer	price	buy
university	justin	review	risk	android
time	wed	dy	health	million
st	award	watch	china	ceo

Table 6.22: Top 10 words associated with TopicVec topics found with t=5 on the News Aggregator data set.

6.3.3 Topic Embedding Clustering

As with LDA, the topic embedded documents are clustered with a variety of parameters with k-Means and HDBSCAN. Values for k and t are decided with comparison to the LDA model and the perceived quality of the topic embeddings. Results follow in this section.

k-Means on Reddit Posts Data Set

A collection of all k-Means results can be found in A.24. In figure 6.15 and 6.16, we see 3-dimensional plots of five clusters created with five topics embeddings and with the original category labels, respectively. It becomes apparent that the clustering is not replicating the original categories, but defining its own document relations. Seeing the distribution of documents across clusters for $t = 5$ and $k = 5$ in 6.23, and these plots, it appears that the purple cluster with label 4 is heavily dominant within this clustering. This cluster has 256 650 of the 292 270 total documents, compared to the largest true category label which has 98 955. The tendency is confirmed across several k's for several numbers of topics in A.7 and A.25. This can indicate that while topic dimensions change, the topics elicited combined with the vector attributes creates similar relationships between documents, or that some topics is overly dominating in the dataset.

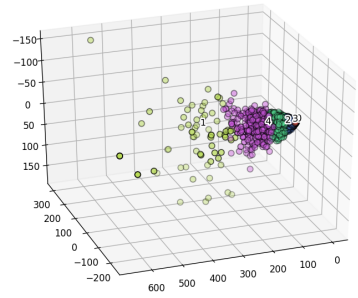

Figure 6.15: 3-dimensional visualization of k-Means clustering for 5 topic embeddings with 5 clusters from the Reddit Posts data set, reduced with PCA.

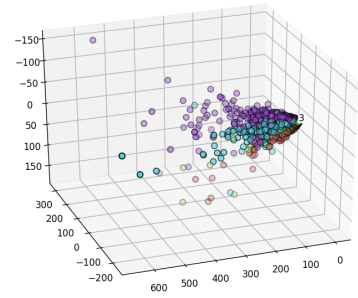

Figure 6.16: 3-dimensional visualization of the Reddit Posts category labels, reduced by PCA.

Figure 6.17: Word cloud for the largest cluster clustering topic embeddings with 5 topics into 5 clusters.

Figure 6.18: Word cloud for the largest cluster when clustering topic embeddings with 40 topics into 5 clusters.

t	Size of clusters				
5	256 650	17 844	3 795	575	72
15	256 930	17 617	3 770	554	65
40	256 803	17 776	3 741	551	65

Table 6.23: Cluster sizes for $k = 5$ with k-Means on Reddit Posts with TopicVec, across values for t.

In addition to sharing similar cluster sizes and shapes, $t = 5$ and $t = 40$ with five clusters share some most frequent words within the clusters, observed in 6.17 and 6.18. These word clouds corresponds to the most common words within each cluster scaled in size by frequency and with the colour mapping to the colour of the cluster in their respective clustering plot in the reduced space.

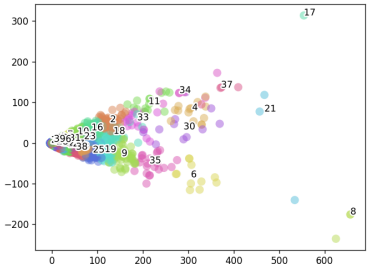

Figure 6.19: k-Means clustering run with 40 clusters on documents with 5 topic embeddings from the Reddit Posts data set, reduced with pca.

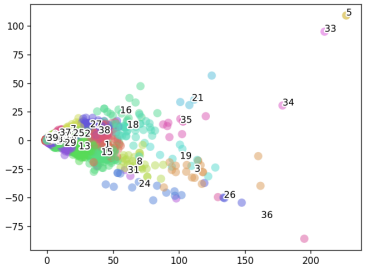

Figure 6.20: k-Means clustering run with 40 clusters on documents with 40 topic embeddings from the Reddit Posts data set, reduced with pca.

The leftmost cluster in 6.15 and A.7 appears to have a lot larger distances between documents than the other clusters, but it is important to remember that the dimensions have been reduced and may not provide an entirely realistic view of document distributions. The other clusters also appear to be grouped closely together. This could be a contributing factor in the drastically declining performance of cluster metrics for higher values of k.

When looking at $t = 5$ and $k = 5$, a natural question becomes whether the clustering is merely clustering together documents containing one topic, possibly diminishing the possibility of gaining information about composite patterns. Looking at the training set documents represented by five topics from embedding, the average gini index for all documents is 0.72. In the sense of purity, this deviates from the optimal value of 0, and it is thus unlikely that all the clusters generated are merely a reflection of pure topics. If this had been the case, interesting documents could probably be more directly inferred without much impact on quality.

Further experiments shows that higher values for k, as expected, reduces the size of the dominant cluster. In combination with higher values for k, higher numbers of topics tend to have higher maximum size values for clusters than lower values for k. This can be seen in A.28.

When looking at high values of k, such as in 6.19 and 6.20, we see that the document points which appeared to be somewhat misplaced in the right half of the plot have been assigned to their own, smaller clusters. These clusters appear to be very small, but will likely achieve relatively high metric values, due to being very well-seperated from the other documents. The remaining clusters, collected on the left side of the plot, may achieve higher cohesion values and low seperation values based on the visual representation in these plots.

The evaluation metrics for the clusterings is the Silhouette coefficient an C-H-index, as in LDA. The metrics achieve better values than expected. This could be because of the large amount of very short texts, which may be achieving too

Data set	Distance	t	k	Silhouette	C-H
Reddit	Euclidean	5	3	0.87	231 858.39
Reddit	Euclidean	5	4	0.83	231 858.39
Reddit	Euclidean	5	5	0.77	231 858.39
Reddit	Euclidean	5	40	0.28	141 592.09
Reddit	Euclidean	40	5	0.73	141 552.61
Reddit	Euclidean	40	40	0.20	45 341.13

Table 6.24: TopicVec Reddit Posts k-Means runs selected for prediction.

high similarities. This could also have an impact on the drastic cluster metric performance decline for high values of k.

In general, small values for both k and t provide the best clusters in terms of evaluation metrics. The decrease in evaluation metric values for the same value of k across different values for t is the most significant for t <10, and is otherwise small. Increasing the k has a significantly larger impact on the evaluation metrics, but it is important to keep in mind that the C-H−index penalizes higher values for k.

The plots on the previous pages can indicate that many documents are close to each other in the embedding space, and this could lead to poor separation in clusterings with higher k values. It is possible that the Silhouette coefficient is penalizing higher values for k due to this, as it is separating documents which are relatively close to each other into different clusters.

A relatively drastic drop in the Silhouette coefficient happens across all values for t at number of clusters $k > 10$. The C-H-index also normally makes a larger dive between 5 to 10 and 10 to 20, than between other measured points. As an indicator that the Reddit data set prefers few clusters, this could be a reflection of the limited number of topics present in the corpus, but this cannot be an answer in itself, because Silhouette and C-H values continue to increase even for k values lower than the number of coarsest category labels.

While low values for t and k are preferred by the clustering metrics, one would intuitively expect clusters with higher values of k to find more composite and nuanced clusters. Also noting the indication that the metrics may mostly be penalizing splitting similar clusters, and that $t = 40$ provided qualitatively good topics, we choose a a variety of t and k values to consider for prediction, presented in 6.24.

k-Means after Updating Word Embedding

Looking at 6.25 compared to previous results for Reddit Posts clustering without word embedding updates, there are not enough changes in the clusterings created by the updated embedding vocabulary to indicate more than a arbitrary difference in clustering quality due to the non-deterministic, random factors involved of the topic training and the slight solidification of corpus by adding a few more words. This may also be influenced by the fact that most the words are added in the category of entertainment, within celebrities and movies, which is already a very large portion of the documents and may have little impact on their topics. The updating thus

appears not to be useful enough at the scale and within the short time span of the data set collection in this experiment to warrant much further consideration. Further experiments could be conducted to determine effects over longer periods of time, where more entirely new terms are likely to be added. In these experiments, additional functionality would be required to allocate the words added after initial training to the most relevant of the existing topics. Otherwise, the new words would not be allowed to impact the topics discovered. These tendencies are confirmed with the News Aggregator dataset in A.27, after updating the vocabulary by 57 new embeddings only one of the clusterings change metric values.

k	Silhouette Coefficient	C-H Index
3	0.87	242 312.22
4	0.83	247 502.69
5	0.77	241 087.22
10	0.66	196 067.67
20	0.34	172 221.59
30	0.32	158 656.79
40	0.29	150 328.36
50	0.25	142 076.72
70	0.24	131 058.79
80	0.22	126 364.71
100	0.22	117 076.49

Table 6.25: k-Means on Reddit Posts with t=5, after word embedding is updated by 31 words.

k-Means on News Aggregator Data Set

A few experiments were run on the News Aggregator data set as well, in order to see how Reddit Posts compared to a data set more modeled on real-life news. Looking at A.26 and A.27, News Aggregator maintains more realistic values for Silhouette coefficients and C-H-value maintains, as well as favouring k equal to the predefined number of categories in the run for $t = 5$. This could be an indication of the News Aggregator data set being less noisy in terms of the categories, as they appear more easily defined. Other than this, the News Aggregator clusters also appear to strongly favour low values for both t and k.

t	Size of clusters				
5	113 103	85 595	77 915	59 268	3
40	131 694	104 636	78 820	20 731	3

Table 6.26: Cluster sizes for $k = 4$ with k-Means on on News Aggregator with TopicVec, across values for t

Seeing the size of the clusters in 6.26, we see that the clusterings still prefer some very large clusters and some excessively small, but there is also a much

better balance between cluster sizes than with Reddit Posts data. Looking at the 3-dimensional visualization of the clustering in 6.21, the choice of the cluster of size 3 across values for k and t seems like a natural result of the three outliers at the left of the figure which composes the cluster with label 1.

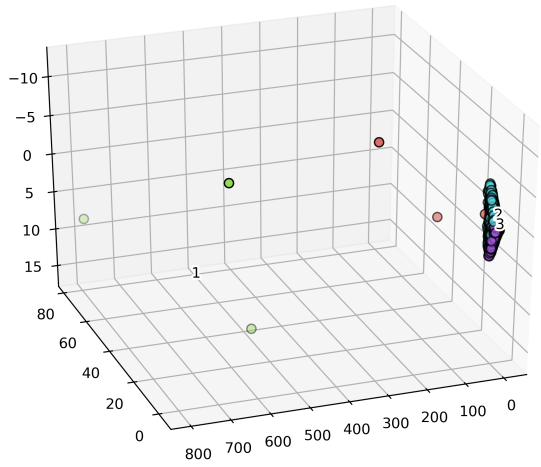

Figure 6.21: News Aggregator documents represented by 5 topics, divided into 4 clusters by k-Means.

HDBSCAN on Reddit Post Data Set

An excerpt of some of the best HSBSCAN results running on topic embedded documents for topic numbers 5 and 40, can be seen in 6.27 and further results can be found in A.29. General trends show that, as expected, smaller values for minimum clusters creates more clusters and higher restrictions on minimum samples requires leads to more outliers. One strange thing is the C-H-index having some better values for runs where the number of clusters is higher, though the Silhouette coefficient drops between the same runs. An increase in topics also leads to severe decrease in number of clusters and increase in outliers for the same running parameters.

HBSCAN did not however not perform well, especially for the intended use case. It generally creates many clusters with very few members and the excessive amount of outliers would negatively impact the clusterer's ability to make useful predictions, as most documents the user has interacted with before is likely to be contained in the outliers. With k-Mmeans allowing for clusters of size 1, we test for very low values of min samples and cluster size. Though the lower values for minimum samples and cluster sizes somewhat improves the Silhouette coefficient, the C-H-index is lower than some other combinations, probably due to the algorithm creating very many clusters which is penalized by this metric. There is still a good 40% of the documents are being labeled as noise. Taking this into consideration,

the HDBSCAN is not evaluated further in the context of topic embeddings.

t	*s*	*c*	Silhouette	CH	# Clusters	# Outliers
5	1	5	-0.35	8.89	17 924	121 005
5	1	10	-0.52	14.02	5 549	146 961
5	10	20	-0.70	13.53	301	194 718
5	15	20	-0.69	15.09	233	201 502
5	20	20	-0.69	13.9	206	207 558
40	1	5	-0.84	1.01	7 615	212 970
40	5	10	-0.97	0.99	744	259 144

Table 6.27: Best HDBSCAN results on Reddit Posts data set with topic embeddings. Min samples is denoted by s and min cluster size by c.

6.4 Prediction Results

A comparison of the prediction results are illustrated in figure 6.22, followed by details about each model in table 6.28.

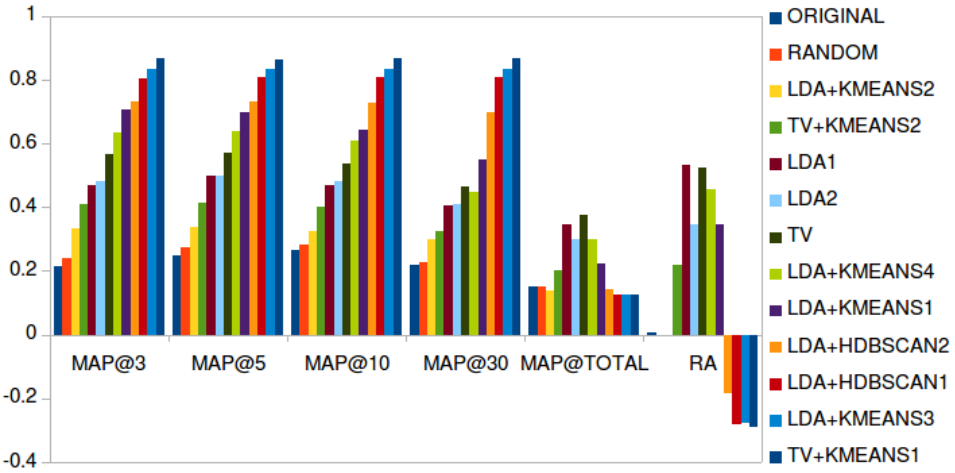

Figure 6.22: Prediction results. Details in table A.30 in appendix.

