CONDENSING INFORMATION: FROM SUPERVISED TO CROWDSOURCED LEARNING

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Ruben Sipos

August 2014

CONDENSING INFORMATION:

FROM SUPERVISED TO CROWDSOURCED LEARNING

Ruben Sipos, Ph.D.

Cornell University 2014

The main focus of this dissertation is new and improved ways of bringing high quality content to the users by leveraging the power of machine learning. Starting with a large amount of data we want to condense it into an easily digestible form by removing redundant and irrelevant parts and retaining only important information that is of interest to the user. Learning how to perform this from data allows us to use more complex models that better capture the notion of good content.

Starting with **supervised** learning, this thesis proposes using structured prediction in conjunction with support vector machines to learn how to produce extractive summaries of textual documents. Representing summaries as a multivariate objects allows for modeling the dependencies between the summary components. An efficient approach to learning and predicting summaries is still possible by using a submodular objective/scoring function despite complex output space.

The discussed approach can also be adapted to **unsupervised** setting and used to condense information in novel ways while retaining the same efficient submodular framework. Incorporating temporal dimension into summarization objective lead to a new way of visualizing flow of ideas and identifying novel contributions in a time-stamped corpus, which in turn help users gain a high level insight into evolution of it.

Lastly, instead of trying to explicitly define an automated function used to condense information, one can leverage **crowdsourcing**. In particular, this thesis

considers user feedback on online user-generated content to construct and improve content rankings. An analysis of a real-world dataset is presented and results suggest more accurate models of actual user voting patterns. Based on this new knowledge, an improved content ranking algorithm is proposed that delivers good content to the users in a shorter timeframe.

BIOGRAPHICAL SKETCH

Ruben Sipos was born in October 1985 in Ljubljana, the capital city of Slovenia. In 2004 he started his studies at University of Ljubljana as a part of newly formed joint interdisciplinary program carried out by the Faculty of Computer and Information Science and the Faculty of Mathematics and Physics. He graduated in 2009 with a bachelor's degree in Computer Science and Mathematics. In the Fall of 2009 he began his Ph.D. studies at Cornell University. He received his Doctor of Philosophy degree in Computer Science in August 2014.

ACKNOWLEDGEMENTS

There is always this one key brick that everything else rests on. If you removed it, then the pages you are reading right now would fly apart. So, first and foremost my sincere thanks go to my Ph.D. advisor Thorsten Joachims. Over the years I was privileged to receive his guidance, in professional and personal matters, insights, time and limitless amounts of patience. Whenever I felt stuck or lost I had someone I could turn to and for that I am extremely grateful.

A thesis does not grow without a committee, and therefore I would like to say thank you to all the rest of my committee members: Noah Snavely, Claire Cardie and Arpita Ghosh. In addition to their direct influence, they also inspired me with their exciting research. A special thanks goes to Arpita, who invested a great amount of time and effort in our meetings and discussions.

A large amount of credit also goes to two of my close collaborators: Pannaga Shivaswamy, who was a postdoctoral researcher here at Cornell, and Adith Swaminathan, my fellow Ph.D. student. Thank you for sharing your ideas, listening to my ramblings and also your time spend on actually materializing some content of our papers.

Without the right environment, one suffers greatly. Well, I am happy to tell you that I was surrounded by great people in stellar location. Cornell University provided plentiful opportunities for expanding my knowledge and Ithaca a quite and beautiful surroundings. Many thanks for all heated (and not so heated) discussions go to my office mates and fellow students. And of course many many professors for their great lectures in classes and outside of them.

One of the big influences was Machine Learning Discussion Group. It kept fuelling my thirst for knowledge and provided an opportunity to discuss the far-fetched ideas. And sorry to all non-voluntary volunteers; but you did make a

difference.

Let me say thank you and once more thank you to all my friends here at Cornell and back home, who made sure I kept my sanity through sun and hail. Also, I can not express how thankful I am for all my parents have done for me and all their sacrifice and encouragement. Furthermore, I have great siblings who were always there when I needed them.

Last but definitely not least, I have to acknowledge the financial support I received that made all this possible in the first place. This research was funded in part by NSF Awards IIS-0812091, IIS-0905467, IIS-1142251, IIS-1217686 and IIS-1247696, and the Cornell-Technion Research Fund.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part I

Overview and Preliminaries

CHAPTER 1

INTRODUCTION

Data. Digital technology revolutionized how we collect, store and share data. Even if we focus only on text directly targeted for human consumption, there is still a staggering amount of it and its growth is rapidly accelerating. In the last 25 years the Web became the main hub for information exchange used by businesses, professionals and a booming number of ordinary everyday users. However, the benefits of shared knowledge could now be approaching a wall: the limit imposed by information overload. Because the amount of interesting and relevant content already exceeds our ability to consume it in its raw form we need a way to select or condense the data and present it to the user in an easy to consume form that still retains the information the user actually wanted.

Looking back, we can clearly see one big building block towards this goal: web search engines. When the web became too complex to navigate, keyword search and result ranking saved the day. Nowadays we have extremely well polished search engines that serve the users well – for that particular targeted group of needs. However, the web has long ago stopped being only a collection of personal and corporate pages with scattered bits of information. New forms of content have evolved over time: on the web and web-enabled content networks we can find news articles, blogs and microblogs, forum and social posts, product, hotel and restaurant reviews, personal messages and email, even patents and research publications to name just a select few in text domain. Information retrieval solutions still allow us to find and accessing individual documents even within these new content types. However, we can not blindly hope that this covers all the diverse user needs (which are also evolving in step with new content).

Some of those user needs could be addressed by condensing information. Multiple news articles, blogs or posts can be condensed into a short passage by removing redundant information and extracting informative fragments. An insight into a larger collection might be gained by constructing timelines of how the contents evolved over time. Ranking user-generated content based on users' perception of it can help promote contributions of high quality contributions.

This thesis provides a small step forward towards addressing the challenge of diverse user needs. First, it explores summarization – instead of picking the right document, we select the right information and present it to the user in a concise package. Secondly, because the data now lives in the digital domain (instead of in the physical form of paper) we can take advantage of how easy it is to manipulate and process – machine learning provides a way to tune complex summarization and ranking models by using data instead of hard-to-express high-level expert knowledge. And thirdly, due to immense number of users and even if only a tiny fraction of them is altruistic, we can still obtain large amounts of user feedback to replace or augment expensive labeled examples.

1.1 Summarization

Summarization in not an alternative to search and ranking but a complementary task. While web search is extremely good at locating a specific piece of information, summarization provides a way to combine multiple sources and addresses some of the needs for a more macroscopic view of a document collection and supports discovery of global insights.

The framework used throughout Parts II and III is based upon submodular scoring functions. Summaries are represented as sets of items and computed by

maximizing a scoring function defined on them. By requiring that this function is submodular we are able to find efficient solutions: the maximization can be solved using greedy algorithm with a constant factor approximation guarantee. Looking at the related literature, we can find many many recent advances in summarization using the same core idea.

One of the novel contributions of this thesis are new problem formulations and corresponding scoring functions that produce summaries addressing currently unanswered user needs. Chapter 4 describes how comparing two products based on their reviews can be viewed as constructing a summary that helps inform purchasing decisions. Looking at a collection of documents that evolved over time, Chapter 5 shows how high level visualizations such as timelines of information spread can be constructed in the same framework.

1.2 Learning

Another big emphasis of this thesis is learning. Simple models are unlikely to capture what is a good summary with sufficient fidelity. As the complexity of the models grows, it becomes infeasible to manually tune the parameters. In a supervised setting, machine learning solves the problem of finding the optimal model parameters by learning from examples.

Document summarization as described in Chapter 3 (applicable to, for example, news articles) uses structured prediction. This means that after learning the model parameters, the summary for a given collection is predicted as a multivariate object. This enables us to capture dependencies between elements within the summary itself. Compared to some other related work, we can now, for example,

explicitly control for redundancy. If we were to select sentences independently in the case of extractive summarization we need additional mechanisms to remove near similar sentences. However, in the structured prediction framework, we can incorporate non-redundancy directly into optimization objective.

The resulting union of submodular scoring function and structured prediction in Chapter 3 pushed the state-of-the-art in extractive multi-document summarization another few steps forward.

1.3 User Feedback

One major hurdle for the learning approaches (at least in this domain) is obtaining quality labeled data. Example summaries created by human annotators are expensive. And in some cases it is not even clear what is a good label as in the case of creating timelines. However, there is another way of determining what is good content that should be surfaced for users.

Looking at today's web we see that users are not only contributing content (e.g. forum posts) but are also at the same time willing to supply peer feedback on the submitted content (e.g. thumbs-up votes on posts). Given large amount of users, we can collect large amounts of user feedback even if this requires additional engagement and thus not everyone opts to do so.

The last part of this thesis looks into how we can exploit this free resource to bring relevant content to the users. Analysis of a real-world dataset (helpfulness votes cast by the users on product reviews) is presented first with the goal of trying to understand what are users actually trying to tell us. It shows presence of

non-trivial biases in the votes. This new finding is then used to improve content ranking algorithm.

1.4 Thesis organization

This thesis is structured as follows. Chapter 2 presents overview of related work in other literature. It discusses the field of summarization, main approaches and different ways of condensing information. Furthermore, it looks at the approaches used for ranking user contributed reviews and reviews current understanding of user voting behaviour in the case of helpfulness votes on product reviews.

Multi-document summarization that combines submodular scoring function and structured prediction is described in Chapter 3. Evaluation was performed on a collection of news articles and demonstrated new state-of-the-art performance. Chapter 4 shows, how we can, by simply changing the objective, create comparative summaries of products based on their corresponding user reviews.

Next, Chapter 5 moves to unsupervised setting and creates high-level overviews of a document collection. In particular, an approach for building timelines (and other views) is described. Retaining the submodular framework and defining new time-aware scoring functions results in macroscopic summaries that outperform the baselines.

Lastly, learning what is good content from the users is explored in Chapter 6. An in-depth analysis of real-world voting data for product review helpfulness is presented. A context-dependent bias is identified and then this new insight is used to improve the content ranking algorithm in Chapter 7.

CHAPTER 2

BACKGROUND

2.1 Automated Summarization

This section looks at related work pertaining to automated summarization. It reviews traditional algorthmic approaches to summarization and describes some of the different forms the summaries can take.

2.1.1 Algorithmic Approaches to Summarization

One of the widely known approaches for summarization is Maximal Marginal Relevance [10]. It uses a greedy approach for selection and considers the trade-off between relevance and redundancy. Later it was extended [27] to support multi-document settings by incorporating additional information available in this case. Moreover, this can be reformulated as a knapsack problem and solved using dynamic programming [63].

Another avenue of approaches is graph based [66]. LexRank [21] uses sentence similarities and selects sentences based on graph eigenvector centrality. Other notable graph based sentence scoring methods are TextRank [67], PageRank [8], HITS [38] and CIG [31].

Iterative reinforcement approach of [109] simultaneously extracts summary and keywords aiming to use one to boost the other. In this graph based approach, sentences are related by content similarity, words by knowledge-based or corpus-based approach and words in sentences by their relative importance. The main

assumption is that saliency should be propagated across links.

Often used tools in summarization are coverage functions. By definition, they nicely fit the task of selecting content to be included in the summary. Furthermore, the have a diminishing returns property which also fits the intuition about summaries repeated content does not increase the utility. Because they fall into the class of submodular functions, there exist efficient algorithms for finding approximate solutions [35] that exhibit good behavior in practice.

Although recent research used set functions in conjunction with completely independent elements (i.e. we can select any subset of sentences), the maximization problem using submodular objective functions can still be efficiently solved in a more general setting of matroids [9]. This allows us to pose additional constraints on how the summary is supposed to look, but to my current knowledge has not yet been sufficiently explored.

Diversity is one of the important aspects when doing information retrieval. One way of achieving it is by using word coverage approaches [125, 102]. In this paper they measure diversity based on coverage of subtopics. Because subtopics are not given for novel documents (only for the training set), they use structural SVM to learn how to select documents covering diverse set of topics in the absence of labels.

A similar idea was applied to document summarization [44]. However, in this case they achieve diversity by incorporating additional constraints into QP for structural SVM instead of solving a coverage problem. They add constraints for covering subtopics, promoting diversity (by placing lower weights on common features) and balance (the amounts of subtopics should be roughly equal). Fur-

thermore, because they do not use coverage objective (which is submodular and efficiently solvable) they propose a novel way of solving the optimization by use of independence graphs (which restrict the space possible additions to the summary to sentences that are not similar to the previous ones).

Determinantal point processes [39] can also be applied to document summarization. In this paper they describe how to apply DPP in supervised setting using feature-based parameterization. The objective is submodular but not monotone (therefore the greedy optimization algorithm has no formal approximation guarantees) and the learned models have elegant probabilistic interpretation.

A coverage based approach to summarization in the blogosphere is presented in [20]. They define coverage in a probabilistic way where each post tries to cover a feature with some probability. This results in diminishing returns property as we add more posts. Furthermore, the notion of feature coverage by a post is softened and defined on top of a generative model providing probability of covering it. The final objective is submodular and can thus be efficiently solved. The authors also provide a way of personalization by introducing additional weights representing preferences (for features).