As can be seen, all models utilizing topic modeling, either through TopicVec or LDA, performs better than random predictions, and predictions using the original coarse categories. Looking at the MAP values, the two naive models are closely followed by one of the predictors utilizing k-Means in combination with LDA, namely LDA+KMEANS2. This is one of the models chosen quantitatively, being among the three best performing with respect to C-H index and Silhouette Coefficient. Despite

Model	t	Cluster Parameters	Training Set
LDA1	4	NA	Reddit
LDA2	15	NA	Reddit Title
LDA+HDBSCAN1	100	s=10, c=15	Reddit Title
LDA+HDBSCAN2	15	s=5, c=5	Reddit
LDA+KMEANS1	4	k=4, distance=euclidian	Reddit
LDA+KMEANS2	15	k=4, distance=KL	Reddit Title
LDA+KMEANS3	100	k=5, distance=euclidian	Reddit Title
LDA+KMEANS4	5	k=10, distance=euclidian	Reddit
TV	5	NA	Reddit
TV+KMEANS1	5	k=3	Reddit
TV+KMEANS2	5	k=4	Reddit
ORIGINAL	15	NA	Reddit Title
RANDOM	15	NA	Reddit Title

Table 6.28: Description of compared models displayed in figure 6.22.

this, it is outperformed by qualitatively chosen LDA+KMEANS4, as well as other models with either or both of worse Silhouette and C-H score. In table 6.28 one can see that the model is the only model using KL divergence rather than Euclidian distances in the training and prediction, which may indicate that KL distances between documents might not be representative, propagating in non-representative user models and clusterings. This could also explain some of the experiment results presented earlier (A.2 and A.3), where the entertainment category generated by the same model seems to partially or fully leak into all categories.

Many of the other models have a notable better precision when only the upper parts of the ranked lists (top 3, 4, 10, and 30), but when evaluating performance on the full list, most of them have excessive drop in performance. This is especially evident in the four rightmost, LDA+HDBSCAN2, LDA+HDBSCAN1, LDA+KMEANS3, and TV+KMEANS1, dropping at MAP@TOTAL and RA. As mentioned earlier, it is important to be aware that precision alone is not a optimal metric for this case, and as MAP only sums over correct predictions, the scores may be deceptive. Thus, the RA metric and supplementary methods with promising qualitative characteristics are considered in order to gain more insight about the results.

Figure 6.23 illustrates where relevant documents most frequently are positioned in the rankings. In the model one can observe that all four models previously referred to shares a rather similar pattern, with a centre of gravity below the 50th percentile. One can also observe that very few documents are recommended within the 30th percentile apart from in the very top, causing the evenly high values observed in figure 6.22. It is important to note that both models utilizing HDBSCAN was proven to classify exceedingly large portions of the documents as noise, as seen in 6.2.2, making them useless in a real scenario as many articles will fall outside of the recommendation scope. Nonetheless they are included in the evaluation in order to get some notion of their potential.

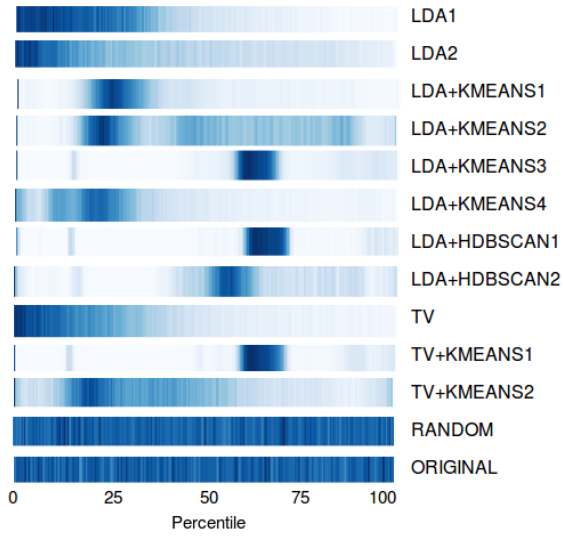

Figure 6.23: Heat maps for where in the list the expected documents are positioned in the ranking. Darker color indicates higher occurence of expected documents at this position.

The dense centres of documents observed in the discussed models are somewhat present in several other models' rankings as well. To get a better understanding of this, an aggregation of document clusters is illustrated in figure 6.24. The diagram shows an aggregated ranking for LDA+KMEANS2, where all user rankings of documents are aggregated on each position. Documents of Cluster 1 is in general ranked low, but the predictor manages to rank a great portion of the relevant documents from the other clusters towards the upper positions, but fails near top 40. The documents predicted in this area does however seem to be of the clusters mostly favored, but as the threshold around 40 seems to be a general trend, it may be an indicator that around 40 key documents in these clusters dominates and are too highly scored by the preference function. Another indicator of the preference function being too discriminative is the smaller Cluster 1 being neglected.

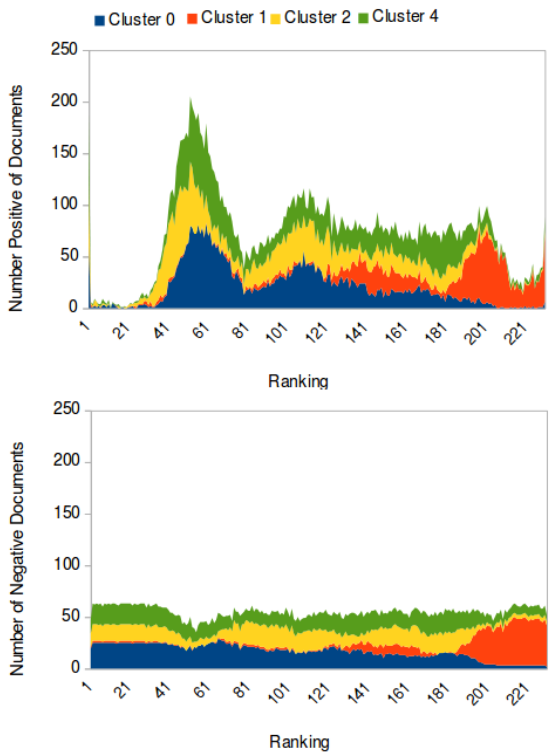

Figure 6.24: Aggregated documents, denoted by cluster, in the ranked lists for LDA+KMEANS2.

The remaining six models performs significantly better than the rest at RA, and also have a decent MAP performance, despite lower values due to more documents found in the upper part of the ranking, illustrated in the heat map (figure 6.23). TV+KMEANS2, LDA+KMEANS4, and LDA+KMEANS1 all manages to position the relevant documents higher up in the ranking, however, looking at the heat maps and the overall MAP and RA, they are outperformed by the purely topic based models, LDA1, LDA2 and TV. As these predictors are based on user models solely built by averaging topic vectors in each users document collections, it is interesting to see how number of topics affects the user models and predictions. Figure 6.25 shows the Gini indices and trend lines for user models, ordered by the users' AP and RA. Being placed far right in the figures indicates that the user has lower average precision score, and thus are more difficult to make good predictions for. The trend for both predictors is that the users with broader range of interests topically is harder to make good recommendations for.

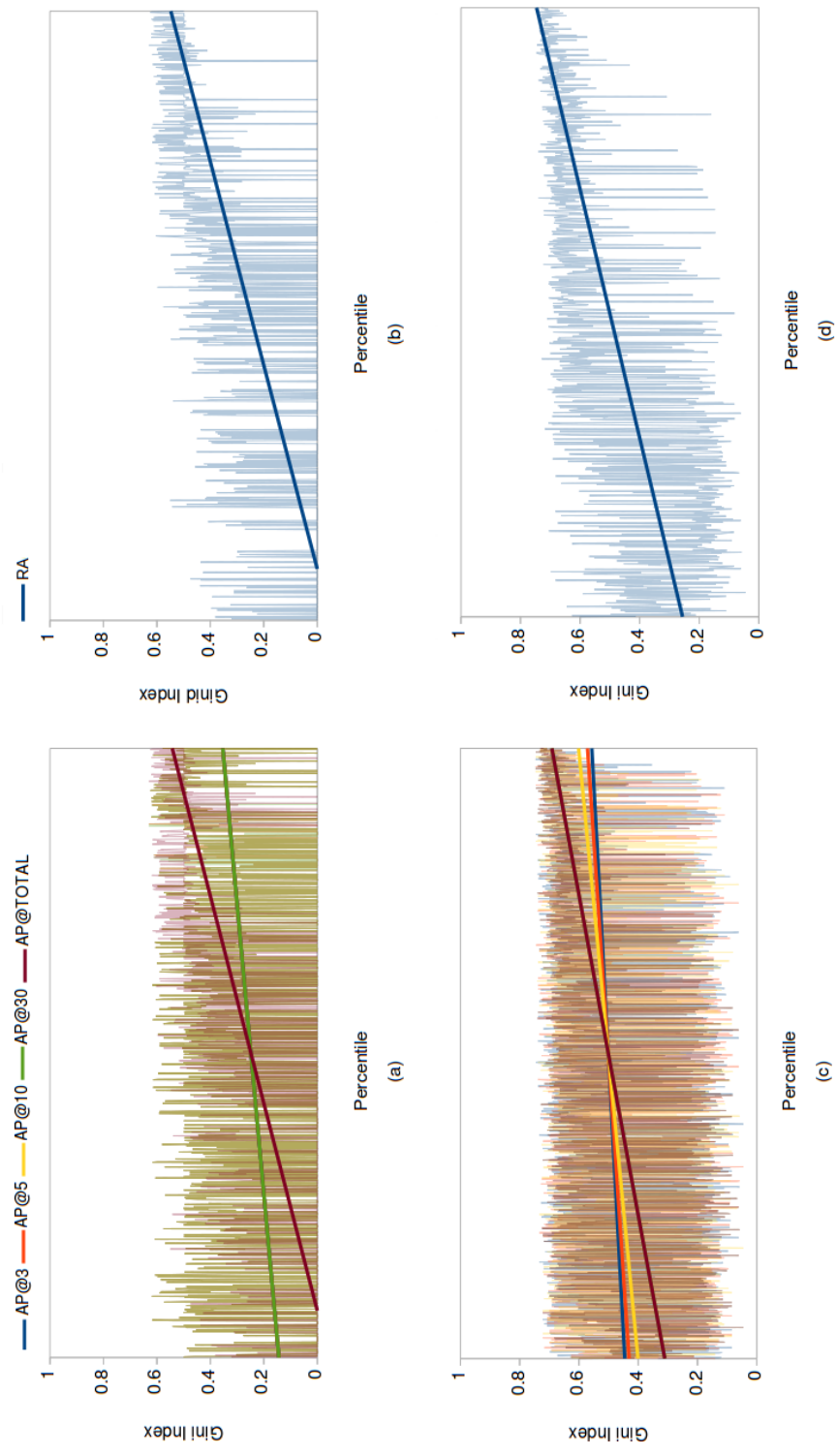

Figure 6.25: Gini indices and for user models ordered by MAP and RA for (a)(b) TV with 5 topics (c)(d) LDA1 with 4 topics. (Some lines are overlapping as a result of same documents found at the positions.)

Still, no general combination of topic extraction and clustering stands out as a obvious choice, as both algorithms and topic extractions methods yield varying results depending on parameters. Though, on specific parameters some of them do seem promising. As just discussed, the purely topic based models do have potential, but there are also well performing models incorporating clustering. LDA+KMEANS4 and LDA+KMEANS1 both have an acceptable performance, and might have the advantage of recommending a broader range of relevant documents to supporting exploration, although this might affect the scores negatively as confirmation or revoke of false positives and negatives is inapplicable. However, tuning the parameters must be done cautiously, as the ones resulting in best predictions not may be representative for the general selection process. LDA+KMEANS4 was one of the models filtered out based on C-H and Silhouette coefficient, but picked manually from qualitative evaluation, indicating need for more sophisticated ways to evaluate the clusterings quantitatively.

The two best performing TopicVec models with respect to MAP and RA are presented in 6.22. The two models are very different, with TV+KMEANS1 achieving a high value for MAP with 5 topics and k equal to 3 and the TV+KMEANS2 achieving a relatively high value for RA with 40 topics and 40 clusters. The first model was chosen due to its best performing clustering metrics. Though in line with the clustering metrics, it is somewhat surprising that k=3 performs the best as it is below the expected number of found categories, as well as the number of well-defined topics found with 5 topic embeddings (4).

The second topic embedding model, TV+KMEANS2, achieved bad cluster metrics values, but was chosen despite of this, due to the qualitative attributes of the TopicVec model at $t = 40$ and to assure sufficient span of values k to observe trends in clusterings. Looking at 6.23, the topic embedding models also display the behavioural tendency of high MAP values indicating a few good predictions early in the list, while the bulk of relevant documents are placed in the lower half of the recommendation list. Despite gaining considerably worse metrics for both clustering and MAP during prediction, the TV+KMEANS2 qualitatively appears superior. The model has a significantly larger portion of the relevant documents presented in the first half of the recommendation list. As with several other models seen in 6.23, TV+KMEANS2 has an area between the low percentile and $\sim 10 - 15$ percentile which appears to have considerably less relevant recommendations.

Looking at some additional prediction results from topic embedding models, found in A.31, a general pattern of increasing k's leading to decrease in MAP and increase in RA can be observed. This could be an indicator of higher values for k providing better user models, as intuitively expected, and that the clustering metrics are merely favouring clusters which does not facilitate more nuanced user models.

6.5 Discussion Summary

This chapter has presented and compared experiment results in detail, introducing findings, challenges, and subjects for improvement. We have employed traditional,

scalable clustering methods for detection of topical communities on documents modeled by traditional LDA and topic embedding, in order to provide nuanced user models for improvement of recommendations, especially for newly introduced items.

One of the main objectives, and challenges, throughout the experiments has been optimizing unsupervised training models across different sub-goals to meet the prediction end goal. Parameter choices become difficult, as available metrics may not provide a correct picture of the quality of the results when moving forward to the next step, or appear contradicting. One challenge observed, is eliciting the best topical clusters for the explicit purpose of user recommendation. And in turn, the learning objectives for topic embeddings and topic modeling may not provide the optimal results for the purpose of clustering either, especially for topic embedding which is traditionally tested on tasks such as classification. The alignment of goals across technologies in the different steps may be complicated, and discerning the optimal solutions for the next step while relying on unsupervised training can be difficult. When using traditional topic modeling, selections of lower-quality communities sometimes yielded better prediction results, highlighting that community detection can prioritize qualities which are not appropriate or optimal for prediction. This was also observed when using topic embedding, where some models with qualitatively chosen topic embeddings combined with lower-performing clusterings yielded prediction results that are preferable in a news recommendation case, compared to models with more optimal clustering metrics.

Other challenges emphasized during the experiments, are the variety of difficulties faced concerning data set limitations. Preserving the imbalanced category distribution to better mirror the real news domain seemingly has negative effects on some of the clusterings where the large entertainment category dominates, and smaller categories, like health, are sometimes not represented. At the same time, we see that the topic modeling manages to split these large categories into subcategories, and undersampling larger categories to obtain balance in the data set introduces the risk of weakening or removing such subcategories. Performance changes in relation to the number of topics elicited were not consistent across the topic modeling methods either, indicating that optimal training sets may vary in size and category distribution, making it difficult to make such adaptions.

Observations were also made comparing the language in the official news domain to internet forums' less formal nature. Despite discussing the same topics and having forum specific stop words removed, some notable deviations were found, which may lead to extra noise in the data. We see that despite the News Aggregator data set being larger than the Reddit data sets, the latter has almost twice as many unique and total words outside of the word embedding vocabulary than the first many with very low support count. While some may be specific terms just not contained in the original word embedding, many are likely slang terms and misspellings not captured in the forum stop word removal. These are words that will not be of use in the topic embedding, and many will likely become noise in LDA too. Reddit also has many very short texts and very frequent use of words such as "help" and "thanks" in posts of people asking questions etc, becoming a

dominant feature here while being less relevant for the news domain. In addition, the appearance threshold for infrequent terms to be removed is set to 5, which may cause removal of significant specialized key terms.

Despite only the Reddit data sets being available for predictions, some overly deviating results were found between data sets before this step. The clustering results for Reddit and News Aggregator data sets when using topic embedding had vastly different metrics compared to the variation when using traditional topic modeling. A mentioned earlier, the Reddit data sets yielding higher clustering values than the News Aggregator data set does not need to indicate that the predictions made on the latter would be of poorer quality. In fact, it could be a consequence of so many words being lost in the preprocessing step of removing words out of the vocabulary, resulting in even shorter documents, that documents become harder to differenciate. This highlights the need for further experiments on the method, and potentially the methods would perform better on a real news data set considering NLP and content quality.

Nevertheless these challenges propagating into the previously discussed optimization problems, some of the methods do seem be able to generate useful models, which can be seen from the prediction results. The results show that exploiting semantic analysis to dimensionality reduce vocabulary to topics improves the cold item prediction process significantly, even though further dimensionality reduction through topical community detection did not make as much improvement with given data. As seen, this may stem in a numerous reason. One not discussed, but mentioned briefly, is the preference function which may be too discriminative. A customized decay rate or time slice could be added to reduce the effect of accumulation of common clusters, strengthening rare clusters which are currently being suppressed. Additionally, a function to aid useful recommendations for users showing interest in only small clusters (rarely having new objects for recommendation) should be considered. Further, as discussed in section 3.4, this require a sophisticated method, whereas methods to merely disfavor older instances is not sufficient, and even low or no decay rate provide better prediction abilities at times. While initial experiments on updating word embeddings while maintaining alignment of topics embedded in the same space, were not successful enough to warrant further research in the context of this thesis, the subject could justify additional investigation. This could allow for alignment considerations and incremental updates within the scope of the topic embedder.

Furthermore the results must be seen in light of the data set deviating from a real news recommendation scenario, both language wise and in terms of other features, such as users' read history, reading context, and background information about users, like age and gender. Data sets with these kind of features are, as discussed earlier, often difficult to obtain for a numerous reasons, such as privacy and strategy. Additionaly, these features may even for private news providers be difficult to obtain, especially due to demand for user consent as a result of privacy regulations, such as General Data Protection Regulation (GDPR)[2]. Demanding users to leave feedback is, as seen, in general a challenge, and assumptions made in

[2] https://gdpr-info.eu/

the Reddit data sets differs from a real news domain by being based on comments rather than users' read history. This may result in more precise positive implicit feedback in the Reddit data set, as a comment is more likely a stronger indication of interest than visiting an article. On the other hand, the Reddit data sets lacks the interest shown through clicks, and absence of activity might be punished too hard, strengthened even more by new categories on Reddit being more difficult to explore than in a regular news domain. Furthermore, evaluation in recommender systems generally suffers from false negatives on items with absence of interaction, which is especially relevant in the news domain due to it's sparsity.

These differences are challenges that might affect the predictors in a real news domain. Unfortunately, the prediction process was not tested on the News Aggregator data set lacking user history, but traits of the deviations could be observed in the sub processes, where different parameters were preferred for the different data sets. The model is thus generally very dependant on the training set, as it sets the basis for where future documents may be placed. This does however not implicate that a predictor purely trained on news article would perform worse, and an interesting improvement could be to attempt to connect the headlines in the News Aggregator data to posts on Reddit to obtain user interactions. This and other subjects for improvements will be further discussed in the final chapter next.

Chapter 7

Conclusion

This thesis addresses the problem of making useful predictions in the volatile and sparse news domain by utilizing dimensionality reduction through semantic analysis, community detection, and user modeling, in order to overcome the lack of sufficient user feedback. A pipeline of processes conducting these reductions has been implemented, where detection of topical communities and generation of user models has been constructed from both topic embedding and topic modeling. The developed predictor models were applied in the user prediction phase, and results have been compared and discussed. The following chapter will summarize the contributions of this research by answering the introductory objectives listed and presented in detail in section 1.3, before finally summing up the key challenges and discuss suggestions to further work.

7.1 Research Contributions

The results discussed in chapter 6 indicate a notable gain from performing semantic based dimensionality reduction in the prediction process. However, deviations and limitations related to the data set are also observed, which might lead to deceptive results in terms of the models' quality and potential in the news domain. Despite being inadequate to draw final conclusions about the models capability and performance in a real news domain, the research gives useful insights on general key problems in cold item recommendation, a topic of significant relevance in the news domain. How the research meets the research objectives, will not be accounted for in turn.

In answer to the first question, we have found that representation of documents through semantic analysis can improve recommendations, being able to retrieve topics of a finer granularity than the manually labeled, traditional news categories. We have looked at how two different generative models that allows multiple labels per article performs this task, namely generative topic modeling through LDA and generative embeddings through TopicVec. When extracting item information, it is important to generate a model which efficiently can infer identically formatted

information for items introduced at a later time, for comparison purposes. These models provide this on textual items through detecting common patterns in training, and applying them to unseen data to label them accordingly. This helps alleviate the cold item-side issue, through item enriching at introduction time.

To deal with the sparsity issue, encompassed in the second research question, dimensionality reduction is essential not only to reduce documents compositions of abstract, semantic topics, but also to represent these in topical communities and user models. By employing topic modeling, each document is given a set-length vector which can be controlled by parameters. This allows documents to be represented in a meaningful manner, while avoiding excessively large representations such as word count vectors. This way, documents can more efficiently be compared to each other, making the clustering more efficient through dimensionality reduction, as well as making the clusters based on rich topical information. When adding clustering, users are able to be described as a set-length vector representing their activity in each community, allowing composite topical interests to be presented. Using these clusters as a manner of representing users, significantly reduces their size from traditional n-lenght vectors, where n is the size of the item vocabulary. This has further positive effect on the efficiency, also encompassed in the second research question.

In relation to the third research question, we use community detection in order to represent more complex user interests. By topic modeling documents with the generative LDA and TopicVec, we gain the ability to infer a new document's topics and attributes at introduction time. This aids in getting enough item information to recommend items that would otherwise be considered cold through lack of user history or descriptive labels. This is shown by predicting user interests for items without user history attached to them. By grouping a number of documents together, we also gain information about the collective tendencies in the cluster. This is noted in the increase of performance when clustering compared to a simpler pairing of documents which match a users average interests.

7.2 Further Work

As discussed in chapter 6, there are several challenges that could be addressed in continuation of this work. In this section we will discuss interesting oppotunities and possible improvements, and how important parts of the framework may be modified and optimised.

One of the largest potential for improvements lie in the data set. The optimal solution would be collecting sufficient articles and persistent user read history from a publishing house over time to perform large-scale training and testing. This would allow for more accurate training of the models, as well as providing an environment for testing topic embedding alignment through word embedding updates. These types of data sets are however difficult to come by, especially with GDPR leading to companies enforcing stricter data collection and sharing policies.

If the lack of news domain data sets containing all article and user information needed to perform further research persists, more domain specific subset selection

and preprocessing should be employed to better simulate a real news setting. This could include, in the case of the Reddit data set, doing more research into which subreddits or sub-data sets which could have more appropriate user behaviour. For preprocessing, removing specific words or documents which displays very distinct forum behavioural patterns could help. Actions such as this requires domain knowledge within both news and forums, in order to approximate a news setting without loosing information or substance contained in the original data set. Additionally, utilizing time stamps to test the models in a more realistic timeline would an important step in evaluating its application in the news domain. This could be done by sorting the data into training and testing sets by date, training on data collected before a set date, and simulate a real-time publishing pace and sequence when testing. This would also put higher weight on the importance of efficiency and optimization of the methods.

The time aspect is also important in terms of evolution of communities and interests. Indeed, identifying short term and long term interests could be helpful, and some important challenges when dealing with long term interests is related to drift of concepts and alignment issues. Over time the words identifying topics, and hence communities may change, and when updating models, they need to be able to be aligned. This is also important in terms of new words in the vocabulary. While the updating of word embeddings to ensure alignment with topic embedding did not provide promising results in this experiment setting, it may warrant further research in long-term systems to discern whether there is a need for updates to maintain usability, and how the new words eventually would be associated with topics.

While this thesis has focused on the needs for articles to be recommendable at system introduction time, it is also important to determine when it becomes outdated for further recommendations. Handling news specific behavioural patterns is difficult, but recognizing different types of articles through manual tags, content or user behavioural patterns surrounding it, could assist in making better recommendations based on more item-specific information. Discovering not only topics, but also different types of articles such as interviews, traffic warnings and breaking news, may assist in determining the rate at which an article's interest level is decaying at.

Comparison with Relevant Data Sets					
Dataset	**Users**	**Items**	**Edges**	**Density (%)**	**Rating Scale**
News Aggregator	NA		422 419	NA	NA
20 Newsgroups	NA		$\sim$ 20 000	NA	NA
Reuters21578	NA		21 578	NA	NA
reddit	33 079	390 831	2 478 733	0.02	Comment Counts
Last.fm	1 892	17 632	92 834	0.28	Play Counts
MovieLens 20M	138 493	27 278	20 000 263	0.52	[0.5-5]

Acronyms

AP Average Precision. 41, 86

BCD Block Coordinate Descent. 18, 33, 34

BoW Bag-of-Words. 32

C-H Calinski-Harabasz. 9, 33, 47, 58–60, 63, 65, 66, 80, 82, 83, 89

CB Content Based. 11, 25

CF Collaborative Filtering. 11, 25, 40

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 8, 32

DL Deep Learning. 16, 17

GDPR General Data Protection Regulation. 91, 94

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise. 32, 33, 46, 65, 66, 83, 85, 122

IDI Department of Computer Science. ii

IR Information Retrieval. 9, 11, 24

KL Kullback–Leibler. 46, 84

LCE Local Collective Embedding. 26, 27

LDA Latent Dirichlet Allocation. 5, 14, 15, 22, 23, 31, 32, 36, 40, 46, 54–57, 60, 73, 77, 80, 83, 89, 90, 93, 111–117

MAP Mean Average Precision. 12, 41, 49, 83, 85, 86, 88, 89

MF Matrix Factorization. 19, 22

ML Machine Learning. 13, 14

MRR Mean Reciprocal Rank. 13

NLP Natural Language Processing. 13, 14, 16, 19, 23, 91

NMF Nonnegative Matrix Factorization. 20

NN Neural Network. 16, 17

NNMA Nonnegative Matrix Approximation. 20

NTNU Norwegian University of Technology and Science. ii, 4

PCA Principal Component Analysis. 20, 33, 47, 54, 78

PR Percentile Ranking. 41

RA Ranking Accuracy. 27, 41, 49, 85, 86, 88, 89

RS Recommender System. 1, 4, 9–12, 26, 28, 37, 39, 43

SVD Singular Value Decomposition. 19, 22

t-SNE t-Distributed Stochastic Neighbor Embedding. 19, 33, 47, 54, 109

Bibliography

[1] Özlem Özgöbek, Jon Atle Gulla, and Riza Cenk Erdur. A survey on challenges and methods in news recommendation. In *WEBIST (2)*, pages 278–285, 2014.