Another submodularity based approach [50, 49] uses pair-wise sentence similarities as the basis for scoring the summary. The objective tries to maximize similarities between the sentences in the summary and the rest of them (promoting coverage) while keeping similarities within the summary low (avoiding redundancy). The terms are traded off using a manually tuned parameter.

Supervised Approaches

In the supervised setting, several early methods [40] made independent binary decisions whether to include a particular sentence in the summary or not. This ignores dependencies between sentences and can result in high redundancy. The same problem arises when using learning-to-rank approaches such as ranking support vector machines, support vector regression and gradient boosted decision trees to select the most relevant sentences for the summary [65].

A solution is to introduce dependencies between selected elements forming the summary. HMMs produce summaries by transitioning between sentences included in the summary and skipped ones while following the original sequence of sentences in the source documents [14]. If we want to employ more powerful features (depending on the whole document instead of only the current sentence) we can use CRFs [91]. However, summaries are not necessarily chains of sentences and therefore we cannot model all types of dependencies between them.

While doing extractive summarization we often include unnecessary words which are part of the selected sentence. To combat this we can use compression (by e.g. using MIRA [62]), i.e. removing unnecessary parts of the sentence before including it in the summary. Authors of [6] use structural SVM framework to learn model parameters and ILP to encode compression model. Arbitrary parts of the parse tree can be cut to compress the sentence while still preserving some grammatical structure. Another model employing compression is SEARN [17] using a vine-growth model to compress and search to construct a policy for generating summaries.

Evaluating summaries

Evaluating the quality of automatically generated summaries is not easy and using human judges is expensive. Rouge [48] has become one of the most widely used performance measure because it shows good correlation with human assessments and allows for automatic evaluation (needed in most supervised approaches to document summarization). The most popular variants of Rouge score are based on unigram and bigram overlap between generated summaries and gold standard.

Clustering-based Approaches

A parallel to summarization can be found in selecting representative elements of clusters [24]. Clustering results in diversity of selected representatives. In this paper authors explore different ways of selecting representatives for each cluster: average, most and least typical elements (according to distances between elements). All three have intuitive use cases. The average gives us the sense of overall cluster content, most typical highlights differences between clusters and least typical are useful for achieving consensus between clusters.

An example of clustering based document summarization approach is MEAD [78]. It is a centroid-based method in which sentence scores are computed based on sentence-level and inter-sentence features.

Targeted specifically at summarization ColabSum [107] combines document clustering with a graph based approach based on sentence similarities. Using clustering to achieve diversity and then avoiding redundancy by selecting a representative sentence for each cluster [75] can be seen as an extension of MMR with diversity.

Some summarization methods based on LSA are also designed on similar prin-

ciples as clustering based ones [97]. After we obtain topics, we can construct a summary by selecting the most informative sentence for each topic. A few other generative models for summarization are explored in [28].

Generative models inspired by LSA can also be extended to apply to summarizing user comments [57]. The news article content and comments posted by users are both taken into account. Comments are clustered into groups by using discovered topics. Finally, they select representative comments to create a summary. Note that good comments are considered to be those that closely match content of the news article (instead of e.g. expressing opinions about the content).

Basic Building Blocks

In many summarization approaches we treat words (or e.g. bigrams) as the basic units. Instead, we could use relations [22] between entities. All pairs of named entities or frequent nouns are extracted as relations preserving the word order. When building the summary we select sentences with the highest sum of all relations it contains (and each relation is weighted as a sum of its constituents). Summary is built greedily as coverage to avoid redundancy.

Another underutilized type of element is temporal expressions within text. In contrast to publication time, these offer finer granularity and refer to particular actions/events. They also give information about story time instead of meta-information about document creation. The paper [61] describes how we can extract this information. They also provide a way of comparing two sets of intervals in terms of coverage.

2.1.2 Different Ways of Summarizing

Authors of [90] propose a few different ways of summarizing, all based on minimum dominating set of the sentence graph. Their method can be applied to produce a generic summary, a query focused summary (by selecting a dominating set that also minimizes distances to the query), an update summary (by first constructing a dominating set for the original set and then extending it for the new content) and a comparative summary (by finding a complementary dominating set). The same types of summaries can also be produced based on term coverage and textual unit similarity [43].

Summarization is not necessarily restricted to only textual content. The approach of [120] creates timelines of news that contain textual as well as visual elements. They use an iterative approach based on mutual reinforcement and co-ranking.

Recommendation Systems

Instead of querying for relevant documents by using keywords, we can pose query as a small set of documents [19]. Relevant papers are selected based on influence (across all concepts and all query documents toward selected summary), while maximizing relevance and diversity. Influence is defined based on citation graph instantiated for each concept with edge weights representing probability of a direct influence between two papers for that concept.

Also looking at the influence between papers is [89]. They look at identifying original ideas contributed by a paper added to a corpus. Using impact and novelty they define a statistical model of passage impact.

Content based recommendation can be combined with collaborative filtering in a hybrid approach [103]. They explore a few different ways of combining the two (e.g. using output of one as input for the other, running them in parallel and combining the produced rankings).

Events and Timelines

When events happened can be used in event-based summarization [98, 2] to construct a summary [118]. Events (e.g. burnt from Yesterday a fire burnt most of the forest.) are placed on a timeline (split into intervals e.g. days) according to accompanying temporal expressions (e.g. yesterday or last week). Weights are then computed using a similar scheme as tf-idf and used to rank the sentences containing those events.

A summary can also be constructed as a timeline of events [121, 101]. Evolutionary Timeline Summarization aims to produce individual summaries of each date on a timeline while achieving sufficient content coverage, coherence of the timeline and cross-date diversity. They define utility function for summaries which is a linear combination of relevance (towards query), coverage (towards local and global collections), coherence (between neighboring days) and diversity terms. To achieve balance across the whole collection and individual days they pose the utility function as linear combination of global and sub-collection utilities.

Method described in [86] provides a way of automatically finding coherent chain linking together news articles. An efficient algorithm for connecting two fixed endpoints is described. Good chains are defined as those with no weak links (therefore they do not average over chain but optimize the minimum) and with influence between consequent articles.

Building on top of the observation that summaries usually select sentences in temporal sequence, authors of [46] explored how good summaries are constructed when the collection spans a time interval. They discovered that sentences from the earliest and the most recent articles get selected more often than the rest. Grouping documents by time features, clustering and selecting representatives for each of them provides a weighting of sentences later used to construct a summary.

Many summaries of events in news present a flat structure. A more informative view would present threads of events within a topic that capture (causal) dependencies between them [71]. They discuss possible ways of modeling the dependencies between events, such as constructing MST based on similarities between events. Summaries are obtained by clustering the news to obtain events, finding dependencies between them and creating a relational structure.

Moving in a similar direction, TSCAN [11] models topic anatomy by forming an evolution graph of the topic. Topics are obtained from the eigenvectors of a term - temporal block (consecutive parts of the documents, ordered by time) association matrix and events by looking at the composition of eigenvectors. Graph of the topic is created by linking temporally and contextually similar events. A different approach considers intra-corpus relations and describes influence between documents [88].

Another possible view is looking at how topics change over time [115]. LDA model is extended with additional observed variable representing documents timestamp. Topic content is still unchanging over time, but the amount of documents covering that topic can vary. Experiments show that the model can capture different patterns, such as bursts of documents on particular topic at some time point and gradual changes over the whole time interval.

Citation sentences

In the scientific literature domain we can summarize papers by looking at what they are cited for [77]. Assuming that citation sentences (i.e. sentences surrounding the citation to the target paper) cover some part of the information in the cited paper, they try to select a set of citation sentences best describing the target paper. This way we obtain a summary of what other think about me.

Single sentence summaries

In some cases we want a really short summary. For example, on Amazon and Rotten Tomatoes we can find a single sentence representing the overall opinion on this item, which gives a very concise insight into shared opinion about it. A supervised approach to this is proposed in [5]. They also explore which features are helpful in selecting such sentences. Location-wise, the first sentence of a paragraph is the most likely to be chosen (and this also holds for generic summarization of news articles [95]). Another good location for such sentences is towards the end of the review. Furthermore, some words (mostly words directly expressing emotion) are more common in the summary quotations than the rest of the text.

2.1.3 Summaries with Internal Structure

In contrast with extractive document summarization where we select sentences to capture the essence of documents, a different approach might be more appropriate for product reviews. A more structured summary [30] can be constructed by first identifying product features present in the reviews (in this case authors only look for explicitly mentioned ones). Next, they identify opinionated sentences, their polarity and to which feature they refer. Summary then consists of a list of

features, each separately listing positive and negative opinions.

A similar kind of summary can be generated in the movie reviews domain [130]. An interesting peculiarity of this domain is that proper nouns (e.g. people names actors) can be features too. Moreover, a name may be expressed in different forms. To deal with this problem they incorporate cast library as additional knowledge.

In addition to providing only relevant opinion snippets for each aspect, we can also try and predict a rating [56]. They take comments consisting of description and overall rating (from feedback on eBay) and construct separate ratings for different aspects (e.g. shipping, communication) and highlight phrases providing the evidence (e.g. fast delivery).

In a similar fashion [114] tries to discover aspects, associated opinion and relative weight placed on them with respect to the whole review. This can then be further used to construct summaries pointing out particular good or bad points. Another useful application is analyzing user behavior. For example, expensive hotels get high ratings mostly due to good service, while cheap hotels get most bad ratings due to low cleanliness. Furthermore, we can use this to personalize rankings such that weights assigned to aspects in the presented reviews align with his preferences.

Using product specification to mine features [64] is also possible. Multiple product specifications are clustered to obtain a hierarchy of features. They use association between features (or, cleverly, in some cases units of measurements) and opinion words to extract implicit features. Furthermore, they avoid sentiment classification and only extract frequently occurring words (for example small and thin for size instead of positive).

Instead of presenting a collection of sentences supporting positive and negative opinions for each aspect, we can instead present to the user a score for each one of them [122]. The main benefit is more compact summary which allows user to read them faster and compare more products. However, there is an implicit need for trust in the scoring system and its alignment with users personal preferences. Such a summary can be constructed by extracting product features and performing sentiment classification. Then, using the combination of both we can assign score to the features.

The approach of [127] takes a different route and does not display opinions grouped under facets by listing positive and negative sentences. Instead, they identify the main topics and rank them according to their importance. Then they select sentences to cover the content of those topics. They do not use LSA but a simpler frequent word sequence based approach. In order to avoid redundancy they use MMR.

Set in an information retrieval setting, CREST [41] provides a way to quickly familiarize ourselves with the consensus opinion on product features in a large corpus of reviews. The search framework allows us to find high-quality reviews capturing the consensus opinion on given product attributes. Consensus is measured by agreement in opinion polarity and confidence by fraction of opinion in agreement with the majority.

Another approach aimed at finding the consensus opinion is Consento [13]. It is a search engine designed to answer subjective queries by aggregating online comments referring to an entity. A query retrieves opinion on requested entity (e.g. movie) and aspect (if specified) as an aggregate fraction of positive and negative opinions.

Summarization can add additional value to search results on Tweeter [58]. If we perform only keyword search over tweets we might be overwhelmed by the amount of returned information. In this paper they enhance the returned search results with a few different summaries: showing most funny tweets, extracted video links, top keywords and popularity over time.

Balanced Inclusion of Aspects

We can try to summarize sentiments about different aspects without explicitly pointing them out. One such approach [59] was applied to the restaurants domain. One important challenge is dealing with underrepresented aspects while avoiding redundancy. The balanced summary is constructed by touching upon all possible aspects. The most extreme snippets are selected from each category.

Balanced inclusion of all aspects into summary can also be achieved by adding aspect balance term to ILP summarization formulation [60]. The new term maximizes the amount of the least present aspect or alternatively the sum of aspect scores.

Aligned Pairs

Most of the approaches that construct a summary as aspects with representative positive and negative opinions on them usually list supporting evidence without any additional structure. However, it is possible to present positive and negative opinions as contrastive pairs [36]. This way we can actually see why and on which particular detail opinions disagree. For example, one review might claim bad batter life while surfing the web while the other points out good battery life if reading

books.

Comparative LexRank [76] is random walk based approach to scoring pairs of sentences from opposite viewpoints. One option is to generate a single ranking of excerpts, separate them into disjoint sets according to polarity and then remove the redundancy the resulting summary will describe related content from both viewpoints. Alternatively, we can rank pairs of sentences and remove cases where both elements come from the same viewpoint. The resulting summary will then have aligned pairs of sentences showing opposing sentiments.

An interesting take on aligned summaries is shown in [23]. They use topic modeling to find complementary sentence-tweet pairs on recent news. This is achieved by combining two-dimensional topic-aspect model with a cross-collection approach. Presented pairs are selected by co-ranking both the news sentences and tweets.

2.1.4 Summarizing the Differences

A comparative summary (emphasizing differences between multiple groups of documents) can also be built using dominating sets [90] on sentence graph. After finding dominating sets for all groups, we construct comparative summaries by finding complementary dominating sets (by adding sentences from the current group while given dominating sets for other groups).