[2] William Yardley and Richard Pérez-Peña. Seattle post-intelligencer shifts to web only, Mar 2009.

[3] Pew Research Center. State of the news media 2016, Jun 2016.

[4] Robert Seamans and Feng Zhu. Responses to entry in multi-sided markets: The impact of craigslist on local newspapers. *Manage. Sci.*, 60(2):476–493, February 2014.

[5] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 158–166, New York, NY, USA, 1999. ACM.

[6] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[7] Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. The adressa dataset for news recommendation. In *Proceedings of the International Conference on Web Intelligence*, pages 1042–1048. ACM, 2017.

[8] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining: concepts and techniques (the morgan kaufmann series in data management systems). *Morgan Kaufmann*, 2000.

[9] M Girvan. Girvan, m. & newman, m. e. j. community structure in social and biological networks. proc. natl acad. sci. usa 99, 7821-7826. *Proceedings of the National Academy of Sciences of the United States of America*, 99:7821, 2002.

[10] Bisma S Khan and Muaz A Niazi. Network community detection: A review and visual survey. *arXiv preprint arXiv:1708.00977*, 2017.

[11] John A Hartigan. Clustering algorithms. 1975.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[13] Richard C Dubes. Cluster analysis and related issues. In *Handbook of pattern recognition and computer vision*, pages 3–32. World Scientific, 1999.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[15] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. Understanding of internal clustering validation measures. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 911–916. IEEE, 2010.

[16] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[17] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002.

[18] Xujuan Zhou, Yue Xu, Yuefeng Li, Audun Josang, and Clive Cox. The state-of-the-art in personalized recommender systems for social networking. *Artificial Intelligence Review*, 37(2):119–132, Feb 2012.

[19] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 824–831, New York, NY, USA, 2005. ACM.

[20] Peter Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1):87–110, Mar 2001.

[21] D. N. Chin. User modeling in uc, the unix consultant. *SIGCHI Bull.*, 17(4):24–28, April 1986.

[22] Douglas W. Oard and Jinmook Kim. Implicit feedback for recommendation systems. 1998.

[23] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. 2004.

[24] Masahiro Morita and Yoichi Shinoda. Information filtering based on user behavior analysis and best match text retrieval. 1994.

[25] Florian Mueller and Andrea Lockerd. Cheese: tracking mouse movement activity on websites, a tool for user modeling. 2001.

[26] F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261 – 273, 2015.

[27] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov 2002.

[28] Michael J. Pazzani and Daniel Billsus. *Content-Based Recommendation Systems*, pages 325–341. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[29] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

[30] Martin Saveski and Amin Mantrach. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 89–96. ACM, 2014.

[31] Diego Saez-Trumper, Daniele Quercia, and Jon Crowcroft. Ads and the city: considering geographic distance goes a long way. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 187–194. ACM, 2012.

[32] Chowdhury Gobinda G. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89.

[33] Jose Bernardo, M J Bayarri, J O Berger, A P Dawid, David Heckerman, A F M Smith, Mike West, Christopher M Bishop, and Julia Lasserre. Generative or discriminative? getting the best of both worlds. 8:3–24, 01 2007.

[34] Tony Jebara and Alex P Pentland. *Discriminative, generative and imitative learning*. PhD thesis, PhD thesis, Media laboratory, MIT, 2001.

[35] Kristina Toutanova. Competitive generative models with structure learning for nlp classification tasks. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 576–584. Association for Computational Linguistics, 2006.

[36] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

[37] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. *SIGIR Forum*, 51(2):268–276, August 2017.

[38] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[39] Matus Telgarsky. Dirichlet draws are sparse with high probability. *CoRR*, abs/1301.4917, 2013.

[40] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[41] Andrew Gelman. *Prior Distribution*, volume 3, pages 1634–1637. John Wiley & Sons, Ltd, 2002.

[42] James M Dickey. Multiple hypergeometric functions: Probabilistic interpretations and statistical uses. *Journal of the American Statistical Association*, 78(383):628–637, 1983.

[43] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296, 2009.

[44] Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 952–961. Association for Computational Linguistics, 2012.

[45] Xitong Yang. Understanding the variational lower bound. 2017.

[46] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[47] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[48] Maciej A. Mazurowski, Piotr A. Habas, Jacek M. Zurada, Joseph Y. Lo, Jay A. Baker, and Georgia D. Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural Networks*, 21(2):427 – 436, 2008. Advances in Neural Networks Research: IJCNN '07.

[49] J. Leonard and M.A. Kramer. Improvement of the backpropagation algorithm for training neural networks. *Computers & Chemical Engineering*, 14(3):337 – 341, 1990.

[50] Ref. 106, p. 276-291.

[51] E. Barnard. Optimization for training neural nets. *IEEE Transactions on Neural Networks*, 3(2):232–240, Mar 1992.

[52] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[53] Ref. 106, p. 401.

[54] Ref. 106, p. 426.

[55] Ref. 106, p. 476-477.

[56] Shaohua Li, Jun Zhu, and Chunyan Miao. A generative word embedding model and its low rank positive semidefinite solution. *CoRR*, abs/1508.03826, 2015.

[57] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[58] Alan Julian Izenman. Linear discriminant analysis. In *Modern multivariate statistical techniques*, pages 237–280. Springer, 2013.

[59] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[60] Suvrit Sra and Inderjit S Dhillon. Generalized nonnegative matrix approximations with bregman divergences. In *Advances in neural information processing systems*, pages 283–290, 2006.

[61] Jaewon Yang and Jure Leskovec. Community-affiliation graph model for overlapping network community detection. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1170–1175. IEEE, 2012.

[62] Hongyi Zhang, Irwin King, and Michael R Lyu. Incorporating implicit link preference into overlapping community detection. In *AAAI*, pages 396–402, 2015.

[63] Jaegul Choo, Changhyun Lee, Chandan K Reddy, and Haesun Park. Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE transactions on visualization and computer graphics*, 19(12):1992–2001, 2013.

[64] Quanquan Gu, Jie Zhou, and Chris Ding. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 199–210. SIAM, 2010.

[65] Melissa Ailem, Aghiles Salah, and Mohamed Nadif. Non-negative matrix factorization meets word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1081–1084. ACM, 2017.

[66] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

[67] Liu Xin, E Haihong, Junde Song, Meina Song, and Junjie Tong. Book recommendation based on community detection. In *Joint International Conference on Pervasive Computing and the Networked World*, pages 364–373. Springer, 2013.

[68] Yuan Wen, Yun Liu, Zhen-Jiang Zhang, Fei Xiong, and Wei Cao. Compare two community-based personalized information recommendation algorithms. *Physica A: Statistical Mechanics and its Applications*, 398:199–209, 2014.

[69] Haoyuan Feng, Jin Tian, Harry Jiannan Wang, and Minqiang Li. Personalized recommendations based on time-weighted overlapping community detection. *Information & Management*, 52(7):789–800, 2015.

[70] Guy Shani, Amnon Meisles, Yan Gleyzer, Lior Rokach, and David Ben-Shimon. A stereotypes-based hybrid recommender system for media items. 2007.

[71] Haoyuan Feng, Jin Tian, Harry Jiannan Wang, and Minqiang Li. Personalized recommendations based on time-weighted overlapping community detection. 2015.

[72] Kan Zhang, Zichao Zhang, Kaigui Bian, Jin Xu, and Jie Gao. A personalized next-song recommendation system using community detection and markov model. In *Data Science in Cyberspace (DSC), 2017 IEEE Second International Conference on*, pages 118–123. IEEE, 2017.

[73] Xin Li, Jun Yan, Weiguo Fan, Ning Liu, Shuicheng Yan, and Zheng Chen. An online blog reading system by topic clustering and personalized ranking. *ACM Transactions on Internet Technology (TOIT)*, 9(3):9, 2009.

[74] Yonghui Wu, Yuxin Ding, Xiaolong Wang, and Jun Xu. A comparative study of topic models for topic clustering of chinese web news. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 5, pages 236–240. IEEE, 2010.

[75] S. Hui and Z. Dechao. A weighted topical document embedding based clustering method for news text. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pages 1060–1065, May 2016.

[76] Guangxu Xun, Yaliang Li, Wayne Xin Zhao, Jing Gao, and Aidong Zhang. A correlated topic model using word embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 4207–4213. AAAI Press, 2017.

[77] Guangxu Xun, Yaliang Li, Jing Gao, and Aidong Zhang. Collaboratively improving topic discovery and word embeddings by coordinating global and local contexts. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 535–543, New York, NY, USA, 2017. ACM.

[78] Junxian He, Zhiting Hu, Taylor Berg-Kirkpatrick, Ying Huang, and Eric P. Xing. Efficient correlated topic modeling with topic embedding. *CoRR*, abs/1707.00206, 2017.

[79] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[80] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[81] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[82] Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. Generative topic embedding: a continuous representation of documents (extended version with proofs). *CoRR*, abs/1606.02979, 2016.

[83] Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313, 2015.

[84] Rajarshi Das, Manzil Zaheer, and Chris Dyer. Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 795–804, 2015.

[85] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *AAAI*, pages 2418–2424, 2015.

[86] Makbule Gulcin Ozsoy. From word embeddings to item recommendation. *CoRR*, abs/1601.01356, 2016.

[87] Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec - product embeddings using side-information for recommendation. *CoRR*, abs/1607.07326, 2016.

[88] Cataldo Musto, Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops. Learning word embeddings from wikipedia for content-based recommender systems. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Fabrizio Mothe, Josiane Gianmaria Silvello, editors, *Advances in Information Retrieval*, pages 729–734, Cham, 2016. Springer International Publishing.

[89] Robinson Meyer. How many stories do newspapers publish per day?, May 2016.

[90] Olga Kouropteva, Oleg Okun, and Matti Pietikäinen. Incremental locally linear embedding. *Pattern Recognition*, 38(10):1764 – 1767, 2005.

[91] D. Zhao and L. Yang. Incremental isometric embedding of high-dimensional data using connected neighborhood graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):86–98, Jan 2009.

[92] Elaine Rich. User modeling via stereotypes*. *Cognitive Science*, 3(4):329–354, 1979.

[93] Béatrice Lamche, Enrico Pollok, Wolfgang Wörndl, and Georg Groh. Evaluation the effectiveness of stereotype user models for recomenndations on mobile devices. 2014.

[94] Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. User modeling with neural network for review rating prediction. In *IJCAI*, pages 1340–1346, 2015.

[95] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. Dynamic user modeling in social media systems. *ACM Transactions on Information Systems (TOIS)*, 33(3):10, 2015.

[96] Titipat Achakulvisut, Daniel E. Acuna, Tulakan Ruangrong, and Konrad Kording. Science concierge: A fast content-based recommendation system for scientific publications. *PLOS ONE*, 11(7):1–11, 07 2016.

[97] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, New York, NY, USA, 2011. ACM.

[98] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1235–1244, New York, NY, USA, 2015. ACM.

[99] Alexey Tsymbal. The problem of concept drift: definitions and related work. 2004.

[100] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 485–492, New York, NY, USA, 2005. ACM.

[101] Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, April 2010.

[102] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[103] Moshe Lichman. Uci machine learning repository [http://archive.ics.uci.edu/ml], 2013. Irvine, CA: University of California, School of Information and Computer Science.

[104] Xiao Yang, Craig MacDonald, and Iadh Ounis. Using word embeddings in twitter election classification. *CoRR*, abs/1606.07006, 2016.

[105] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL*, 2014.

[106] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Appendix

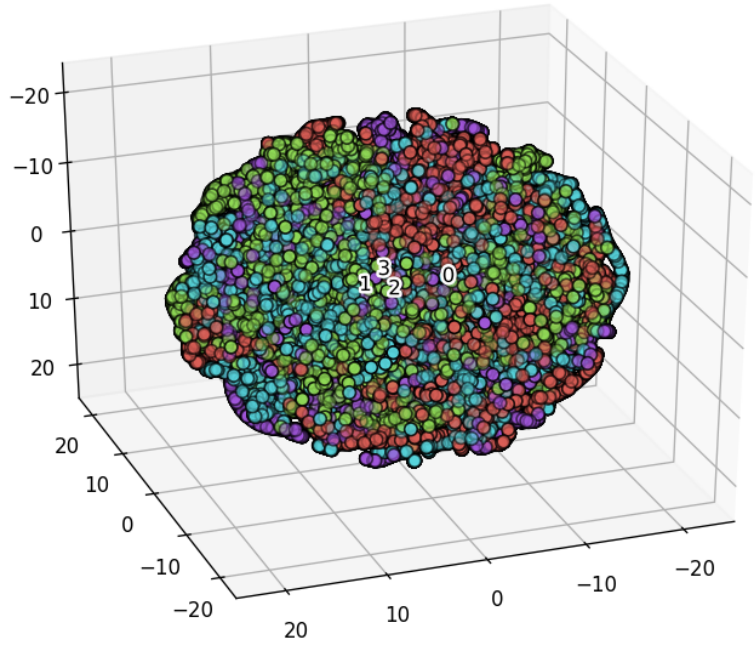

Figure A.1: An example of a typical t-SNE visualization. The following shows the K-Means clustering with 4 clusters and 4 topics, which can also be seen figure 6.5.

Topic Embedding Components	
α	The Dirichlet prior, applied to the document-topic mixing proportions using the principle that one document normally incorporates a few topics [82].
θ_d	The distribution parameter for inference approximation of the mixing proportions.
t	A topic embedding contained in doc d.
z_c	Topic assignment for the focus word c in document d.
w_n	Each of the context words from $w_0 : w_{c-1}$ influencing the current focus word w_c, also presented in the word embedding model from [56].
w_c	The current context word, the only constantly observable variable.
Word Embedding Specific components	
μ_i	Embedding magnitude penalty coefficient [82].
h_{ij}	Empirical bigram probabilities [82].
v_{s_i}	Word embedding of word s_i from the embedding vocabulary trained by the method in [56].
$a_{s_i s_j}$	The bigram residual of word s_i and s_j, sometimes referred to as a bias term. It captures noisy and non-linear interaction between the bigram words [56].