Comparative summaries can also be constructed by extending summary scoring functions [95] by incorporating a new term pertaining to differences between collections [43]. Coverage based and submodular score of the summary is augmented with an additional term penalizing sentence-wise similarities between the current

summary and all other collections.

Another approach [83] captures similar ideas. They assign weights to terms based on how likely it is to appear in the target collection versus any other collection we are comparing against. Each sentence is then assigned a score representing payload (sum of term scores) per length. Summary is built greedily by adding highest scoring sentences and setting term weights to zero for all already added ones (to avoid redundancy).

In the spirit of LSA and LDA [7], [126] tries to find common topics across collection and topics specific to particular collection using generative mixture models. The generative process selects between background model, common themes (valid across all collections) and specific themes (corresponding to common themes but separate for each collection). User set parameters control how much commonality we want to enforce across collections.

An information theoretic approach to constructing comparative summaries [113] uses entropy as a basis for building a discriminative classifier. This in turn can be used to select discriminative sentences and thus create a summary. However, I think that highly discriminative sentences might not necessarily be useful to the user. For example, a misspelled word has a high discriminative power but is useful in helping user decide which product to buy.

A clustering based approach [116] looks at which (content based) clusters span different documents. Clusters spanning all documents (i.e. being present in all of them) represent common themes. Document specific clusters represent contrastive themes and can be used for summarization of differences. A variation of this approach can also be used for update summarization (by looking at clusters present

only in the new documents after we add them to the old ones).

Comparative summaries can also be constructed using linear programming [32]. The objective function tries to maximize concept coverage over both sets of documents. At the same time, it prefers to include highly contrastive concepts in the summary (by using contrastivness scores for pairs of concepts). The resulting summary describes concepts from two opposite or changed viewpoints.

An approach based on sentiment aspect match (SAM) model and KL divergence is described in [42]. In the case of generic summarization we want to minimize KL divergence between SAM models for the corpus and the summary. Contrastive summarization is achieved by additional terms penalizing similarity between summary and the other set of documents (while still being close to its own set of documents). This is necessary because if we use penalty for similarity between the two summaries we can get into degenerate case we describe one of two (same) prominent aspects (for both products) in each summary instead of smaller but opposite aspect.

Focusing on microblogs [112] tries to summarize differences using weighted feature subset non-negative matrix factorization. This unsupervised approach combines clustering with feature selection. Selected keywords do not represent clusters (as in e.g. LSA) but discriminate between them. Starting with term-microblog matrix, we compute microblog cluster membership and also terms clustering. Terms with the highest weights are the most discriminating ones.

Another matrix factorization approach [111] is used for multi-document summarization. Symmetric matrix factorization is used to obtain clusters and then they select representative sentences by maximizing similarity to the topic as well

as other sentences in the same cluster.

An interesting application of summarizing the differences is multilingual news. If we focus on new event, we expect the news articles in different languages to describe the same content. However, there can be differences in how different people perceive events. For example, an event can be seen as a political move against a country while others perceive it as pro. CoRank [108] is a method for finding differences in this setting. It is a graph based algorithm in which sentences from both languages are ranked simultaneously in a unified framework. Highly ranked sentences from both languages are selected as a summary after greedily removing the redundancy.

Update Summarization

Update summarization strives to provide summaries of new content added to an already existing corpus of documents. The approach of [54] is based on information distance. The best summary is the one with the smallest information distance the whole collection of documents. In the update summarization case, we look for the smallest distance given that we already know previous documents. Information distance can be approximated using compression or using semantic information (e.g. which named entities are the same).

MarginRank [45] is a graph based approach to update summarization. It extends manifold ranking by adding suppression terms which prevent inclusion of content present in both corpora.

Hierarchical clustering can be used to generate update summaries [110]. Two main advantages of this method are incremental construction and hierarchical

structure. Sentences are added into a tree, where internal nodes hold summary sentences for their children and leafs hold all included sentences. Summaries are generated by cutting the tree to a certain depth so that we obtain summary length within the budget. Update summarization is performed by including only sentences from the new documents in the final summary.

2.1.5 Understanding Information Flow

With somewhat similar motivations as in our work [96] the approach [87] also tries to capture the content and development of a corpus in a succinct way. The main components are coherent chains of articles (built bottom up). The summary is then a map of those chains, which can intersect at the shared articles. The objective used to build the map, in addition to coherence of chains, tries to achieve good coverage and increase the connectivity (i.e. the number of intersecting chains).

Relational topic model (RTM) can be used to summarize a network of documents, predict links between them, and predict words within them. This model has predictive power even when we are missing links are only have node attributes, because it jointly models node attributes and links. The generative model for content is similar to LDA. Links are binary and depend on topic assignments. For example, RTM could be used to model topics of local news that incorporates geographic information so that topic retain geographic coherence.

2.2 Crowdsourced Summarization

This section looks at related work pertaining to crowdsourced summarization. It briefly discusses particulars of product reviews, user voting behavior on user generated content, helpfulness votes (including predicting review quality) and ranking (based on content and crowdsourced votes).

A study [18] looked at review creation in the setting of expert critics writing book reviews. They explore possible biases, such as, media outlets being more likely to review a book by the author that also writes for it. They also explore other factors influencing the likelihood of a book being reviewed and its rating, and differences in rating between expert and consumer reviewers (e.g. e.g. expert critics being more favorable to book that won prizes).

Looking at the helpfulness votes [52] we can also discover many interesting properties. Most of the feedback tends to be positive (i.e. users vote reviews as being helpful a lot more than not being helpful). The higher ranked reviews get a lot more votes (in a behavior similar to number of click in search results) and reviews published shortly after the release of a new product tend to accumulate more votes.

Furthermore, [52] notes biases in user voting behavior as causing a deviation between gold standard ratings from independent human raters and helpfulness votes. In particular, the imbalance bias mentioned in [52] is potentially closely related to the issue of voting participation, since it suggests that users might preferentially choose to vote when they have a positive rather than a negative opinion of a review's helpfulness. However, [52] only cites user behavior as a possible cause for divergence between gold standard ratings of quality and aggregated helpfulness

votes, and primarily focuses on the problem of identifying features to build an accurate classifier for separating high and low quality reviews.

Another study [69] looked at review helpfulness using regression as the main tool. They also tried to look at the differences between experience (e.g. music CD) and search goods (e.g. digital camera). They found that reviews with extreme ratings are less helpful in the case of experience goods. Length of the review had a positive effect on the helpfulness and it has larger impact in the case of search goods.

Authors of [16] looked at social mechanisms underlying helpfulness evaluation. They explored whether reviews with a star ranking closer to the average are considered more helpful; if user vote more for reviews agreeing with their own opinion; if negative reviewers are perceived as more competent; and if votes are actually based purely on the quality of the reviews.

More recently, there has been experimental work on herding effects on user ratings of online reviews [117, 68, 70], investigating how a user's awareness of previous votes on a review impacts his own voting decision (e.g. users being more likely to cast a positive vote after seeing previous positive votes compared to the case of not being able to observe the previous votes).

Finally, there is a small literature [3, 26] that looks at theoretical questions regarding ranking and incentivizing high-quality contributions respectively assuming simple quality-based Bernoulli models of voting behavior. Our empirical analysis of user voting patterns can potentially supply richer, more realistic models of rating behavior upon which to base such theoretical studies on algorithms and mechanisms for eliciting and identifying high-quality online content.

2.2.1 Ranking Reviews

A common approach to ranking is using machine learning to learn how to rank [92, 33, 69, 128]. The learned model relies on textual content, polarity of opinions, social features and more to predict the content quality. Training data is often obtained from user feedback (e.g. by allowing users to vote which content is of high quality) because it is cheaper to obtain than manual annotations.

RevRank [106] is another way of ranking reviews striving to place highly helpful reviews in top locations. It is unsupervised and content based. The main idea is to build virtual core review represented by a vector of lexical items that best describe the whole collection of reviews (they use their own weighting scheme reminiscent of tf-idf and select the highest scoring terms). Then we can obtain review scores by computing a distance to the virtual core review.

A coverage based approach [104] selects a set of reviews that best capture the content of the whole collection. They propose a few different variations of the objective function (quality based, split into groups based on polarity of the opinion and requiring to cover all different levels of polarity at the same time). Their approach performs better than simple baselines, but does not significantly outperform simply using the longest review on the quality performance measure.

An interesting paper [25] points out that ranking by helpfulness is not the only option we might want to rank reviews in a way that maximizes sales. They found out that more subjective reviews lead to increased sales which suggests that customers prefer to read reviews describing individual experiences of other buyers. Moreover, reviews containing a mixture of objective and highly subjective sentences also increased the sales. However, for feature based goods (e.g. electronics) users

prefer mainly objective information (confirming facts from product description).

2.2.2 Predicting Review Helpfulness

The majority of this literature has focused on the natural problem of machine-based methods for inferring and predicting content quality, since the large volume of user-generated content rules out the possibility of trusted human editors rating and ranking all contributions to a site. [37, 129, 92, 33, 53, 69, 29, 85] all address the fundamental question of what features of a contribution can help accurately predict its quality, where the gold standard for quality is based either on ratings by independent human raters [52, 85], or the actual received upvotes and downvotes on the website [37, 129, 92, 53, 85]. A number of features are found to be predictors of (appropriate notions of) quality in various settings, ranging from textual content, review length, star rating and product category for Amazon reviews [37, 129, 69], to comment sentiment in Youtube [92], to the topic and payment amount on online Q&A sites such as Google Answers [29]. Later work uses increasingly sophisticated features such as social context [1, 55] in addition to the textual content of a contribution to improve prediction accuracy.

We can rank reviews by their quality expressed as the number of positive helpfulness votes. For new reviews, however, we do not have this information. A possible solution is to build a model that predicts helpfulness based on features [53] such as reviewer expertise, review length and the elapsed time since the introduction of the product. Their model is a weighted sum of individual regression models for each feature.

Doing binary classification of reviews [52] into low and high quality ones can be

done in a supervised manner. Using sentence and product level features, readability and subjectiveness in conjunction with SVM and manually labeled product quality they show the feasibility of low quality review detection.

Authors of [129] build a global regression model for predicting review helpfulness. They note that the review length is a very weak predictor of helpfulness. The most of the predictive power comes from shallow syntactic features (e.g. tendency of quantification, comparative adjectives), while subjectivity and similarity to product description play a minor role.

Another paper [37] also looks into predicting helpfulness in a supervised setting. They note that the most helpful features were the reviews star rating, its length and unigram word content (using bigrams suffers from sparsity due to relatively short reviews). This agrees with finding of other researchers.

Considering the amount of information available in social networks we can try to incorporate it into review ranking [55]. The authors incorporate additional features about authors when they try to predict review quality, such as number of reviews by that author and the quality of his past reviews. Furthermore, they incorporate link structure between authors as a basis for trust links represent belief in authors producing high quality content.

Part II

Supervised Approaches to Condensing Information

A traditional approach to condensing information is document summarization, where we take a large collection of text documents and condense the gist of them into a short summary. Such summary then allows for easier and less time intensive consumption by the users. Creating such summaries become possible and necessary with the advances in technology and the amount of available digital data. Due to automatic way of constructing summaries, we can take this one step further. Instead of manually tunning the parameters of the approach we can improve it by combining it with learning. By using supervised learning we can leverage data to automatically learn how to produce good summaries based on examples.

The following chapter discusses a supervised learning approach to extractive multi-document summarization. Learning is implemented with a large-margin method that directly optimizes a convex relaxation of the desired performance measure. The proposed learning method applies to all summarization methods with submodular scoring functions. Furthermore, it is possible to use a structured prediction approach that captures dependencies between elements of the constructed summary and thus allows to control for redundancy. Experiments demonstrate this approach's effectiveness for different scoring functions on multiple datasets and tasks. Compared to functions that were tuned manually, method described in the following sections significantly improves performance and enables high-fidelity models with number of parameters well beyond what could reasonably be tuned by hand.

CHAPTER 3

AUTOMATIC DOCUMENT SUMMARIZATION

Automatic document summarization is the problem of constructing a short text describing the main points in a (set of) document(s). Example applications range from generating short summaries of news articles, to presenting snippets for URLs in web-search. This part of the thesis focuses on extractive multi-document summarization, where the final summary is a subset of the sentences from multiple input documents. In this way, extractive summarization avoids the hard problem of generating well-formed natural-language sentences, since only existing sentences from the input documents are presented as part of the summary.

A current state-of-the-art method for document summarization was recently proposed by [50], using a submodular scoring function based on inter-sentence similarity. On the one hand, this scoring function rewards summaries that are similar to many sentences in the original documents (i.e. promotes coverage). On the other hand, it penalizes summaries that contain sentences that are similar to each other (i.e. discourages redundancy). While obtaining the exact summary that optimizes the objective is computationally hard, they show that a greedy algorithm is guaranteed to compute a good approximation. However, their work does not address how to select a good inter-sentence similarity measure, leaving this problem as well as selecting an appropriate trade-off between coverage and redundancy to manual tuning.