Table A.1: Description of TopicVec components.

Spec.	**Server**				
	1	**2**	**3**	**4**	**5**
OS	Ubuntu 16.04.4 LTS	Ubuntu 16.04.3 LTS	Ubuntu 16.04.3 LTS	Ubuntu 16.04.3 LTS	macOS High Sierra10.13.3
CPU	Intel(R) Xenon(R) CPU E5-2640 v4@2.40GHz	Intel(R) Core(TM) i7-4470 CPU@340GHz	Intel(R) Core(TM) i5-3317U CPU@1.70GHz	Intel(R) Core(TM) i7-4470 CPU@340GHz	Intel(R) Core(TM) i5-5257U CPU@2.70GHz
Cores	4x10	2x4	2x2	2x4	2x2
Memory	∼132GB	∼16GB	∼8GB	∼16GB	∼8GB

Table A.2: Runtime Environments.

t	Perplexity			Bounds			Topic Coherence		
	Reddit Post	**Reddit Title**	**News Aggregator**	**Reddit Post**	**Reddit Title**	**News Aggregator**	**Reddit Post**	**Reddit Title**	**News Aggregator**
4	2.15E-03	2.12E-03	3.00E-03	-3.53E+06	-3.54E+06	-4.22E+06	-4.28	-4.83	-7.3
5	2.05E-03	2.02E-03	2.92E-03	-3.56E+06	-3.57E+06	-4.24E+06	-4.12	-4.12	-7.31
10	1.85E-03	9.90E-04	1.31E-03	-3.62E+06	-4.74E+06	-6.18E+06	-5.38	-5.69	-7.31
15	1.71E-03	1.74E-03	2.74E-03	-3.67E+06	-3.66E+06	-4.28E+06	-5.85	-6.86	-7.9
20	1.61E-03	8.68E-04	1.26E-03	-3.70E+06	-4.83E+06	-6.21E+06	-6.42	-7.16	-8.7
30	1.48E-03	1.49E-03	2.54E-03	-3.75E+06	-3.75E+06	-4.34E+06	-7.26	-7.14	-8.62
40	1.39E-03	7.30E-04	1.19E-03	-3.78E+06	-4.95E+06	-6.26E+06	-8.09	-8.15	-9.36
50	1.34E-03	7.00E-04	1.16E-03	-3.81E+06	-4.98E+06	-6.29E+06	-8.51	-8.33	-9.71
60	1.32E-03	6.76E-04	1.16E-03	-3.82E+06	-5.00E+06	-6.29E+06	-9.56	-9.5	-9.75
70	1.27E-03	6.49E-04	1.14E-03	-3.84E+06	-5.03E+06	-6.31E+06	-10.01	-9.91	-10.52
80	3.89E-05	4.00E-05	5.42E-04	-5.84E+06	-5.83E+06	-5.47E+06	-10.56	-10.93	-11.07
90	4.03E-06	2.34E-08	1.72E-07	-7.15E+06	-1.20E+07	-1.45E+07	-11.22	-10.83	-11.52
100	2.06E-06	2.07E-06	5.94E-05	-7.54E+06	-7.53E+06	-7.07E+06	-11.66	-11.76	-11.98
110	1.04E-06	2.10E-09	2.29E-08	-7.93E+06	-1.37E+07	-1.64E+07	-11.94	-12.13	-11.97
120	5.36E-07	6.41E-10	8.32E-09	-8.31E+06	-1.45E+07	-1.73E+07	-12.31	-12.6	-12.03
130	2.72E-07	1.94E-10	3.00E-09	-8.70E+06	-1.53E+07	-1.83E+07	-12.92	-12.68	-12.51
140	1.39E-07	5.74E-11	1.09E-09	-9.09E+06	-1.62E+07	-1.92E+07	-13.42	-13.25	-12.66
150	7.14E-08	1.75E-11	4.00E-10	-9.47E+06	-1.70E+07	-2.01E+07	-13.72	-13.5	-12.84
160	3.64E-08	5.14E-12	1.43E-10	-9.86E+06	-1.78E+07	-2.11E+07	-13.79	-13.47	-13.25
170	1.86E-08	1.57E-12	5.28E-11	-1.02E+07	-1.86E+07	-2.20E+07	-13.72	-14.02	-13.12
180	9.50E-09	4.71E-13	1.89E-11	-1.06E+07	-1.95E+07	-2.30E+07	-14.24	-13.88	-13.37
190	4.81E-09	1.41E-13	6.88E-12	-1.10E+07	-2.03E+07	-2.39E+07	-14.45	-14.23	-13.54
200	2.47E-09	4.25E-14	2.48E-12	-1.14E+07	-2.11E+07	-2.49E+07	-14.55	-14.38	-13.85
210	1.25E-09	1.29E-14	9.03E-13	-1.18E+07	-2.19E+07	-2.58E+07	-14.99	-14.52	-14.08
220	6.45E-10	3.86E-15	3.24E-13	-1.22E+07	-2.27E+07	-2.68E+07	-14.83	-14.75	-14.24
230	3.27E-10	1.16E-15	1.17E-13	-1.26E+07	-2.36E+07	-2.77E+07	-15.03	-14.86	-14.3

Table A.3: LDA topic modeling results

Data Set	Topic 1	Topic 2	Topic 3	Topic 4
Reddit Posts	rock	thank	movie	watch
	music	help	online	film
	electronic	question	video	song
	hip hop	try	free	india
	world	lose weight	review	cover
	live	start	hip hop	movie
	star war	month	company	play
	love	time	release	live
	indie rock	body	hd	band
	alternative	eat	service	rock
Reddit Title	trailer	cover	online	movie
	film	hip hop	review	electronic
	technology	rock	time	watch
	movie	metal	video	start
	internet	friend	buy	download
	home	game throne	service	tv
	time	love	pop	star war
	data	cut	help	hd
	video	music	house	india
	help	episode	business	story
News Aggregator	justin bieber	microsoft	game throne	kim kardashian
	lindsay lohan	report	samsung galaxy	miley cyrus
	video	award	google glass	kanye west
	selena gomez	china	time	star war
	samsung galaxy	apple	company	help
	gm recall	confirm	woman	captain america
	american idol	rise	feature	box office
	fan	bank	death	report
	deal	list	facebook	video
	kim kardashian	increase	april	dance star

Table A.4: Top 10 terms for LDA topic models with four topics where sub-terms in discovered bi-grams are not kept in documents.

Reddit Posts				
Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**
eat	music	market	service	business
weight	rock	heart	google	technology
lose	song	pain	company	product
fat	album	news	watch	android
body	band	repair	season	free
cut	indie	deeper	episode	system
time	video	disease	business	apple
diet	pop	cause	sale	phone
calorie	listen	garage door	internet	iphone
gain	play	update	change	benefit
Topic 6	**Topic 7**	**Topic 8**	**Topic 9**	**Topic 10**
scientist	help	post	hiphop	watch
life	guy	thread	rap	trailer
brain	look	top	godzilla	stream
science	thank	online	metal	hd
cell	start	link	monster	live
time	people	jam	game throne	download
human	try	jersey	west	movie online
researcher	fitness	list	post	amaze
child	idea	tv	green	spider
discover	gym	comment	star	online free
Topic 11	**Topic 12**	**Topic 13**	**Topic 14**	**Topic 15**
lift	house	loan	movie	cosmos
squat	official	remix	film	design
workout	world	cash	watch	website
lb	mobile	credit	song	film
set	video	air	time	th
weight	care	street	scene	development
exercise	bank	medium	love	oscar
start	window	wall	favorite	real
routine	winter	social	character	gt
bench	captain america	car	remember	oddysey

Table A.5: Top 10 terms extracted with LDA topic model run on Reddit Posts data set with 15 topics.

Reddit Title				
Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**
workout	tv	hd	song	train
lift	phone	movie online	love	facebook
change	garage door	real	rock	scientist
look	service	online free	house	system
book	window	film	cover	build
routine	age	song	music video	cell
climate	michael	winter	pop	apple
freeze	door repair	captain america	rise	life
drink	amazon	box	official	time
time	series	free download	acoustic	create
Topic 6	**Topic 7**	**Topic 8**	**Topic 9**	**Topic 10**
lose	hip hop	health	movie	diet
car	rap	fitness	favorite	question
star war	squat	help	film	weight loss
episode	android	body	time	solar
find	app	care	guy	break
self	market	fat	awesome	cut
service	io	tip	theme	song
drive	business	home	playlist	obat
customer	form check	exercise	song	eat
city	beat	weight	robot	clean
Topic 11	**Topic 12**	**Topic 13**	**Topic 14**	**Topic 15**
trailer	advice	amaze	game	business
loan	world	future	pop	service
internet	team	metal	album	company
cash	heart	day	indie rock	design
official	trade	past	live	website
short	cover	release	review	credit
neutrality	cheap	song	folk	development
wall	food	glass	punk	free
financial	girl	spider	country	godzilla
true	stock	pop	alternative	solution

Table A.6: Top 10 terms extracted with LDA topic models run on reddit data set with 15 topics.

News Aggregator				
Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**
stock	season	game_throne	apple	facebook
rate	voice	court	samsung_galaxy	future
market	tesla	box	launch	day
data	fire	tv	iphone	internet
dollar	bachelorette	justin_bieber	android	user
feed	finale	dance	miley_cyrus	past
euro	surface	season	htc	health
miss	spider	office	release	watch
gain	walk	home	google	chris
ecb	time	video	smartphone	celebrate
Topic 6	**Topic 7**	**Topic 8**	**Topic 9**	**Topic 10**
google	woman	microsoft	cancer	kim kardashian
meet	drug	deal	report	wed
story	fda	mobile	mar	video
mother	lindsay lohan	tax	test	kanye_west
white	risk	window	record	earth
io	fcc	billion	climate change	music
app	neutrality	china	obama	photo
late	trailer	mers	ebay	cover
harry	heart	roll	top	discover
gay	link	russia	american idol	marry
Topic 11	**Topic 12**	**Topic 13**	**Topic 14**	**Topic 15**
movie	bank	oil	apple	sale
amazon	ebola	rise	beyonce	heartbleed
buy	xbox	profit	beat	car
google	outbreak	fall	jay	ban
award	share	challenge	job	month
service	captain_america	age	cut	google_glass
review	target	china	space	stock
star war	moon	drop	run	million
stream	update	alibaba	attack	bug
drive	george	dy	buy	march

Table A.7: Top 10 terms extracted with LDA topic models run on News Aggregator data set with 15 topics.

Reddit Titles				
Topic 65	**Topic 49**	**Topic 14**	**Topic 36**	**Topic 76**
electronic	apple	workout	people	guy
list	director	lift	blue	bad
remix	screen	mobile	bank	exercise
title	direct	phone	shoot	get
cloud	billion	version	key	credit
pick	marvel	uk	strong	theater
host	government	co	feedback	protect
march	difference	paper	smoke	help
bölüm	choice	apps	virus	manage
drum	standard	decide	bird	publish
Topic 34	**Topic 78**	**Topic 71**	**Topic 69**	**Topic 7**
gym	muscle	world	rock	scene
million	gain	power	indie	product
design	bench	cheap	band	social
website	press	team	pop	medium
web	develop	jersey	album	go
die	smartphone	energy	folk	security
drop	growth	solar	song	disney
return	fuck	produce	share	week
voice	kid	national	love	studio
beautiful	process	oil	mean	solution
Topic 8	**Topic 48**	**Topic 39**	**Topic 21**	**Topic 27**
favorite	music	break	price	find
post	video	hour	financial	app
jam	weight	cause	dream	news
summer	lose	lb	charge	android
soul	listen	blood	sun	launch
song	loss	fun	offer	update
time	space	patent	dog	hate
leak	fall	effective	teach	store
barbell	official	equipment	core	io
spend	vote	beginner	decision	genre

Table A.8: Top 10 terms for then 15 most significant topics extracted with LDA topic model run on reddit title data set with 80 topics.

News Aggregator				
Topic 57	**Topic 20**	**Topic 59**	**Topic 39**	**Topic 65**
record	game	death	lead	office
break	season	voice	smartphone	box
reach	throne	continue	online	age
fast	finale	happy	tweet	girl
hollywood	recap	williams	support	baby
level	kill	pharrell	store	biggest
dy	episode	moment	amazon	transformer
researcher	spoiler	video	image	pregnant
reduce	network	west	propose	human
rooney	harris	weather	forget	hack
Topic 19	**Topic 22**	**Topic 79**	**Topic 70**	**Topic 74**
family	bieber	price	april	watch
bachelorette	justin	sale	house	dog
htc	love	rise	quarter	texas
steal	selena	home	white	bear
andi	gomez	profit	six	scene
france	view	earn	kick	watchdog
pm	late	fall	marriage	hike
ape	secret	gas	johnson	highlight
learn	reason	street	field	trailer
prevent	prince	jump	nd	video
Topic 54	**Topic 26**	**Topic 55**	**Topic 56**	**Topic 43**
feed	charge	car	review	tech
move	acquire	drive	movie	ipo
tesla	surge	ecb	future	alibaba
model	social	europe	day	file
improve	medium	trial	lg	boost
yellen	meteor	policy	past	mark
word	acquisition	oscar	ready	scientist
toyota	shower	self	perform	local
worry	check	community	hope	bnp
shock	directv	draghi	performance	guilty

Table A.9: Top 10 terms for then 15 most significant topics extracted with LDA topic model run on News Aggregator data set with 80 topics.

Table A.10: K-Means with Euclidian distance. (Displayed below.)

Parameters		Reddit Posts		Reddit Title		News Aggregator	
topics	**k**	**Silhouette**	**C-H**	**Silhouette**	**C-H**	**Silhouette**	**C-H**
4	4	-0.01	2.05	-0.01	0.28	3.92	-0.03
4	5	-0.01	1.02	-0.01	0.62	4.90	-0.03
4	10	-0.02	0.83	-0.03	0.77	9.80	-0.04
4	20	-0.05	0.77	-0.05	0.94	19.60	-0.07
4	30	-0.06	0.76	-0.06	0.71	29.40	-0.08
4	40	-0.08	0.76	-0.08	0.82	39.20	-0.10
4	50	-0.09	0.71	-0.09	0.82	-0.08	0.78
4	60	-0.10	0.69	-0.10	0.73	58.80	-0.12
4	70	-0.12	0.75	-0.12	0.91	68.60	-0.14
4	80	-0.14	0.82	-0.14	0.92	78.40	-0.16
4	90	-0.14	1.05	-0.15	0.91	88.20	-0.16
4	100	-0.17	1.07	-0.16	0.95	98.00	-0.19
5	5	-0.01	1.29	-0.02	0.95	4.90	-0.03
5	10	-0.02	2.01	-0.04	0.60	9.80	-0.04
5	20	-0.04	1.34	-0.06	0.60	19.60	-0.06
5	30	-0.05	1.00	-0.09	0.90	29.40	-0.07
5	40	-0.08	1.10	-0.08	0.89	39.20	-0.10
5	50	-0.07	0.87	-0.14	0.91	49.00	-0.09
5	60	-0.10	0.92	-0.11	0.86	58.80	-0.12
5	70	-0.11	0.92	-0.17	0.69	68.60	-0.13
5	80	-0.11	0.98	-0.19	0.9	78.40	-0.13
5	90	-0.14	0.88	-0.14	0.7	88.20	-0.16
5	100	-0.13	1.02	-0.22	0.73	98.00	-0.15
15	5	-0.01	0.42	-0.02	0.3	4.90	-0.03
15	10	-0.02	0.33	-0.04	1.92	9.80	-0.04
15	20	-0.03	0.68	-0.07	1.25	-0.04	1.18
15	30	-0.04	0.72	-0.09	1.33	-0.07	1.14
15	40	-0.07	0.77	-0.08	1.62	-0.15	1.28
15	50	-0.07	0.68	-0.13	1.22	-0.16	1.16
15	60	-0.09	0.95	-0.1	1.31	-0.1	1.17
15	70	-0.09	1.06	-0.16	1.00	-0.08	1.05
15	80	-0.09	0.97	-0.17	1.22	-0.1	1.19
15	90	-0.12	1.20	-0.13	1.11	-0.13	1.13
15	100	-0.10	0.90	-0.19	1.11	-0.19	1.19
40	5	-0.01	0.88	-0.02	0.36	-0.03	1.79
40	10	-0.03	0.71	-0.04	1.2	-0.04	1.52
40	20	-0.04	1.28	-0.07	0.93	-0.06	1.43
40	30	-0.05	0.79	-0.09	0.76	-0.08	1.16
40	40	-0.07	0.88	-0.07	0.98	-0.07	1.36
40	50	-0.07	1	-0.14	0.84	-0.12	1.47
40	60	-0.09	0.95	-0.11	0.93	-0.10	1.44
40	70	-0.09	0.97	-0.17	1.18	-0.15	1.37
40	80	-0.1	1.18	-0.2	0.92	-0.18	1.36
40	90	-0.13	1.1	-0.15	0.92	-0.13	1.56
40	100	-0.13	1.05	-0.22	0.94	-0.20	1.32
80	5	-0.02	0.84	-0.03	0.38	-0.03	1.28
80	10	-0.03	0.59	-0.05	0.8	-0.06	2.08
80	20	-0.04	1.3	-0.07	1.21	-0.08	1.03
80	30	-0.05	1.03	-0.11	0.84	-0.10	1.02
80	40	-0.08	0.95	-0.09	0.87	-0.08	1.34

Continued on next page

Table A.10 – continued from previous page

Parameters		Reddit Posts		Reddit Title		News Aggregator	
80	50	-0.08	0.84	-0.14	0.69	-0.13	1.17
80	60	-0.10	0.97	-0.11	0.94	-0.10	1.21
80	70	-0.10	1.13	-0.18	0.93	-0.15	0.94
80	80	-0.10	0.97	-0.20	0.74	-0.17	1.04
80	90	-0.13	0.92	-0.15	0.76	-0.13	1.09
80	100	-0.13	1.05	-0.25	0.92	-0.19	1.09
100	5	-0.02	1.58	-0.03	2.58	-0.03	0.99
100	10	-0.03	1.47	-0.06	0.82	-0.05	1.79
100	20	-0.04	2.01	-0.08	0.73	-0.08	1.10
100	30	-0.06	1.05	-0.11	0.59	-0.09	0.87
100	40	-0.08	1.10	-0.08	0.81	-0.08	1.04
100	50	-0.08	0.96	-0.15	0.97	-0.13	1.00
100	60	-0.10	0.97	-0.1	1.12	-0.10	1.22
100	70	-0.09	0.89	-0.18	0.98	-0.16	1.12
100	80	-0.09	0.85	-0.19	1.15	-0.17	0.94
100	90	-0.14	0.81	-0.14	1.06	-0.13	1.05
100	100	-0.12	0.80	-0.22	0.93	-0.19	1.01

Table A.11: k-Means with KL Divergence. (Displayed below.)

Parameters		Reddit Posts		Reddit Title		News Aggregator	
topics	**k**	**Silhouette**	**C-H**	**Silhouette**	**C-H**	**Silhouette**	**C-H**
4	4	-0.01	0.08	-0.03	0.72	-0.01	0.83
4	5	-0.07	0.07	-0.03	0.58	-0.01	1.37
4	10	-0.04	0.92	-0.15	0.59	-0.11	1.32
4	20	-0.23	0.76	-0.11	1.00	-0.12	2.02
4	30	-0.42	1.03	-0.06	0.41	-0.14	1.22
4	40	-0.30	0.99	-0.29	1.07	-0.22	1.13
4	50	-0.12	0.43	-0.26	0.68	-0.14	0.89
4	60	-0.39	1.00	-0.41	0.62	-0.37	0.89
4	70	-0.36	1.24	-0.25	0.79	-0.19	1.35
4	80	-0.36	0.87	-0.41	0.85	-0.38	1.08
4	90	-0.31	0.32	-0.31	1.12	-0.38	0.85
4	100	-0.43	0.94	-0.35	0.83	-0.29	1.15
5	4	-0.01	2.03	-0.01	1.46	-0.01	2.79
5	5	-0.01	1.28	-0.01	0.73	-0.01	1.85
5	10	-0.18	1.41	-0.06	1.37	-0.12	1.24
5	20	-0.13	0.78	-0.25	0.82	-0.19	1.26
5	30	-0.28	1.29	-0.24	0.71	-0.41	0.98
5	40	-0.42	1.21	-0.43	0.89	-0.48	1.03
5	50	-0.55	0.9	-0.2	0.85	-0.47	0.94
5	60	-0.51	1.00	-0.56	0.79	-0.29	1.06
5	70	-0.52	0.89	-0.71	0.87	-0.47	1.23
5	80	-0.44	0.73	-0.47	1.01	-0.74	1.02
5	90	-0.57	1.00	-0.64	0.67	-0.57	1.13
5	100	-0.52	0.99	-0.62	1.09	-0.43	1.11
15	4	-0.01	2.04	-0.01	2.21	-0.01	1.1
15	5	-0.01	0.39	-0.01	1.65	-0.01	1.14
15	10	-0.03	0.8	-0.03	1.90	-0.02	2.10
15	20	-0.11	0.68	-0.07	1.26	-0.04	0.68
15	30	-0.12	0.98	-0.12	0.88	-0.21	1.32

Continued on next page

Table A.11 – continued from previous page

Parameters		Reddit Posts		Reddit Title		News Aggregator	
15	40	-0.22	1.02	-0.29	1.7	-0.44	1.00
15	50	-0.5	1.12	-0.43	1.37	-0.4	0.99
15	60	-0.61	1.16	-0.4	1.03	-0.52	1.07
15	70	-0.65	0.91	-0.57	1.18	-0.5	0.79
15	80	-0.66	0.85	-0.63	0.96	-0.67	1.40
15	90	-0.65	0.87	-0.67	0.98	-0.71	1.13
15	100	-0.71	0.73	-0.67	0.98	-0.77	1.08
40	4	-0.01	0.52	-0.01	0.4	-0.01	0.15
40	5	-0.01	0.72	-0.01	1.18	-0.01	0.59
40	10	-0.02	0.88	-0.03	0.58	-0.03	1.81
40	20	-0.04	1.38	-0.05	0.93	-0.05	1.43
40	30	-0.06	1.04	-0.08	1.24	-0.07	0.89
40	40	-0.08	0.94	-0.11	1.54	-0.09	1.21
40	50	-0.17	1.09	-0.11	0.65	-0.12	1.38
40	60	-0.14	0.98	-0.14	1.21	-0.14	0.76
40	70	-0.15	0.91	-0.15	0.89	-0.18	1.38
40	80	-0.19	1.2	-0.27	0.70	-0.18	1.31
40	90	-0.32	0.87	-0.23	0.77	-0.28	0.98
40	100	-0.41	0.91	-0.43	0.87	-0.44	1.38
80	4	-0.01	1.98	0.09	1.98	-0.04	0.54
80	5	-0.01	0.34	0.09	0.32	-0.01	0.34
80	10	-0.02	1.61	0.08	1.61	-0.01	0.44
80	20	-0.04	0.56	0.06	0.55	-0.02	1.50
80	30	-0.06	1.57	0.04	1.57	-0.05	0.98
80	40	-0.07	0.79	0.03	0.78	-0.07	0.99
80	50	-0.09	0.83	0.01	0.82	-0.08	0.87
80	60	-0.11	0.91	-0.01	0.90	-0.09	0.97
80	70	-0.12	0.88	-0.02	0.87	-0.11	1.21
80	80	-0.15	1.09	-0.05	1.08	-0.13	1.23
80	90	-0.19	1.24	-0.09	1.23	-0.15	1.09
80	100	-0.17	1.16	-0.07	1.15	-0.16	1.03
100	4	-0.01	0.27	0.09	0.25	-0.01	0.83
100	5	-0.01	1.78	0.09	1.78	-0.01	0.94
100	10	-0.02	0.48	0.08	0.46	-0.02	1.00
100	20	-0.04	0.73	0.06	0.72	-0.04	1.00
100	30	-0.06	1.22	0.04	1.21	-0.06	1.1
100	40	-0.07	0.79	0.03	0.78	-0.07	1.28
100	50	-0.08	1.23	0.02	1.22	-0.08	0.81
100	60	-0.1	1.00	0.00	0.99	-0.09	0.71
100	70	-0.12	1.15	-0.02	1.14	-0.1	1.11
100	80	-0.12	0.89	-0.02	0.88	-0.12	1.12
100	90	-0.14	1.36	-0.04	1.35	-0.15	1.17
100	100	-0.16	0.96	-0.06	0.95	-0.14	1.03

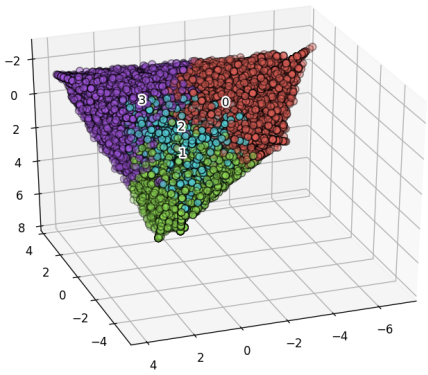

Figure A.2: k-Means clustering run with k=4 and KL distances on documents with 15 topics from the Reddit Titles data set.

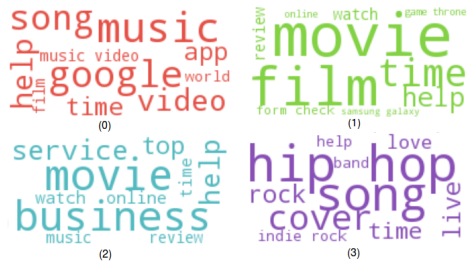

Figure A.3: Word clouds for each cluster in model from figure A.2.

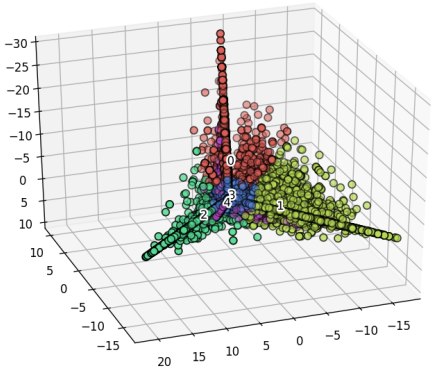

Figure A.4: k-Means clustering run with 5 clusters and euclidian distances on documents with 100 topics from the Reddit Titles data set.

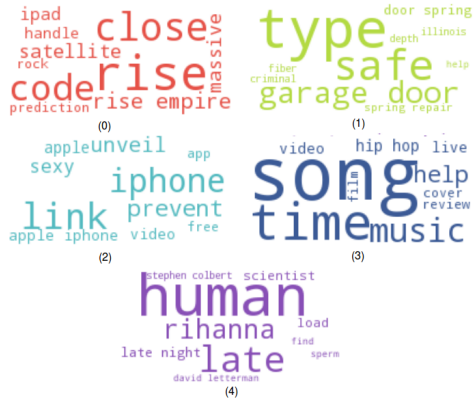

Figure A.5: Word clouds for each cluster in model from figure A.4.