To overcome this problem, we can construct a supervised learning method that can learn both the similarity measure as well as the coverage/reduncancy trade-off from training data. Furthermore, the learning algorithm used is not limited to the model of [50], but applies to all monotone submodular summarization

models. Due to the diminishing-returns property of monotone submodular set functions and their computational tractability, this class of functions provides a rich space for designing summarization methods. To illustrate the generality of this approach, empirical evaluation also includes experiments for a coverage-based model originally developed for diversified information retrieval [99].

In general, the described method learns a parameterized monotone submodular scoring function from supervised training data. Given a set of documents and their summaries as training examples, the learning problem is formulated as a structured prediction problem using a maximum-margin algorithm in the structural support vector machine (SVM) framework. Note that, unlike other learning approaches, this method does not require a heuristic decomposition of the learning task into binary classification problems [40], but directly optimizes a structured prediction. This enables the algorithm to directly optimize the desired performance measure (e.g. ROUGE) during training. Furthermore, the method is not limited to linear-chain dependencies like [14, 91], but can learn any monotone submodular scoring function.

This ability to easily train summarization models makes it possible to efficiently tune models to various types of document collections. In particular, experiments demonstrate that the learning method can reliably tune models with hundreds of parameters based on a training set of about 30 examples. This increases the fidelity of models compared to their hand-tuned counterparts, showing significantly improved empirical performance. Experimental section also provides a detailed investigation into the sources of these improvements, identifying further directions for research.

3.1 Submodular Document Summarization

This section illustrates how document summarization can be addressed using submodular set functions. The set of documents to be summarized is split into a set of individual sentences $x = \{s_1, ..., s_n\}$. The summarization method then selects a subset $\hat{y} \subseteq x$ of sentences that maximizes a given scoring function $F_x : 2^x \to \mathbb{R}$ subject to a budget constraint (e.g. less than B characters).

$$\hat{y} = \arg\max_{y \subseteq x} F_x(y) \qquad s.t. \; |y| \leq B \tag{3.1}$$

In the following we restrict the admissible scoring functions F to be submodular.

Definition 1 *Given a set x, a function $F : 2^x \to \mathbb{R}$ is submodular iff for all $u \in U$ and all sets s and t such that $s \subseteq t \subseteq x$, we have,*

$$F(s \cup \{u\}) - F(s) \geq F(t \cup \{u\}) - F(t).$$

Intuitively, this definition says that adding u to a subset s of t increases f at least as much as adding it to t. Using two specific submodular functions as examples, the following sections illustrate how this diminishing returns property naturally reflects the trade-off between maximizing coverage while minimizing redundancy.

3.1.1 Pairwise Scoring Function

The first submodular scoring function considered here was proposed by [50] and is based on a model of pairwise sentence similarities. It scores a summary y using the following function, which [50] show is submodular:

$$F_x(y) = \sum_{i \in x \setminus y, j \in y} \sigma(i, j) - \lambda \sum_{i,j \in y : i \neq j} \sigma(i, j). \tag{3.2}$$

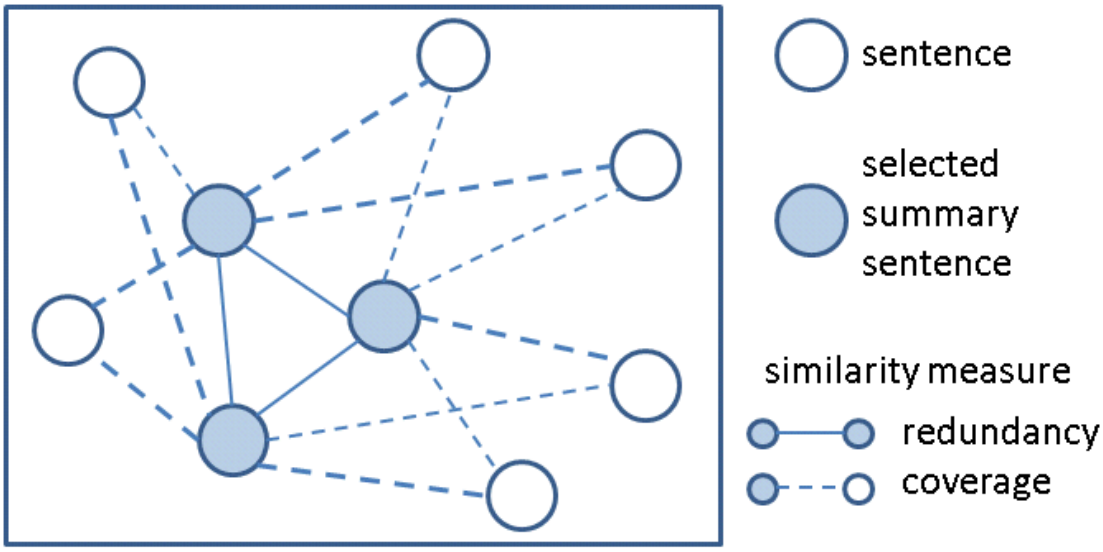


Figure 3.1: Illustration of the pairwise model. Not all edges are shown for clarity purposes. Edge thickness denotes the similarity score.

In the above equation, $\sigma(i, j) \geq 0$ denotes a measure of similarity between pairs of sentences i and j. The first term in Eq. 3.2 is a measure of how similar the sentences included in summary y are to the other sentences in x. The second term penalizes y by how similar its sentences are to each other. $\lambda > 0$ is a scalar parameter that trades off between the two terms. Maximizing $F_x(y)$ amounts to increasing the similarity of the summary to excluded sentences while minimizing repetitions in the summary. An example is illustrated in Figure 3.1. In the simplest case, $\sigma(i, j)$ may be the TFIDF [84] cosine similarity, but a later description will show how to learn sophisticated similarity functions.

3.1.2 Coverage Scoring Function

A second scoring function to consider was first proposed for diversified document retrieval [99, 123], but it naturally applies to document summarization as well [44].

It is based on a notion of word coverage, where each word v has some importance weight $\omega(v) \geq 0$. A summary y covers a word if at least one of its sentences contains the word. The score of a summary is then simply the sum of the word weights its covers (though we could also include a concave discount function that rewards covering a word multiple times [81]):

$$F_x(y) = \sum_{v \in V(y)} \omega(v). \tag{3.3}$$

In the above equation, $V(y)$ denotes the union of all words in y. This function is analogous to a maximum coverage problem, which is known to be submodular [35].

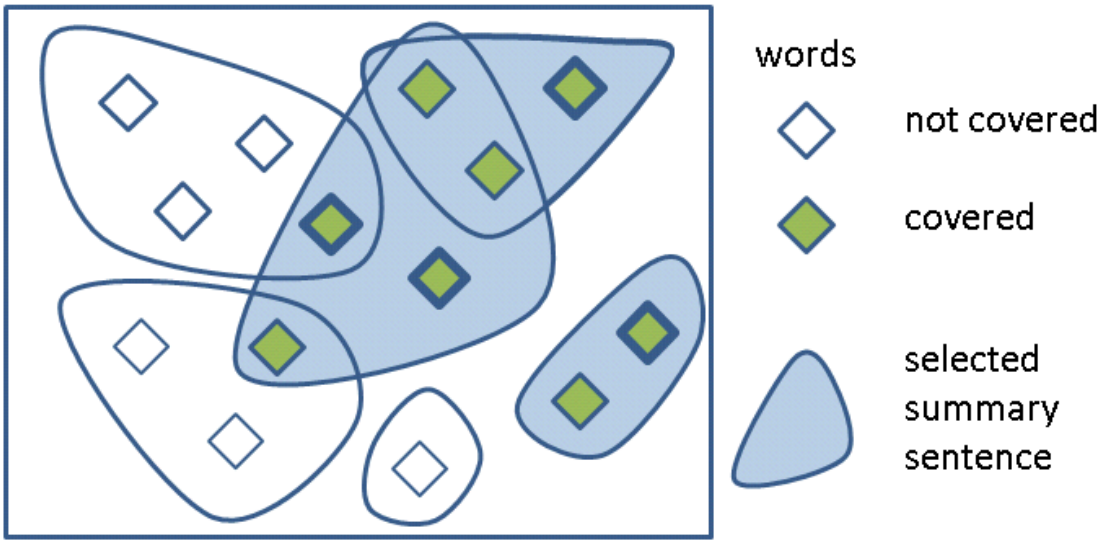


Figure 3.2: Illustration of the coverage model. Word border thickness represents importance.

An example of how a summary is scored is illustrated in the Figure 3.2. Analogous to the definition of similarity $\sigma(i, j)$ in the pairwise model, the choice of the word importance function $\omega(v)$ is crucial in the coverage model. A simple heuristic is to weigh words highly that occur in many sentences of x, but in few other documents [99]. However, how to learn $\omega(v)$ from training data will be shown in

the following section.

Algorithm 1 Greedy algorithm for finding the best summary $\hat{y}$ given a scoring function $F_x(y)$.

Parameter: $r > 0$.
$\hat{y} \leftarrow \emptyset$
$A \leftarrow x$
while $A \neq \emptyset$ **do**
  $k \leftarrow \arg\max_{l \in A} \frac{F_x(\hat{y} \cup \{l\}) - F_x(\hat{y})}{(c_l)^r}$
  if $c_k + \sum_{i \in \hat{y}} c_i \leq B$ **and** $F_x(\hat{y} \cup \{k\}) - F_x(\hat{y}) \geq 0$ **then**
    $\hat{y} \leftarrow \hat{y} \cup \{k\}$
  end if
  $A \leftarrow A \backslash \{k\}$
end while

3.1.3 Computing a Summary

Computing the summary that maximizes either of the two scoring functions from above (i.e. Eqns. (3.2) and (3.3)) is NP-hard [63]. However, it is known that the greedy algorithm 1 can achieve a $1 - 1/e$ approximation to the optimum solution for any linear budget constraint [50, 35]. Even further, this algorithm provides a $1 - 1/e$ approximation for any monotone submodular scoring function.

The algorithm starts with an empty summarization. In each step, a sentence is added to the summary that results in the maximum relative increase of the objective. The increase is relative to the amount of budget that is used by the added sentence. The algorithm terminates when the budget B is reached.

Note that the algorithm has a parameter r in the denominator of the selection rule, which [50] report to have some impact on performance. In the algorithm, c_i represents the cost of the sentence (i.e., length). Thus, the algorithm actually

selects sentences with large marginal unity relative to their length (trade-off controlled by the parameter r). Selecting r to be less than 1 gives more importance to "information density" (i.e. sentences that have a higher ratio of score increase per length). The $1 - \frac{1}{e}$ greedy approximation guarantee holds despite this additional parameter [50]. More details on the choice of r and its effects are provided in the experiments section.

3.2 Learning Algorithm for Structured Prediction

This section describes a supervised learning method for training a submodular scoring function to produce desirable summaries. In particular, for the pairwise and the coverage model, this encompases learning the similarity function $\sigma(i, j)$ and the word importance weights $\omega(v)$ respectively. In particular, $\sigma(i, j)$ and $\omega(v)$ are parameterized using a linear model, allowing that each depends on the full set of input sentences x:

$$\sigma_x(i, j) = \mathbf{w}^T \phi_x^p(i, j) \quad \omega_x(v) = \mathbf{w}^T \phi_x^c(v). \tag{3.4}$$

In the above equations, $\mathbf{w}$ is a weight vector that is learned, and $\phi_x^p(i, j)$ and $\phi_x^c(v)$ are feature vectors. In the pairwise model, $\phi_x^p(i, j)$ may include feature like the TFIDF cosine between i and j or the number of words from the document titles that i and j share. In the coverage model, $\phi_x^c(v)$ may include features like a binary indicator of whether v occurs in more than 10% of the sentences in x or whether v occurs in the document title.

The weights are learned following a large-margin framework using structural

SVMs [105]. Structural SVMs learn a discriminant function

$$h(x) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^\top \Psi(x, y) \tag{3.5}$$

that predicts a structured output y given a (possibly also structured) input x. $\Psi(x, y) \in \mathbb{R}^N$ is called the joint feature-map between input x and output y. Note that both submodular scoring function in Eqns. (3.2) and (3.3) can be brought into the form $\mathbf{w}^T \Psi(x, y)$ for the linear parametrization in Eq. (3.6) and (3.7):

$$\Psi^p(x, y) = \sum_{i \in x \setminus y, j \in y} \phi_x^p(i, j) - \lambda \sum_{i,j \in y : i \neq j} \phi_x^p(i, j), \tag{3.6}$$

$$\Psi^c(x, y) = \sum_{v \in V(y)} \phi_x^c(v). \tag{3.7}$$

After this transformation, it is easy to see that computing the maximizing summary in Eq. (3.1) and the structural SVM prediction rule in Eq. (3.5) are equivalent.

To learn the weight vector $\mathbf{w}$, structural SVMs require training examples (x^1, y^1), ..., (x^n, y^n) of input/output pairs. In document summarization, however, the "correct" extractive summary is typically not known. Instead, training documents x^i are typically annotated with multiple manual (non-extractive) summaries (denoted by Y^i). To determine a single extractive target summary y^i for training, we find the extractive summary that (approximately) optimizes ROUGE score – or some other loss function $\Delta(Y^i, y)$ – with respect to Y^i.

$$y^i = \operatorname*{argmin}_{y \in \mathcal{Y}} \Delta(Y^i, y) \tag{3.8}$$

The y^i determined in this way is called the "target" summary for x^i. Note that y^i is a greedily constructed approximate target summary based on its proximity to Y^i via Δ. Because of this, the learned model can only predict approximately good summaries y^i from x_i. However, the experiments in Section 3.3.3 suggest that most of the score difference between manual summaries and y^i is due to it being an extractive summary and not due to greedy construction.