Table A.12: HDBSCAN. (Displayed below.)

Parameters			Reddit Posts		Reddit Title		News Aggregator	
t	**s**	**c**	**Silhouette**	**C-H**	**Silhouette**	**C-H**	**Silhouette**	**C-H**
4	5	5	-0.76	1.02	-0.85	1.00	-0.72	1.02
4	5	10	-0.85	1.00	-0.93	1.00	-0.82	1.02
4	5	15	-0.9	0.96	-0.96	1.00	-0.87	1.04
4	10	5	-0.88	1.01	-0.93	0.96	-0.86	1.05
4	10	10	-0.9	1.00	-0.95	0.96	-0.88	1.05
4	10	15	-0.93	1.01	-0.97	0.98	-0.91	1.05
5	5	5	-0.74	0.98	-0.89	0.98	-0.69	1.01
5	5	10	-0.83	0.99	-0.95	0.97	-0.79	1.02
5	5	15	-0.89	0.96	-0.98	0.96	-0.87	1.05
5	10	5	-0.86	0.99	-0.95	0.98	-0.82	1.01
5	10	10	-0.89	0.99	-0.97	0.96	-0.85	1.02
5	10	15	-0.92	0.99	-0.98	0.95	-0.91	1.03
15	5	5	-0.74	1.00	-0.88	1.00	-0.64	1.01
15	5	10	-0.83	1.00	-0.94	0.99	-0.75	1.02
15	5	15	-0.89	0.98	-0.97	1.01	-0.85	1.01
15	10	5	-0.86	1.01	-0.95	1.00	-0.81	1.03
15	10	10	-0.88	1.00	-0.96	1.00	-0.83	1.02
15	10	15	-0.91	1.00	-0.98	1.00	-0.88	1.01
40	5	5	-0.84	0.99	-0.91	0.98	-0.83	0.98
40	5	10	-0.9	0.99	-0.97	0.96	-0.89	0.97
40	5	15	-0.93	0.98	-0.98	0.91	-0.93	0.99
40	10	5	-0.92	0.99	-0.97	0.95	-0.92	0.97
40	10	10	-0.93	0.98	-0.98	0.96	-0.93	0.98
40	10	15	-0.94	0.97	-0.98	0.89	-0.95	1.03
80	5	5	-0.85	1.04	-0.91	1.04	-0.83	1.02
80	5	10	-0.91	1.05	-0.96	1.06	-0.9	0.97
80	5	15	-0.94	1.03	-0.98	1.11	-0.93	0.96
80	10	5	-0.92	1.01	-0.96	1.31	-0.92	0.97
80	10	10	-0.93	1.01	-0.97	1.34	-0.94	0.98
80	10	15	-0.95	1.03	-0.98	1.45	-0.96	1.00
100	5	5	-0.86	1.13	-0.91	1.87	-0.82	1.11
100	5	10	-0.91	1.05	-0.96	1.91	-0.89	1.05
100	5	15	-0.95	1.08	-0.98	2.22	-0.93	1.03
100	10	5	-0.92	1.37	-0.96	3.82	-0.91	1.20
100	10	10	-0.94	1.31	-0.97	4.39	-0.93	1.17
100	10	15	-0.96	1.3	-0.98	6.19	-0.95	1.18

Threshold	Window Size	Average Similarity	Minimum Similarity	Maximum Similarity	Median	# Positive similarities	# Negative Similarities
100	**3**	0.11687	-0.37818	0.59561	0.10915	152 625.00	27 376.00
	6	0.12267	-0.39351	0.60494	0.11609	154 806.00	25 195.00
	12	0.11868	-0.38232	0.65753	0.11243	155 450.00	24 551.00
	24	0.11540	-0.38207	0.70625	0.10867	154 664.00	25 337.00
300	**3**	0.11686	-0.37818	0.59561	0.10915	152 624.00	27 276.00
	6	0.12267	-0.39351	0.60494	0.11609	154 805.00	25 195.00
	12	0.11868	-0.38232	0.58683	0.11243	155 449.00	24 551.00
	24	0.11540	-0.38207	0.58607	0.10867	154 663.00	25 337.00
400	**3**	0.11686	-0.37818	0.59561	0.10915	152 624.00	27 376.00
	6	0.12267	-0.39351	0.60494	0.11609	154 805.00	25 195.00
	12	0.11868	-0.38232	0.58683	0.11243	155 499.00	24 551.00
	24	0.11540	-0.38207	0.58607	0.10867	154 663.00	25 337.00

Table A.13: Brexit embedding statisticsacross window sizes and threshold

Threshold	Window Size	Average Similarity	Minimum Similarity	Maximum Similarity	Median	# Positive Similarities	# Negative Similarities
100	**3**	0.11993	-0.33085	0.91757	0.11591	156 578.00	23 425.00
	6	0.11685	-0.32295	0.94184	0.10977	155 839.00	24 164.00
	12	0.10902	-0.29563	0.96905	0.10151	155 461.00	24 542.00
	24	0.11093	-0.30915	0.97441	0.10270	155 349.00	24 609.00
300	**3**	0.11973	-0.33063	0.63630	0.11561	156 452.00	23 548.00
	6	0.11689	-0.32386	0.64181	0.10971	155 690.00	24 310.00
	12	0.10906	-0.29618	0.67332	0.10149	155 386.00	24 614.00
	24	0.11093	-0.30964	0.67557	0.10267	155 349.00	24 651.00
400	**3**	0.11973	-0.33063	0.63630	0.11561	156 452.00	23 548.00
	6	0.11689	-0.32386	0.64181	0.10971	155 690.00	24 310.00
	12	0.10906	-0.29618	0.67332	0.10149	155 386.00	24 614.00
	24	0.11093	-0.30964	0.67557	0.10267	155 349.00	24 651.00
1000	**3**	0.09530	-0.37711	0.96551	0.08265	138 289.00	41 712.00
	6	0.10056	-0.36964	0.98063	0.08879	144 934.00	35 067.00
	12	0.09784	-0.33292	0.98927	0.08770	148 732.00	31 269.00
1200	**3**	0.09529	-0.37711	0.69213	0.08265	138 288.00	41 712.00
	6	0.10055	-0.36964	0.67135	0.08879	144 933.00	35 067.00
	12	0.09783	-0.33292	0.69721	0.08770	148 731.00	31 269.00
1500	**3**	0.10511	-0.35231	0.68865	0.09433	145 468.00	34 352.0
	6	0.10649	-0.35487	0.66798	0.09573	148 865.00	31 135.00
	12	0.10177	-0.32320	0.69332	0.09229	151 253.00	28 747.00

Table A.14: Ransomware embedding statistics across window sizes and thresholds.

Word \ Position	#1	#2	#3	#4	#5
americawinter	movie	newsarama	grindhouse	killzone	scifi
amazespider	amaze	spider	newsarama	homestar	tmnt
alternativerock	rock	alternative	indie	punk	pop
altrock	rock	alt	punk	indie	pop
bodyfat	weight	bodyweight	fat	human's	swimmer's
captainamerica	movie	comicbook	newsarama	grindhouse	homefront
dayfuture	past	future	recapping	scariest	movie
deadlifts	squat	dumbbells	deadlift	lifter	flexing
formcheck	deadlift	dumbbells	watch	squat	tweak
futurepast	past	movie	recapping	gametrailers	featurettes
heyguy	thank	please	your	how's	want
gamethrone	throne	baldurs	lufia	laharl	kahless
garagedoor	repair	repairs	maintenance	repairing	repaired
imgur	closeup	doorknob	image	pics	door's
indierock	rock	indie	punk	pop	grunge
megashare	movie	online	movies	video	gamezebo
movieonline	online	streaming	hd	video	niconico
moviewatch	online	watch	video	datapiff	gamezebo
musicvideo	video	music	videos	soundtrack	youtube
obat *indonesian or malaysian	untuk	hati	dalam	malam	cinta
pushup	dumbbell	flexing	workout	workouts	swimmer's
samsunggalaxy	glaxy	samsung	smartphone	touchwiz	nokia
socialmedium	medium	smes	ecommerce	leverages	icts
srilanka	pakistan	bangladesh	karachi	pakistan's	lahore
starwar	movie	homefront	trancers	sequel	flyboys
subreddit	want	anyone's	tweak	thats	please
watchmovie	movie	online	movies	video	datpiff
weightlift	workout	workouts	dumbbells	treadmill	leashes
weightloss	loss	weight	cachexia	dieting	glycemic
youtu	boombox	whammy	shorties	animatic	revved

Table A.15: The 30 words added to the word embedding for Reddit topics with new embeddings 5 closest words.

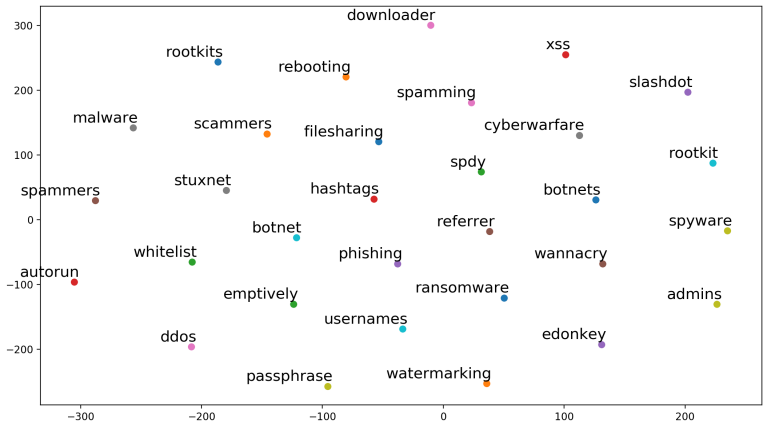

Figure A.6: Visualisation of 30 closest words to Ransomware after dimension reduction, with parameters window size = 3 and threshold = 1000.

Data set	**Reddit**	**News-aggregator**
Average Similarity	0.08582	0.09064
Minimum Similarity	-0.27153	-0.25728
Maximum Similarity	0.69697	0.73680
# Positive Similarities	140 526.47	138 391.24
# Negative Similarities	39 473.53	41 608.76
Median Similarity	0.07847	0.08430

Table A.17: Mean statistics for word embeddings added to vocabulary per dataset.

allergan	gasprice	lindsaylohan	sethrogen
amazespider	gmrecall	malaysiaairline	starwar
americanidol	georgeclooney	meteorshower	stephencolbert
angelinajolie	googleglass	michaeljackson	supremecourt
beyoncejay	guardiangalaxy	milakunis	surfacepro
captainamerica	gwynethpaltrow	mileycyrus	throneseason
chrisbrown	heartbleed	mortgagerate	titanfall
climatechange	heartbleedbug	moviereview	walkdead
dancestar	homesale	paulwalker	wallstreet
dayfuture	justinbieber	robinthicke	windowphone
ebolaoutbreak	kanyewest	rolfharris	wrenscott
futurepast	kardashiankanye	rollstone	zacefron
galaxynote	kimkardashian	samsunggalaxy	
galaxytab	khloe	seasonepisode	
gamethrone	khloekardashian	selenagomez	

Table A.16: Words added when updating embedding vocabulary with the Newsaggregator dataset at threshold = 500 with window size 3.

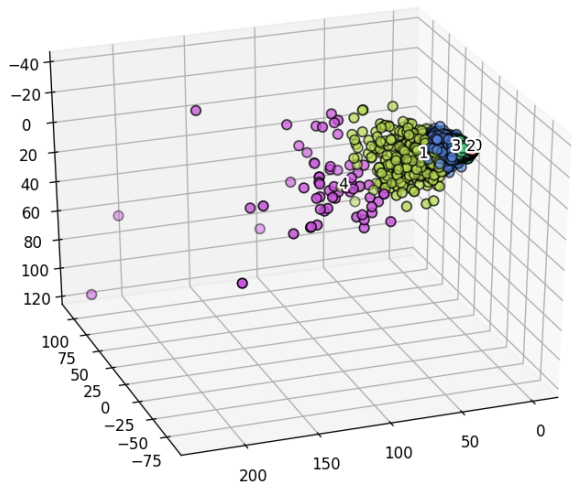

Figure A.7: k-Means clustering run with 5 clusters on documents with 40 topic embeddings from the reddit data set, reduced with pca.

Topic 6	Topic 13	Topic 15	Topic 14	Topic 11
10.0%	**9.1%**	**8.9%**	**8.5%**	**8.3%**
amp	hip	movie	amaze	rock
fat	hop	amp	movie	cover
eat	pop	watch	start	movie
weight	song	hd	squat	time
help	rap	apple	music	song
treatment	trailer	video	train	squat
protein	indie	phone	watch	gym
diet	punk	iphone	amp	watch
people	workout	car	song	video
free	music	film	weight	love
Topic 9	**Topic 4**	**Topic 12**	**Topic 8**	**Topic 2**
7.5%	**7.0%**	**6.8%**	**6.2%**	**6.1%**
movie	amp	rock	movie	mobile
tv	song	help	watch	alternative
song	google	hop	film	music
music	movie	eat	business	online
rock	watch	hip	time	internet
film	weight	hiphop	weight	service
time	online	song	human	market
favorite	stream	fat	online	free
google	free	try	help	company
hiphop	lb	feel	design	weight

Table A.18: Top 10 terms and total percentage of words associated with topics found by TopicVec with $t = 15$ on Reddit Posts.

Topic 16	Topic 19	Topic 36	Topic 33	Topic 34
7.5%	**6.5%**	**6.0%**	**5.6%**	**5.5%**
amaze	watch	weight	online	hiphop
cover	movie	gain	free	hop
spider	star	strength	google	hip
rock	cast	body	app	rap
metal	oscar	lose	mobile	amp
tv	film	muscle	stream	electronic
pop	michael	lift	service	video
folk	actor	bulk	software	dj
dark	episode	cut	download	phone
beautiful	fan	start	amazon	rapper
Topic 14	**Topic 13**	**Topic 40**	**Topic 3**	**Topic 21**
4.5%	**4.0%**	**3.9%**	**3.7%**	**3.4%**
movie	eat	workout	start	protein
film	fat	cardio	exercise	calorie
character	food	fitness	feel	diet
star	meal	routine	time	creatine
actor	diet	gym	run	vitamin
episode	chicken	exercise	routine	supplement
director	drink	bodybuilding	hour	disease
imdb	breakfast	yoga	month	intake
series	snack	crossfit	try	nutrition
war	meat	run	sleep	whey

Table A.19: Top 10 terms and total percentage of words associated with topics found by TopicVec with $t = 40$ on Reddit Posts.

Topic 7	Topic 74	Topic 64	Topic 2	Topic 26
4.2%	**2.7%**	**2.5%**	**2.5%**	**2.1%**
movie	asam	workout	squat	movie
tv	cover	watch	punk	review
hd	song	netflix	hip	film
online	tweet	stream	hop	remix
rock	cancer	leak	edm	girl
video	feel	sore	hiphop	teenage
amp	people	movie	deadlift	boy
official	amp	chernobyl	rock	turtle
download	help	telemarketing	garage	trailer
watch	woman	time	daft	shakey
Topic 77	**Topic 70**	**Topic 30**	**Topic 9**	**Topic 54**
2.1%	**2.1%**	**2.0%**	**1.8%**	**1.7%**
ft	workout	amaze	weight	rock
watch	photo	buy	muscle	amp
lift	support	help	loss	love
amaze	hotmail	system	bodyweight	song
rise	maah	eat	repair	feat
movie	xperia	computer	body	rap
wind	program	learn	lift	hiphop
squat	dumbbell	electronic	cardio	bad
gym	chromecast	jenis	squat	video
workout	gmail	sev	lb	que

Table A.20: Top 10 terms and total percentage of words associated with topics found by TopicVec with $t = 80$ on Reddit Posts.

Topic #	First run 5	Second run 5	First run 40	Second run 40
Iterations best run	**65**	**70**	**194**	**229**
Train precision micro average	Recall: 0.697 F1: 0.697 Accuracy: 0.697	Recall: 0.695 F1:0.695 Accuracy: 0.695	Recall: 0.791 F1: 0.791 Accuracy: 0.791	Recall: 0.796 F1:0.796 Accuracy: 0.796
Train precision macro average	Recall: 0.556 F1: 0.542 Accuracy: 0.697	Recall:0.640 F1:0.568 Accuracy: 0.695	Recall: 0.772 F1: 0.741 Accuracy: 0.791	Recall: 0.727 F1: 0.745 Accuracy: 0.796
Test precision micro average	Recall: 0.749 F1: 0.749 Accuracy: 0.749	Recall:0.749 F1:0.749 Accuracy:0.749	Recall: 0.873 F1: 0.873 Accuracy: 0.873	Recall: 0.847 F1: 0.847 Accuracy: 0.847
Test precision macro average	Recall: 0.624 F1: 0.609 Accuracy: 0.749	Recall: 0.682 F1:0.641 Accuracy:0.749	Recall: 0.780 F1: 0.791 Accuracy: 0.837	Recall: 0.795 F1: 0.805 Accuracy: 0.847

Table A.21: Performance on classification task for topic embeddings after word embeddings are updated from training corpus.

Topic 14	Topic 9	Topic 6	Topic 5	Topic 10
17.1%	**12.6%**	**10.0%**	**9.5%**	**7.8%**
kardashian	billion	price	google	amaze
kim	million	stock	microsoft	yellen
kanye	buy	dollar	apple	spider
miley	share	ecb	android	batman
cyrus	growth	market	samsung	superman
beyonce	deal	gain	update	harry
bieber	sale	rate	iphone	potter
justin	gm	inflation	smartphone	sequel
wed	bank	euro	ipad	comic
jay	china	tax	htc	rowling
Topic 2	**Topic 8**	**Topic 3**	**Topic 13**	**Topic 7**
6.8%	**5.5%**	**5.5%**	**5.2%**	**4.8%**
galaxy	dy	blood	launch	drug
apple	episode	trueblood	malaysia	cancer
ceo	season	moon	plane	fda
samsung	throne	ice	court	food
phone	game	cinco	airline	alzheimer
facebook	star	water	mh	treat
amazon	vii	white	china	autism
guardian	war	grey	flight	brain
review	trailer	red	supreme	alcohol
comcast	mother	mayo	search	treatment

Table A.22: Top 10 terms and total percentage of words associated with topics found by TopicVec with $t = 15$ on News Aggragator.

Topic 30	Topic 11	Topic 16	Topic 23	Topic 12
9.7%	**6.2%**	**6.1%**	**5.9%**	**4.6%**
star	bachelorette	rate	oil	million
actor	andi	ebola	stock	deal
movie	yellen	risk	gm	billion
jolie	coachella	growth	price	bn
clooney	gosling	percent	obamacare	ipo
colbert	dorfman	cancer	iraq	stock
angelina	ryan	increase	crude	pay
george	potter	report	market	trade
rogen	rowling	rise	gas	bnp
tv	emma	unemployment	recall	price
Topic 7	**Topic 19**	**Topic 13**	**Topic 17**	**Topic 6**
4.1%	**3.7%**	**3.4%**	**3.1%**	**3.0%**
apple	euro	beyonce	amazon	season
google	mortage	cyrus	xbox	boxoffice
facebook	ecb	miley	microsoft	box
android	eurozone	bieber	netflix	record
twitter	dollar	jay	buy	recap
app	bank	justin	prime	office
phone	rate	tour	stream	captain
microsoft	argentina	selena	service	episode
ipad	loan	gomez	ebay	finale
io	bln	kanye	console	america

Table A.23: Top 10 terms and total percentage of words associated with topics found by TopicVec with $t = 40$ on News Aggragator.

t	k	Silhouette	C-H	t	k	Silhouette	C-H
5	3	0.87	237 264.19	**15**	3	0.86	184 865.84
	4	0.83	239 977.27		4	0.81	175 075.25
	5	0.77	231 858.39		5	0.74	160 303.42
	10	0.65	185 237.88		10	0.61	112 603.29
	20	0.32	161 965.16		15	0.30	91 126.27
	30	0.28	152 548.54		20	0.28	79 904.75
	40	0.28	141 592.09		30	0.24	68 037.84
	50	0.28	133 461.41		50	0.20	55 333.80
	70	0.24	122 592.47		70	0.17	49 255.04
	80	0.24	119 454.71		80	0.17	46 625.36
	100	0.23	111 254.96		100	0.16	42 347.04
t	**k**	**Silhouette**	**C-H**	**t**	**k**	**Silhouette**	**C-H**
40	3	0.85	168 564.12	**80**	-	-	-
	4	0.80	156 643.46		4	0.80	153 553.99
	5	0.73	141 552.61		5	0.73	138 291.22
	10	0.61	94 716.06		10	0.61	93 684.04
	20	0.26	64 058.61		20	0.21	63 135.86
	30	0.24	52 402.56		30	0.21	52 435.05
	40	0.20	45 341.13		40	0.21	45 868.97
	50	0.22	40 812.43		50	0.19	41 674.59
	70	0.16	35 241.63		70	0.19	36 112.17
	80	0.16	33 396.93		80	0.14	33 869.38
	100	0.14	29 775.43		100	0.14	30 208.40

Table A.24: TopicVec clustering results for k-Means on Reddit Posts documents across several values for k and t.

t=5	243 943	23 494	7 784	2 240	714	486	161	68	42	4
t=15	243 329	23 617	8 041	2 347	769	522	172	66	43	3
t=40	243 616	23 747	7 880	2 224	745	459	159	59	43	4

Table A.25: TopicVec cluster sizes for RedditPosts with $k = 10$ across topic embeddings.

t=5 k	Silhouette Coefficient	C-H Index
3	0.32	348 858.11
4	0.29	311 166.90
5	0.28	283 059.56
10	0.25	254 901.03
20	0.21	207 029.35
30	0.20	178 462.07
40	0.19	158 844.15

Table A.26: TopicVec clustering results for k-Means with t=5 with News Aggregator data set.

k	No update		Updated	
	Silhouette	**C-H**	**Silhouette**	**C-H**
3	0.18	161 058.74	0.18	165 756.60
4	0.20	136 176.86	0.20	136 176.86
5	0.15	118 221.88	0.15	118 221.88
10	0.17	87 688.45	0.17	87 688.45
20	0.14	62 281.25	0.14	62 281.25
30	0.14	51 585.87	0.14	51 585.87
40	0.13	43 871.92	0.13	43 871.92

Table A.27: TopicVec clustering results for k-Means with t=40 on News Aggregator data set, before and after updating the embedding vocabulary.

k=30			k=50		
t=5	**t=15**	**t=40**	**t=5**	**5=15**	**t=40**
123 897	129 575	150 720	67 022	96 814	124 992
56 063	47 559	65 638	48 149	47 793	43 614
43 664	38 083	28 710	39 006	29 284	35 747
18 689	28 675	10 408	33 205	21 847	21 491
10 655	10 204	7 866	25 993	20 407	15 666
6 196	8 319	4 359	16 646	15 001	10 599
4 100	4 293	3 030	13 277	11 438	4 537
3 951	3 564	2 404	6 532	7 172	3 819
3 638	2 294	1 460	5 099	5 803	3 729
2 437	1 384	1 314	4 526	4 106	2 331
1 460	1 334	893	3 233	3 859	2 205
1 049	1 326	531	3000	2 565	1 418
777	546	451	2 844	2 390	1 274
729	495	326	1 482	1 456	1 166
407	445	232	1 465	1 414	1 153
370	249	148	1 214	1 151	786
183	213	125	998	1 020	578
178	95	115	901	882	565
168	77	49	700	826	530
103	54	40	699	766	503
72	45	33	534	759	398
39	29	25	346	370	349
31	21	22	332	284	254
24	16	13	320	265	231
21	11	11	276	204	192
14	10	4	201	163	163
10	8	4	194	153	110
6	7	2	145	118	90
4	3	2	118	99	87
1	2	1	105	96	87
			81	75	41
			58	69	41
			55	55	29
			31	46	28
			30	45	28
			27	39	23
			19	23	17
			13	15	16
			13	13	11
			10	10	8
			7	9	7
			7	8	6
			6	6	4
			5	6	4
			4	4	2
			3	2	2
			2	2	2
			1	2	1
			1	1	1
			1	1	1

Table A.28: Sorted sizes of clusters when clustering with k-Means on TopicVec documents, across values for t and k.

Topics	Min samples	Min cluster size	Silhouette	CH	# Clusters	# Outliers
5	1	5	-0.35	8.89	17 924	121 005
5	1	10	-0.52	14.02	5 549	146 961
5	1	15	-0.59	15.33	2 588	145 155
5	5	5	-0.70	4.17	4 990	212 488
5	5	10	-0.72	5.89	1 888	209 811
5	10	5	-0.75	5.45	1 539	210 383
5	10	10	-0.73	10.34	745	209 563
5	10	15	-0.71	11.6	450	197 306
5	10	20	-0.70	13.53	301	194 718
5	15	10	-0.72	9.33	435	202 701
5	15	15	-0.71	11.27	326	202 728
5	15	20	-0.69	15.09	233	201 502
5	20	10	-0.71	10.06	295	207 385
5	20	15	-0.70	12.68	234	206 919
5	20	20	-0.69	13.9	206	207 558
40	1	2	-0.52	1.00	52 669	134 215
40	1	5	-0.84	1.01	7 615	212 970
40	1	15	-0.96	1.02	927	243 249
40	5	10	0.97	0.99	744	259 144

Table A.29: TopicVec clustering results with HDBSCAN on Reddit Posts dataset across topics and parameters.

Predictor	MAP@3	MAP@5	MAP@10	MAP@30	MAP@TOTAL	RA
LDA1	0.47	0.50	0.47	0.40	0.35	0.53
LDA2	0.48	0.50	0.48	0.41	0.30	0.35
LDA+HDBSCAN1	0.81	0.81	0.81	0.81	0.13	-0.28
LDA+HDBSCAN2	0.73	0.73	0.73	0.70	0.14	-0.18
LDA+KMEANS1	0.71	0.70	0.64	0.55	0.22	0.34
LDA+KMEANS2	0.33	0.34	0.33	0.30	0.14	0.00
LDA+KMEANS3	0.83	0.83	0.83	0.83	0.12	-0.28
LDA+KMEANS4	0.63	0.64	0.61	0.45	0.30	0.46
TV	0.57	0.57	0.54	0.46	0.38	0.52
TV+KMEANS1	0.87	0.86	0.87	0.87	0.13	0.64
TV+KMEANS2	0.41	0.41	0.40	0.32	0.20	0.22
ORIGINAL	0.22	0.25	0.26	0.22	0.15	0.01
RANDOM	0.24	0.27	0.28	0.23	0.15	0.00

Table A.30: Results from models selected for comparison after predicting documents to users.

t	k	MAP@3	MAP@5	MAP@10	MAP@30	MAP@TOTAL	RA
5	**4**	0.82	0.82	0.82	0.81	0.13	-0.27
5	**5**	0.73	0.73	0.74	0.72	0.13	-0.27
5	**10**	0.65	0.65	0.64	0.62	0.13	-0.17
5	**40**	0.22	0.22	0.22	0.20	0.13	-0.10
40	**5**	0.72	0.72	0.72	0.70	0.13	-0.23

Table A.31: TopicVec prediction evaluation results for models not selected for comparison.