Algorithm 2 Cutting-plane algorithm for solving the learning optimization problem.

Parameter: desired tolerance $\epsilon > 0$.
$\forall i : \mathcal{W}_i \leftarrow \emptyset$
repeat
  for $\forall i$ **do**
    $\hat{y} \leftarrow \arg\max_{y} w^T \Psi(x^i, y) + \Delta(Y^i, y)$
    if $w^T \Psi(x^i, y^i) + \epsilon \leq w^T \Psi(x^i, \hat{y}) + \Delta(Y^i, \hat{y}) - \xi_i$ **then**
      $\mathcal{W}_i \leftarrow \mathcal{W}_i \cup \{\hat{y}\}$
      $w \leftarrow$ solve QP (3.9) using constraints $\mathcal{W}_i$
    end if
  end for
until no $\mathcal{W}_i$ has changed during iteration

Following the structural SVM approach, we can now formulate the problem of learning $\mathbf{w}$ as the following quadratic program (QP):

$$\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \tag{3.9}$$
$$\text{s.t. } \mathbf{w}^\top \Psi(x^i, y^i) - \mathbf{w}^\top \Psi(x^i, \hat{y}^i) \geq$$
$$\Delta(\hat{y}^i, Y^i) - \xi_i, \quad \forall \hat{y}^i \neq y^i, \quad \forall 1 \leq i \leq n.$$

The above formulation ensures that the scoring function with the target summary (i.e. $\mathbf{w}^\top \Psi(x^i, y^i)$) is larger than the scoring function for any other summary $\hat{y}^i$ (i.e., $\mathbf{w}^\top \Psi(x^i, \hat{y}^i)$). The objective function learns a large-margin weight vector $\mathbf{w}$ while trading it off with an upper bound on the empirical loss. The two quantities are traded off with a parameter $C > 0$.

Even though the QP has exponentially many constraints in the number of sentences in the input documents, it can be solved approximately in polynomial time via a cutting plane algorithm [105]. The steps of the cutting-plane algorithm are shown in Algorithm 2. In each iteration of the algorithm, for each training document x^i, a summary $\hat{y}^i$ which most violates the constraint in (3.9) is found.

This is done by finding

$$\hat{y} \leftarrow \arg\max_{y \in \mathcal{Y}} w^T \Psi(x^i, y) + \Delta(Y^i, y),$$

for which we use a variant of the greedy algorithm in Figure 1. After a violating constraint for each training example is added, the resulting quadratic program is solved. These steps are repeated until all the constraints are satisfied to a required precision ϵ.

Finally, special care has to be taken to appropriately define the loss function Δ given the disparity of Y^i and y^i. Therefore, we first define an intermediate loss function

$$\Delta_R(Y, \hat{y}) = \max(0, 1 - ROUGE1_F(Y, \hat{y})),$$

based on the ROUGE-1 F score. To ensure that the loss function is zero for the target label as defined in (3.8), we normalized the above loss as below:

$$\Delta(Y^i, \hat{y}) = \max(0, \Delta_R(Y^i, \hat{y}) - \Delta_R(Y^i, y^i)),$$

The loss Δ was used in the experiments. Thus training a structural SVM with this loss aims to maximize the ROUGE-1 F score with the manual summaries provided in the training examples, while trading it off with margin. Note that we could also use a different loss function (as the method is not tied to this particular choice), if we had a different target evaluation metric. Finally, once a $\mathbf{w}$ is obtained from structural SVM training, a predicted summary for a test document x can be obtained from (3.5).

3.3 Evaluation

In this section, the empirical evaluation of the approach described in the preceeding section is presented. Following [50], experiments were conducted on two different datasets (DUC '03 and '04). These datasets contain document sets with four manual summaries for each set. For each document set, all the articles were concatenated and then split into sentences using the tool provided with the '03 dataset. The supervised setting used 10 resamplings with a random 20/5/5 ('03) and 40/5/5 ('04) train/test/validation split. The best C value in (3.9) was determined using the performance on each validation set and then the average performence was reported over the corresponding test sets. Baseline performance (the approach of [50]) was computed using all 10 test sets as a single test set. All experiments and datasets used $r = 0.3$ in the greedy algorithm as recommended in [50] for the '03 dataset. Changing r has only a small influence on performance, though setting r to 1 and thus eliminating the non-linearity does lower the score (e.g. to 0.38466 for the pairwise model on DUC '03 compared with the results on Figure 3.3).

The construction of features for learning is organized by word groups. The most trivial group is simply all words (*basic*). Considering the properties of the words themselves, several features were constructed from properties such as capitalized words, non-stop words and words of certain length (*cap+stop+len*). Another set of features was obtained from the most frequently occuring words in all the articles (*minmax*). The position of a sentence (containing the word) in the article was also considered as another feature (*location*). All those word groups can then be further refined by selecting different thresholds, weighting schemes (e.g. TFIDF) and forming binned variants of these features.

The pairwise model is defined on top of cosine similarity between sentences

using only words in a given word group during computation. In the word coverage model separate features are created for covering words in different groups. This gives us fairly comparable feature strength in both models. The only further addition is the use of different word coverage levels in the coverage model. First, how well does a sentence cover a word (e.g. a sentence with five instances of the same word might cover it better than another with only a single instance)? And secondly, how important it is to cover a word (e.g. if a word appears in a large fraction of sentences we might want to be sure to cover it)? Combining those two criteria using different thresholds we get a set of features for each word. The coverage features are motivated from the approach of [123]. In contrast, the hand-tuned pairwise baseline uses only TFIDF weighted cosine similarity between sentences using all words, following the approach in [50].

The resulting summaries are evaluated using ROUGE version 1.5.5 [47]. The evaluation is based on the ROUGE-1 F measure because it was used by [50] and because it is one of the commonly used performance scores in recent work. However, the same learning method applies to other performance measures as well. Note that the approach uses the ROUGE-1 F measure both for the loss function during learning, as well as for the evaluation of the predicted summaries.

3.3.1 How does Learning Compare to Manual Tuning?

In the first experiment, the supervised learning approach is compared to the hand-tuned approach. The results from this experiment are summarized in Figure 3.3. First, supervised training of the pairwise model [50] resulted in a statistically significant ($p \leq 0.05$ using paired t-test) increase in performance on both datasets compared to the reimplementation of the manually tuned pairwise model. Note

that the reimplementation of the approach of [50] resulted in slightly different performance numbers than those reported in [50] – better on DUC '03 and somewhat lower on DUC '04, if evaluated on the same selection of test examples as theirs. We conjecture that this is due to small differences in implementation and/or preprocessing of the dataset. Furthermore, as authors of [50] note in their paper, the '03 and '04 datasets behave quite differently.

model	dataset	ROUGE-1 F (stderr)
pairwise	DUC '03	**0.3929** (0.0074)
coverage		**0.3784** (0.0059)
hand-tuned		0.3571 (0.0063)
pairwise	DUC '04	**0.4066** (0.0061)
coverage		0.3992 (0.0054)
hand-tuned		0.3935 (0.0052)

Figure 3.3: Results obtained on DUC '03 and '04 datasets using the supervised models. Increase in performance over the hand-tuned is statistically significant ($p \leq 0.05$) for the pairwise model on the both datasets, but only on DUC '03 for the coverage model.

Figure 3.3 also reports the performance for the coverage model as trained by the described algorithm. These results can be compared against those for the pairwise model. Since the features are of a comparable strength in both approaches, and they both use the same greedy algorithm and structural SVM learning method, this comparison largely reflects the quality of models themselves. On the '04 dataset both models achieve the same performance while on '03 the pairwise model performs significantly ($p \leq 0.05$) better than the coverage model.

Overall, the pairwise model appears to perform slightly better than the coverage model with the datasets and features we used. Therefore, the following exploration focuses on the pairwise model.

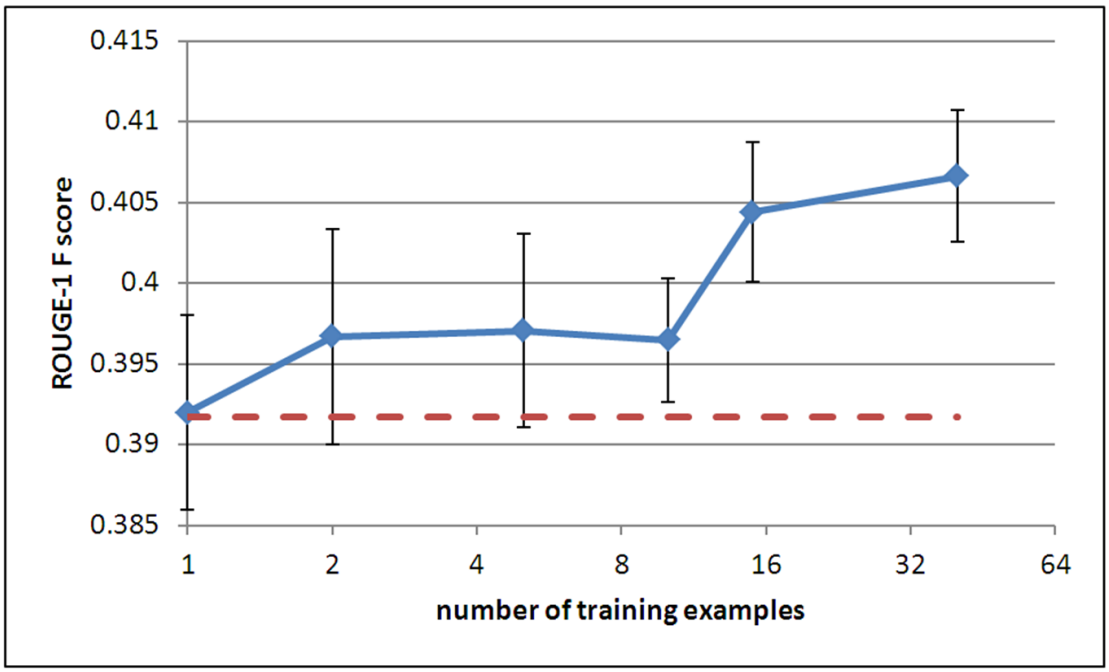


Figure 3.4: Learning curve for the pairwise model on DUC '04 dataset showing ROUGE-1 F scores for different numbers of learning examples (logarithmic scale). The dashed line represents the preformance of the hand-tuned model.

3.3.2 How Fast does the Algorithm Learn?

Hand-tuned approaches have limited flexibility. Whenever we move to a significantly different collection of documents we have to reinvest time to retune it. Learning can make this adaptation to a new collection more automatic and faster – especially since training data has to be collected even for manual tuning.

Figure 3.4 evaluates how effectively the learning algorithm can make use of a given amount of training data. In particular, the figure shows the learning curve for the approach. Even with very few training examples, the learning approach already outperforms the baseline. Furthermore, at the maximum number of training examples available to us the curve still increases. Therefore we can conjecture that more data would further improve performance.

3.3.3 Where is Room for Improvement?

To get a rough estimate of what is actually achievable in terms of the final ROUGE-1 F score, we can look at different "upper bounds" under various scenarios (Figure 3.5). First, ROUGE score is computed by using four manual summaries from different assessors, so that we can estimate inter-subject disagreement. If one computes the ROUGE score of a held-out summary against the remaining three summaries, the resulting performance is given in the row labeled *human* of Figure 3.5. It provides a reasonable estimate of human performance.

Second, in extractive summarization we can restrict summaries to sentences from the documents themselves, which is likely to lead to a reduction in ROUGE. To estimate this drop, the greedy algorithm is used to select the extractive summary that maximizes ROUGE on the test documents. The resulting performance is given in the row *extractive* of Figure 3.5. On both datasets, the drop in performance for this (approximately) optimal extractive summary is about 10 points of ROUGE. Comparing the greedy algorithm with exhaustive search can give us some sense of how close to the optimal is the computed approximate solution. In about half the cases for selecting up to three selected sentences (more than that would take too long) we get the same solution, in other cases the soultion was on average about 1% below optimal confirming that greedy selection works quite well.

Third, we expect some drop in performance, since the model may not be able to fit the optimal extractive summaries due to a lack of expressiveness. This can be estimated by looking at training set performance, as reported in row *model fit* of Figure 3.5. On both datasets, we see a drop of about 5 points of ROUGE performance. Adding more and better features might help the model fit the data better.

Finally, the last drop in performance may come from overfitting. The test set ROUGE scores are given in the row *prediction* of Figure 3.5. Note that the drop between training and test performance is rather small, so overfitting is not an issue and is well controlled in our algorithm. Therefore we can conclude that increasing model fidelity seems like a promising direction for further improvements.

bound	dataset	ROUGE-1 F
human	DUC '03	0.56235
extractive		0.45497
model fit		0.40873
prediction		0.39294
human	DUC '04	0.55221
extractive		0.45199
model fit		0.40963
prediction		0.40662

Figure 3.5: Upper bounds on ROUGE-1 F scores: agreement between manual summaries, greedily computed best extractive summaries, best model fit on the train set (using the best C value) and the test scores of the pairwise model.

3.3.4 Which Features are Most Useful?

To understand which features affected the final performance of our approach, we can assess the strength of each set of the features. In particular, how does the final test score change when we remove certain features groups (described in the beginning of Section 3.3) is shown on Figure 3.6.

The most important group of features are the *basic* features (pure cosine similarity between sentences) since removing them results in the largest drop in performance. However, other features play a significant role too (i.e. only the basic ones are not enough to achieve good performance). This confirms that performance can be improved by adding richer fatures instead of using only a single similarity score

as in [50]. Using learning for these complex model is essential, since hand-tuning is likely to be intractable.

The second most important group of features considering the drop in performance (i.e. *location*) looks at positions of sentences in the articles. This makes intuitive sense because the first sentences in news articles are usually packed with information. The other three groups do not have a significant impact on their own.

removed group	ROUGE-1 F
none	0.40662
basic	**0.38681**
all except basic	**0.39723**
location	**0.39782**
sent+doc	0.39901
cap+stop+len	0.40273
minmax	0.40721

Figure 3.6: Effects of removing different feature groups on the DUC '04 dataset. Bold font marks significant difference ($p \leq 0.05$) when compared to the full pariwise model. The most important are basic similarity features including all words (similar to [50]). The last feature group actually lowered the score but is included in the model because we only found this out later on DUC '04 dataset.

3.3.5 How Important is it to Train with Multiple Summaries?

While having four manual summaries may be important for computing a reliable ROUGE score for evaluation, it is not clear whether such an approach is the most efficient use of annotator resources for training. In the final experiment for this setting, the method was trained using only a single manual summary for each set of documents. When using only a single manual summary, the first one out of

the provided four reference summaries was arbitrarily selected and used only to compute the target label for training (instead of using average loss towards all four of them). Otherwise, the experimental setup was the same as in the previous subsections, using the pairwise model.

For DUC '04, the ROUGE-1 F score obtained using only a single summary per document set was 0.4010, which is slightly but not significantly lower than the 0.4066 obtained with four summaries (as shown on Figure 3.3). Similarly, on DUC '03 the performance drop from 0.3929 to 0.3838 was not significant as well.

Based on those results, we can conjecture that having more documents sets with only a single manual summary is more useful for training than fewer training examples with better labels (i.e. multiple summaries). In both cases, we spend approximately the same amount of effort (as the summaries are the most expensive component of the training data), however having more training examples helps (according to the learning curve presented before) while spending effort on multiple summaries appears to have only minor benefit for training.

3.4 Summary

This chapter presented a supervised learning approach to extractive document summarization based on structual SVMs. The learning method applies to all submodular scoring functions, ranging from pairwise-similarity models to coverage-based approaches. The learning problem is formulated into a convex quadratic program and was then solved approximately using a cutting-plane method acchieving a constant factor approximation. In an empirical evaluation, the structural SVM approach significantly outperforms conventional hand-tuned models on

the DUC '03 and '04 datasets. A key advantage of the learning approach is its ability to handle large numbers of features, providing substantial flexibility for building high-fidelity summarization models. Furthermore, it shows good control of overfitting, making it possible to train models even with only a few training examples.

CHAPTER 4

CREATING PRODUCT COMPARISONS

Drawing inspiration from the approach described in the previous sections, we can apply the same ideas to a novel task. This will demonstrate the flexibility of the approach and provide another use case where learning helps to streamline the information access.

To facilitate direct comparisons between different products, an approach to constructing short and comparative summaries based on product reviews is described in the following sections [94]. In particular, the user can view automatically aligned pairs of snippets describing reviewers' opinions on different features (also selected automatically by the approach) for two selected products. Again, it uses a submodular objective function that avoids redundancy, that is efficient to optimize, and that aligns the snippets into pairs. Snippets are chosen from product reviews and thus easy to obtain. The experiments show that the method constructs qualitatively good summaries, and that it can be tuned via supervised learning.

4.1 Motivation

After deciding what kind of product to buy (e.g. a cell phone), it is often still hard to make a good choice due to abundance of different brands and models. Using the internet as a convenient source of information, we can usually find a wealth of data describing many products – even more so if we plan to make our purchase online too. First, reading in-depth professional reviews might help us to familiarize ourselves with strengths and weaknesses of a certain product, but does not directly give us direct understanding of how it compares to other choices. Second, we

can find tables comparing products, but they usually list only product feature specifications as documented by the manufacturer and do not provide insight into how those features translate into usefulness during the actual use. Third, we can find user reviews of products which describe experiences of using the product and are often helpful in making the purchasing decision. However, the major drawbacks are the need to read many reviews to get an estimate of agreement between users and the need to manually compare different products (by keeping the contents of the reviews in mind).

The aim is to fill-in this space and provide a way of explicitly comparing products of the same type (e.g., competing cell phones). These comparisons should reflect users' experiences (i.e., are based on product reviews), so that they can complement specification-based comparisons tables.

Many online retailer websites already provide user reviews. In this approach, an additional view of these reviews is providing a compact (to avoid the need to read many long reviews) and comparative (to facilitate decision support) summary of them. In other words, the approach selects important snippets from the reviews of one product and present them aligned with snippets talking about the same aspect from the reviews of the other product. This allows users to quickly read through the aligned pairs describing reviewers' opinions on important aspects of the two products they are trying to compare.

Figure 4.1 shows an example of what we would expect from the system. Each pair talks about one aspect of the product and the selection should represent the most important ones (as defined by the reviewers' selection of which ones to mention). The snippets give insight into how well the product specifications translate into real-world utility based on reviewers' experiences and how do they

Product	Comparative Snippet Pairs
A	battery lasted for about 7h of web browsing
B	I got about 8h but only if I disabled wireless
A	screen has a good uniform lighting
B	there was a slight light bleed on the screen
A	buttons were hard to press
B	despite small buttons, they were easy to use

Figure 4.1: Illustrative example of a comparative summary for products A and B.

compare to a competing product. Reading such summary provides an alternative to time-consuming reading of many reviews and at the same time provides direct comparison between different purchasing choices.

4.2 Model

Let's assume that the user selects two products A & B to compare against each other, and that each product $p \in \{A, B\}$ comes with a respective set of reviews $R^p = r_1^p, ..., r_{n^p}^p$ (obtained from e.g. online retailer's website). Each set of reviews R^p is then split into snippets $S^p = s_1^p, ..., s_{m^p}^p$ (by e.g. using sentences as snippets). Finally, objective function F is used to select a set of snippet pairs (s_i^A, s_j^B) (where both snippets describe the same aspect but for two different products) that represent the final summary.

Each snippet is represented as a vector (in the simplest case using a bag-of-words TFIDF scores; otherwise snippets can be represented using any features) and $v_k^p(x)$ represents the value of feature x in snippet s_k^p (e.g. TFIDF score of word x). To construct a pair, we want to select two snippets (one for each product) such that they talk about the same aspect (i.e. are aligned). To promote good alignment we use sum of features in the intersection of the two snippets (Eq. 4.1). Furthermore,

we want to also account for the information outside the intersection to promote showing the differences (instead of just finding the most similar snippets). To achieve this we use the sum of features outside of the intersection (Eq. 4.2), but clamped to be less or equal to the value of the intersection. Clamping the score avoids assigning high scores to some bad corner-cases (e.g. two long snippets being aligned only on the word "the"). Finally, for any candidate ($s_i^A \in S^A, s_j^B \in S^B$) we then use Eq. 4.3 (with the favorable properties of having a good alignment and including useful comparative information) to score the given pairing of snippets.

$$c(s_i^A, s_j^B) = \sum_{x \in s_i^A \wedge x \in s_j^B} v_i^A(x) \tag{4.1}$$

$$g(s_i^A, s_j^B) = \min\{ \sum_{x \in s_i^A \wedge x \notin s_j^B} v_i^A(x), c(s_i^A, s_j^B)\} \tag{4.2}$$

$$\begin{aligned} f(s_i^A, s_j^B) = g(s_i^A, s_j^B) + c(s_i^A, s_j^B) + \\ c(s_j^B, s_i^A) + g(s_j^B, s_i^A) \end{aligned} \tag{4.3}$$

Let's consider some examples of bad snippet pairs to better understand why we selected this scoring function. Snippet pair *(the battery life is good, the screen had some light bleed)* is a bad choice because the snippets overlap only on the word "the". It does not get chosen because the score is limited by the intersection. Pair *(all buttons are big and very easy on the fingers, has buttons)* is highly unbalanced (left snippet adds a lot of weight if we naively summed the word weights) but the score is again limited by the intersection. In the third example *(and the screen has fantastic colors, while the screen has fantastic colors)* we have a large intersection, but still a low score because we do not have much product specific information (and

thus parts outside the intersection do not add much weight). The approach still includes coverage terms (and thus gives weight to frequently occurring phrases), however it prefers to select and point out the differences (to facilitate deciding between the products).

In contrast to scoring a single pair of snippets (Eq. 4.3), the main objective function (Eq. 4.7) is a submodular set function. In addition to selecting aligned pairs talking about important aspects, we can now select diverse pairs that give a good coverage of the source information (original reviews) while avoiding redundancy. We parametrize this objective with weight vector $\mathbf{w}$ (where scalar w_x corresponds to the feature x), which allows for supervised learning, but can be substituted with $\mathbf{1}$ in the simple case of *uniform weights*.

The objective function F (Eq. 4.7) follows a similar pattern to the individual pair scoring. To simplify the notation we use $\alpha(S)$ to represent the union of all words present in snippets for product A included in S but not for B, similarly $\beta(S)$ for words in B but not A and $\gamma(S)$ for the union of all words in the intersections. Function H sums the largest weights for all words (thus resulting in diminishing returns for covering a word multiple times) in the target set (i.e. intersection $X = \gamma(S)$ or the remainder corresponding to A or B). The final objective (Eq. 4.7) is composed of four parts (corresponding to intersections and remainders using weights for A or B).

$$H^A(S, X) = \sum_{x \in X} \max_{(i,j) \in S} w_x v_i^A(x) \tag{4.4}$$

$$H^B(S, X) = \sum_{x \in X} \max_{(i,j) \in S} w_x v_j^B(x) \tag{4.5}$$

$$G^p(S, X) = \min\{H^p(S, X), H^p(S, \gamma(S))\} \tag{4.6}$$

$$\begin{aligned} F(S) = G^A(S, \alpha(S)) + H^A(S, \gamma(S)) + \\ H^B(S, \gamma(S)) + G^B(S, \beta(S)) \end{aligned} \tag{4.7}$$

Eq. 4.7 can be efficiently maximized using a greedy algorithm (linear in the number of candidate pairs and selected set cardinality) which achieves a constant factor approximation [73] and works well in practice.

4.3 Evaluation

Data. Evaluation of this approach was performed using reviews from Amazon's web site. The reviews were scraped for 8 tablets (from different manufacturers) and then split into sentences based on punctuation. Next, sentences are parsed and, if necessary, further split them into smaller snippets (e.g. in the case of two clauses connected by "and"). By doing this we obtain in the order of 10000 snippets per product. Experiments were performed using TFIDF weighting, where we treat each review as one document.

Filtering. Because we are selecting pairs of snippets, the number of candidate pairs (and thus running time of the greedy algorithm) grows quadratically with the number of snippets obtained from the reviews. To speed up the selection of which pairs to present to the user, the set of the top most similar snippets (using cosine similarity) was precomputed. Each product pair (for which we want to show a comparison) is limited to the top 10000 most similar pairs while running the

greedy algorithm (for performance reasons). This is a reasonable number based on my empirical observations and the fact that even within this limited set the similarities in the bottom part already became very weak.

Labels. For the purposes of the evaluation and supervised learning we require labeled data. The labels are per snippet pair and are defined as follows:

+1.0 a good pair (the two snippets are talking about the same aspect and contain relevant/useful information about the product)

+0.5 a misaligned pair (at least one snippet contains useful information but it does not talk about the same thing as the other one, e.g. *"battery life was good"* and *"the battery is hard to replace"*)

-0.5 irrelevant comments (e.g. reviewer discussing seller's customer service response)

-1.0 a bad pair (the pair contains no useful information, is nonsensical, etc.)

Qualitative Evaluation. Figure 4.2 shows the top 5 pairs selected by the method comparing the Apple iPad with the Google Nexus tablet. All selected pairs are "good" according to the labeling and they also fit our goals: they are aligned (both snippets talk about the same aspect, e.g. microsd slot), they are balanced (no very short snippets paired with extremely long ones), and they talk about a non-redundant set of topics (e.g. storage space, expansion slots, email accounts etc.).

Uniform weights. This approach can be used in a non-learning setting (where we do not require labels except for the evaluation purposes). In this case we use *uniform weights* by setting $\mathbf{w} = \mathbf{1}$. The results of this scoring function are

snippets for Apple iPad	snippets for Google Nexus
an ipod built in so you can listen to your favorite tunes via the music app or download new music via the itunes app	have to download specific apps to be able to download anything since safari doesnt handle downloads and you cant add music to the ipod app without first synching with itunes unless youre purchasing from itunes on the device itself
to sync my gmail contacts i had to set up my gmail account as a microsoft exchange account this is stupid why cant i just set it up as a gmail account and automatically sync my contacts without the extra hassle	setup was easy and it synced flawlessly with my gmail account automatically downloading apps
no microsd card slots hdmi or anything	miss from other tablets when using this one is the microsd expansion slots that so many android tablets have
millions of people use their computers for gaming and with the iphone and ipod touch having taken on a clear role as a gaming console that has been as revolutionary for mobile gaming as the wii was for livingroom gaming	you plan on gaming and storing music you may have a bit of trouble with running out of space
you care to pay for the extra space or connectivity is a matter of personal preference i opted for just the 16 gig wifi only model	if you think you might like some extra storage space then i suggest getting the new 32gb model that was recently released

Figure 4.2: Aligned snippet pairs selected by the method comparing two tablets.

compared against the following *baseline*, which simply selects the top most similar snippets as the final output. The only additional restriction is that no snippet may be selected more than once. Note that this simple baseline does not account for redundancy (except for not repeating the same snippet) and coverage of what is important, but only strives to maximize the intersections of snippets.

Comparison of the method with the baseline for selecting 40 pairs is shown in

Table 4.1. It demonstrates that the simple baseline results in substantially worse performance than using the approach with uniform weights. Furthermore, after manually comparing them it appears that the pairs selected by the baseline are qualitatively worse overall than the ones selected by the described approach.

Learning. The goal of learning in this approach is to generalize across product pairs. For example, if we get some labels for comparison of camera models A and B, we would like to use this information to improve the performance for the comparisons of C and D as well. We can expect this to be possible because, for example, indicating that resolution is an important factor most likely applies across all models.

In the experiments for the supervised learning case, an online setting is simulated as we would expect it in a deployed version. For a given product pair that we want to compare, 5 pairs to be presented to the user are selected using the approach. For each individual pair, we receive user feedback expressed as labels as defined in the *Labels* paragraph. After each such iteration, the model is retrained using all the labels obtained so far. This is easily doable due to small number of training examples, but one could also use an incremental learning approach. Furthermore, to facilitate exploration and to simplify the labeling process, the same pair is not allowed to be selected again in the following iterations. In this way 100 labels are obtained by doing 20 iterations of presenting 5 pairs (using weights computed in the previous iteration). Altogether, one annotator labeled 100 training snippet pairs for each of 4 distinct product pairs. The training used a linear support vector regression (using the label values) on a bag-of-words representation (with TFIDF weighting). Note that it is possible to use structured prediction here (similar to how it was done in the Section 3.2). However, this would require

Table 4.1: Average performance scores of *baseline*, using *uniform weights* and *learned weights* across product pairs. the described approach outperforms the baseline and achieves more than 20% improvement on previously unseen product pairs through learning.

method	score	standard error
baseline	7.6	4.2
uniform weights	17.3	0.9
learned weights	22.1	1.1

more complex labels, while simpler independent approach already achieves good performance.

The performance is measured by selecting 40 pairs from a different product pair with a disjoint set of reviews (to avoid any possible overlap in the data) but using the same weights, and computing the score according to the labeling. The reason for selecting a larger set of pairs (40) is to obtain a more robust score, because measuring smaller performance changes on only 5 pairs is unreliable due to low granularity. The results are then averaged across all combinations of one training and one testing product pair. The comparison in Table 4.1 shows more than 20% increase in the performance above the uniformly weighted case (which is already good in itself by looking at the qualitative evaluation on Figure 4.2).

Model variants. I experimented with other possible features in addition to bag-of-words TFIDF scores, but the ones tried did not noticeably improve the score (which is already high for the basic model). Furthermore, minor changes to the scoring function do not immediately break the model from what was observed. Also, selecting pairs individually instead of using the global submodular objective still produces reasonable results, but introduces noticeable amounts of redundancy into the summary as expected.

4.4 Summary

Building upon the framework presented in the previous chapter, this chapter described an approach to selecting pairs of snippets from reviews in a way that creates a summarizing product comparison demonstrating the flexibility – summarization is not limited only to condensing articles into short passages of text, but can also be used to create different kinds of summaries. The modified scoring function strives to select aligned pairs (both snippets are about the same aspect) with good coverage of important aspects and low redundancy. The objective function is again submodular and thus efficient to optimize. The experiments show that it outperforms a naive baseline even with the uniform weights model. Using a supervised learning provides generalization across different product pairs by using labes obtained from user feedback on the presented pairs.

Part III

Unsupervised Approaches to Condensing Information

The previous chapter used supervised learning to improve performance of document-focused summarization approaches. This chapter describes how we can leverage large amounts of unlabeled data to create interesting summaries providing high-level insight into contents of document collections. The focus moves from within-document to corpus-wide temporal evolution of the contents. The two tasks, however, still remain connected via similar objective function – the described method uses the same framework of submodular scoring functions as before, but now uses it to define an unsupervised task.

Currently, we have only rather limited methods for analyzing and understanding electronic archives (reaching back for well over two decades) such as the body of research papers in computer science. While keyword-based retrieval systems allow efficient access to individual documents in archives, we still lack methods for understanding a corpus as a whole. This chapter explores methods that provide a temporal summary of such corpora in terms of landmark documents, authors, and topics. In particular, the temporal nature of influence between documents is explicitly modeled, while summarization is re-interpreted as a coverage problem over words anchored in time. The resulting models provide monotone sub-modular objectives for computing informative and non-redundant summaries over time, which can be efficiently optimized with greedy algorithms. The empirical study shows the effectiveness of the approach over several baselines.

CHAPTER 5

TEMPORAL CORPUS SUMMARIZATION

From news and blogs to twitter feeds, and from research papers to patents, we are accumulating unprecedented amounts of text in digital form. Advances in storage technology have allowed us to maintain complete records of these text streams, and information retrieval research has developed excellent tools for accessing individual documents in the resulting collections. However, our ability to analyze and interpret archives on a macroscopic level is still limited. Macroscopic questions one may ask about a collection range from the creation of a timeline of influential documents or authors, to the automatic summarization of the main chains of discussion.

To answer such macroscopic questions about a corpus of text documents, we can draw upon methods from document summarization (see Chapter 2 and [74]). Instead of summarizing a single (or small number of) individual documents using extracted sentences, the aim is to summarize a collection using extracted documents, authors, or keywords. This shift implies substantial differences in what constitutes a meaningful summary. In particular, time is more important for the creation of corpus summaries than it is for conventional summaries, and we can argue that corpus summaries should reflect the influence that a document or author had on the future development of the collection. Therefore, the summaries now take the form of timelines, where components of a summary are defined with respect to intervals or points of time.

More specifically, we can formulate several variants of the corpus summarization problem. First, we seek to identify k documents that had the largest influence on the content of the corpus. Second, for each point in time, we seek to identify those

documents that were most influential for that time. Third, we similarly want to identify the most influential authors for each time-point. And fourth, we would like to identify key phrases at each time-point that were influential and represent a coherent segment of the corpus.

All four corpus-summarization problems will be formulated as coverage problems, where coverage of words in time approximates coverage of abstract concepts. For conventional summarization and diversified retrieval, coverage approaches [100, 125, 82, 102, 95] and, more generally, submodular summarization methods [51] represent the state of the art. In particular, they provide an elegant model of the relevance/redundancy trade-off inherent in all summarization problems. The key technical challenge for the problem of corpus summarization is the ability to model time-points and time-intervals effectively, without sacrificing the computational approximation guarantees that the greedy algorithm provides for submodular function maximization [72, 35]. This approach models the fact that different ideas have varied novelty, and that the influence of an idea changes over time. An important feature of the approach is that it does not rely on observed or inferred link structure between documents, but requires only the time-stamped document text. This makes the approach applicable to a wide range of corpora for which citation information is not available or not reliable. Furthermore, it allows us to use citation information as "ground truth" for quantitative evaluation. Such evaluations were performed on three scientific corpora and compared to several baselines. Results show that the described method provides qualitatively interesting results.

5.1 Summarization as Coverage

This chapter explores several variants of the corpus summarization problem, providing a temporal summary of the corpus in terms of landmark papers, authors and key-phrases. All approaches are formulated as maximum coverage problems, which have been found to provide elegant and effective methods for conventional summarization and diversified retrieval problems [51, 125, 100]. Let's start by reviewing the coverage-based summarization idea in the remainder of this section, and then extend it to corpus summarization in Section 5.2.

5.1.1 Information Coverage as Word Coverage

Coverage-based summarization methods make a direct analogy between a summary covering the information content in the object to be summarized, and maximum coverage problems as defined in theoretical computer science [35]. The key assumption of coverage-based summarization methods is that coverage of words can be used as a proxy for the coverage of information content. By achieving a good coverage over words, the word coverage approach aims to select a summary which covers different topics. In this way, coverage-based summarization methods elegantly avoid redundancy and promote diversity.

While document summarization involves selecting a diverse set of sentences from it, the idea can be naturally extended to corpus summarization by selecting a diverse set of documents that maximizes coverage. In this approach, every word in a document has a weight associated with it, indicating how important it is to cover this word in the summary. These document weights are either determined through a heuristic [51] or learned [95]. Documents are selected so that the total

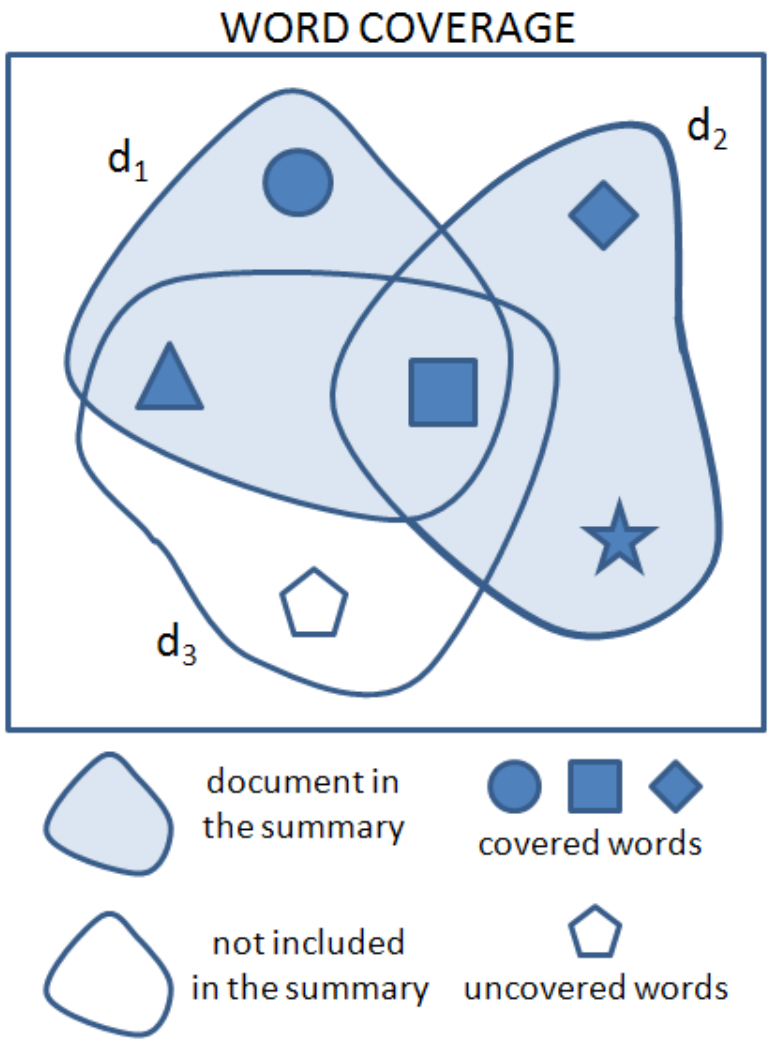


Figure 5.1: Illustration of word-coverage objective.

weight of the covered words is maximized; this is illustrated in Figure 5.1. In this example we want to select at most two documents out of three and each covered word has unit weight. We see that by selecting d_1 and d_2 we achieve the best score of the summary, since we cover the maximum number of words. Note that this is very similar to the problem described in Section 3.1.2, however here we are now selecting documents instead of sentences when constructing a summary.

Formally, let $U = \{d_1, d_2, \ldots\}$ be the set of all documents in the corpus, where

each document is represented as a bag of words. The word coverage objective function, for any $S \subseteq U$, is defined as follows:

$$F(S) = \sum_{w} \theta(w) \max_{d \in S} \phi(d, w), \tag{5.1}$$

where, $\phi(d, w)$ represents the weight of a word w in the document d. One common choice of ϕ is the TFIDF score [84]. Moreover $\theta(w)$ is the weight for the word w depending on our belief of the word's importance. Again, this is similar in general form to the coverage based scoring function used in Section 3.1.2, except that now the candidate set consists of documents instead of sentences and thus the summary can be now described as a set of whole documnets.

5.1.2 Optimization via Greedy Algorithm

With the objective function defining the score of a summarization S, corpus summaries are constructed by finding the set S with the highest score $F(S)$. To obtain a summary we have to solve the following optimization problem:

$$S^* = \arg\max_{S \subseteq U} F(S). \tag{5.2}$$

An important property that enables the fast and accurate solution of this optimization problem lies in the structure of $F(S)$. It is well known that the coverage objective $F(S)$ is monotone (i.e. $|S'| \geq |S| \implies F(S') \geq F(S)$) and submodular [35]. This is now the same setting as was described in Section 3.1.3. The main difference here is that we have a cardinality constraint (instead of length-dependent cost) on the number of selected documents in the summary. The greedy Algorithm 3 now proceeds as follows. The algorithm starts with an empty summary. In each step, a document is added to the summary that results in the maximum

Algorithm 3 for greedy submodular function maximization with cardinality constraint.

$S^* \leftarrow \emptyset$
$A \leftarrow U = \{d_1, ...\}$
while $A \neq \emptyset$ **and** $|S^*| < k$ **do**
  $z \leftarrow \arg\max_{d \in A} F(S^* \cup \{d\}) - F(S^*)$
  $S^* \leftarrow S^* \cup \{z\}$
  $A \leftarrow A \backslash \{z\}$
end while

relative increase of the objective. The algorithm terminates when the budget k is reached.

5.2 Corpus Summarization Problem

While submodular summarization approaches have been very successful for conventional summarization problems, corpus summarization should not only optimize coverage of information content, but also reflect which documents and authors were important in the development of the corpus. In particular, the aim is to include influence between documents into the summarization objective.

To illustrate the difference between a conventional summarization problem and the type of corpus summarization envisioned here, consider a corpus consisting of research papers covering two decades of a particular field. In such a scenario, conventional word-based coverage approaches would pick several (non-redundant) survey papers, or papers that otherwise touch on a lot of different areas, since their union will tend to cover the largest subset of words in the corpus. However, while these survey papers are indeed a good summary of the content, this selection will not provide any information about how the corpus developed over time, what papers opened new areas of activity, and which authors influenced the

direction of the field. The subsequent sub-sections will show how the conventional coverage-based approach can be extended to provide summaries that not only optimize information content, but also reflect influence and importance of individual documents and authors.

To achieve this goal, the remainder of this section shows how to (a) incorporate time into the summarization problems, (b) formulate the summarization objective in terms of influence, and (c) show how corpora can be summarized not only through landmark documents, but also through influential authors and key-phrases.

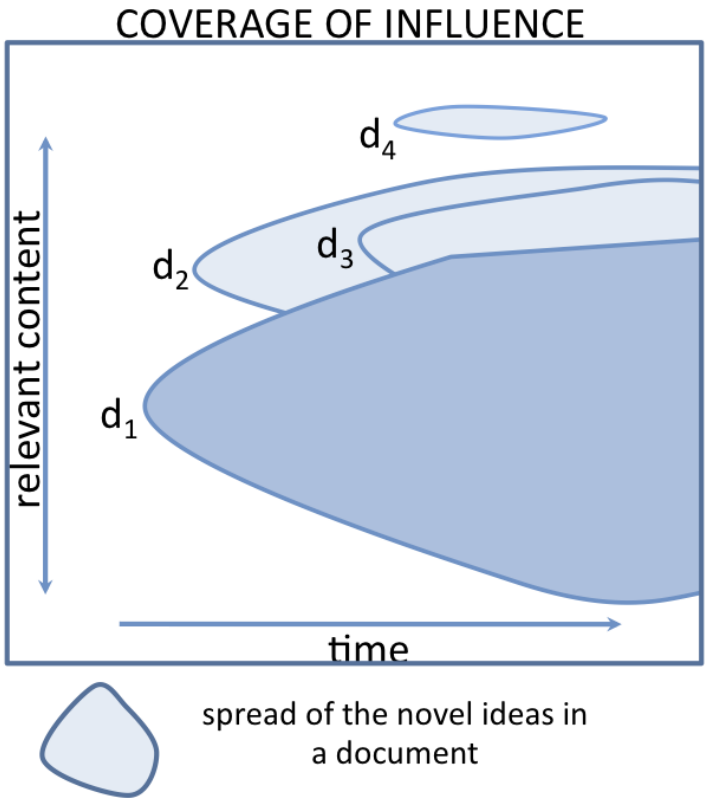


Figure 5.2: Illustrating the coverage function for revealing influential documents.

5.2.1 Summarization through Influential Documents

Let's now explore how the word-based coverage objective can be extended to summarize a corpus through a non-redundant set of influential documents. A pictorial illustration is shown in Figure 5.2, indicating how influential documents introduce ideas that increasingly cover the content of documents observed in later years. To get to an operational formalization of influence in the coverage model, we start with the following properties:

Spread: An influential document contains ideas that spread to other documents. The more an idea spreads, the greater is its influence. Note that this aspect of influence requires us to include a notion of time into the coverage objective, since ideas can only spread forward in time.

Novelty: A document should only be credited for generating influence with respect to some idea, if this idea was first proposed in that document. If an earlier document already contained that idea, influence should be credited to that earlier document. Note that this can be quite different from citation-based impact measures, which may credit influence to review papers or other papers that popularize an existing idea.

To capture *novelty* in the word-based coverage approach, we can redefine $\phi(d, w)$ in the coverage objective (5.1). Intuitively, we do not want to give document d credit for an idea – as represented by its word distribution – if there already exist older documents d', $t(d') < t(d)$ that already cover this idea. $t(d)$ denotes the year of publication of the document d. More formally, let $\mathcal{N}(d)$ denote the k-nearest neighbors (for example, based on cosine similarity) of the document d among all the documents published before it, then we capture the novel contribution of a

document as

$$\nu(d, w) = \max\left\{0, \min_{d' \in \mathcal{N}(d)} \{\phi(d, w) - \phi(d', w)\}\right\}. \tag{5.3}$$

In order to capture *spread* in the coverage objective, we enlarge the set of objects that need to be covered to word-time pairs w_y for all words w and years y. This allows the coverage objective to make a distinction between covering the word w in a year y and covering the same word in year y'. Formally, we generalize $\theta(w)$ in (5.1) and make it dependent on time, where each $\theta(w, y)$ separately defines how important word w is for the given time y. We say that the importance of a word w in year y is determined by the sum of the TFIDF scores of the documents in year y:

$$\theta(w, y) = \sum_{d:t(d)=y} TFIDF(d, w). \tag{5.4}$$

Other weighting schemes can work here too (e.g. per-year inverse document frequencies) because the model is oblivious to the precise choice of θ (as long as it well represents the spread). Note that this allows for modeling the fact that some documents cover a word in certain years, but not in others. In particular, we say that a document d only covers a word in those years that are later than its publication date $t(d)$. This allows us to formulate the objective for finding influential papers as follows:

$$F(S) = \sum_{w} \sum_{y} \theta(w, y) \max_{d \in S, y > t(d)} \nu(d, w). \tag{5.5}$$

The above objective multiplies the novel aspect of a word in a paper with how important a word is in the future years. Intuitively, the score is large when the set of selected documents S contains documents with high novelty scores as well as a high influence in the future. We can maximize the above objective using the greedy Algorithm 3.

Note that this approach will not tend to select survey papers unlike the word coverage approach for two reasons. First (and most importantly), a survey paper will have a low novelty score since it is mostly based on previous work. Second, usually survey papers are written after a field is well developed, hence it does not cover all those documents that appeared before it in (5.5).

The key feature that differentiates this model from related coverage-based approaches is the insight that modelling the temporal context in which words appear (and not just the within-document context) can provide a strong signal for summarization tasks. This is achieved by not just using words as proxies for ideas in a coverage objective, but anchor words at specific time-points to gauge the influence. Additionally, modelling novelty helps us correctly attribute impact and avoids the pitfalls of citation-based impact measures.

5.2.2 Timelines of Document Influence

The previous section showed how the coverage objective can be extended to focus on influential papers, producing summaries that are organized by the publication date of the influential documents. However, dual to such summaries, we may also ask the following question: for each year y, what are the documents that most influenced the content of this year? This is illustrated in Figure 5.3.

The following subsection formulates a coverage objective that identifies the k documents that had the most influence in a year (for every year). Intuitively, a document d influences a year y, if it was published before year y, i.e. $t(d) < y$, and the novel ideas from d have substantial coverage in year y. Instead of selecting documents independent of year as in the previous section, we now allow our method

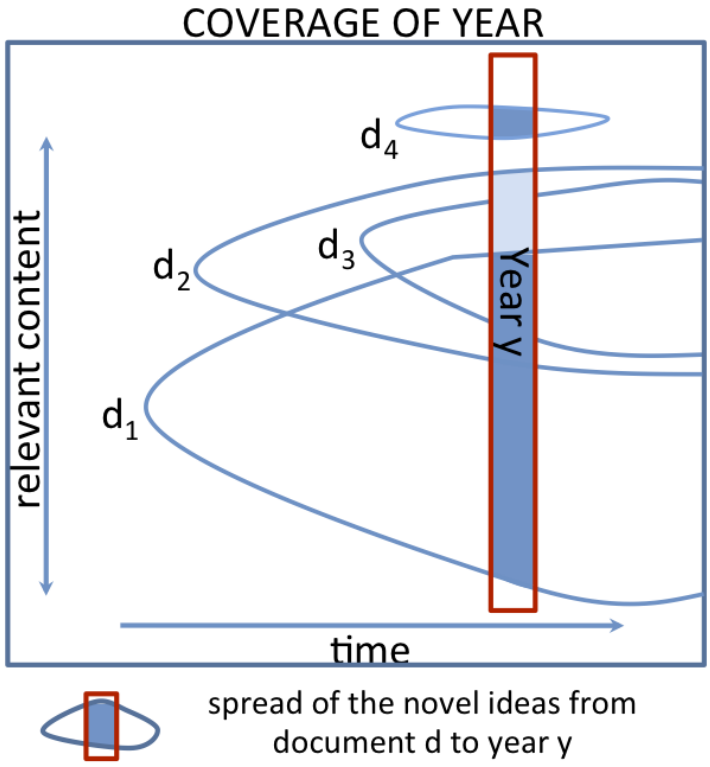


Figure 5.3: Illustrating the influence of documents in a particular year.

to select an influential document to cover a particular year y. This means that our optimization problem now selects from a universe of document-year pairs $U_y = \{(d_i, y_i), ...\}$. Here d_i is any document from the corpus, and y_i is any year such that $y_i > t(d_i)$. This leads to the following objective which we seek to maximize.

$$F(S) = \sum_{w} \sum_{y} \theta(w, y) \max_{\substack{(d,\, y_d) \in S \\ y = y_d > t(d)}} \nu(d, w). \tag{5.6}$$

Similar to (5.5), the above objective multiplies the novelty score of word w in a document d with the importance of the word for a year y. However, (5.6) allows picking a different set of documents for each year y_i.

It is easy to see that $F(S)$ in (5.6) decomposes into a set of independent optimization problems – one for each year. We may therefore solve the following

subproblem separately for each year and concatenate the solution for each year to obtain the solution of the original problem. Formally,

$$F_y(S_y) = \sum_w \theta(w, y) \max_{d \in S_y, y > t(d)} \nu(d, w). \tag{5.7}$$

Each of the above objectives is monotone submodular and can be solved using the greedy Algorithm 3.

5.2.3 Timelines of Author Influence

Analogous to selecting documents that had a large influence on a given year, we can also ask which authors were most influential. It is easy to extend the optimization problem from the previous section so that it selects influential authors. Denote with $d(a)$ the documents in the corpus that were authored by author a. The universe of items to select from now consists of author-year pairs $U_y = \{(a_i, y_i), ...\}$. Selecting an author a_i for year y_i implies that all documents the author wrote before year y_i get selected. This leads to the following objective,

$$F(S) = \sum_w \sum_y \theta(w, y) \max_{\substack{(a,\, y_a) \,\in\, S \\ d \,\in\, d(a) \\ y \,=\, y_a \,>\, t(d)}} \nu(d, w). \tag{5.8}$$

which again can be broken into independent optimization problems for each year.

5.2.4 Summarizing Timelines with Key-Phrases

Summaries in terms of documents and authors still require the user to read through some documents from the collection. Let's now explore whether timelines of influence can be summarized through key-phrases. In particular, the aim is to identify

the points in time when new and influential ideas – as represented by a key-phrase – entered the collection.

While we already have operational definitions of novelty and influence, we still need to define what makes a key-phrase a good representative of an idea. We can conjecture that a key-phrase that represents an idea well will be accompanied by stable word distribution over the years. For instance, documents that mention the phrase "HITS algorithm" will probably also mention several words related to that idea, whereas documents mentioning "Related work" need not have such a coherent set of overlapping words. The keyphrase T1 in Figure 5.4 is an example of a good key-phrase, since documents that contain T1 also share many other words. On the other hand, a key-phrase that is not a good representative of an idea may occur in documents talking about a variety of different ideas. T2 in Figure 5.4 is an example of a bad key-phrase.

Let's formalize this definition of key-phrases as follows. Define the universe of elements to choose from, $U = \{(p, y), \dots\}$, where p is a candidate key-phrase and y denotes the year when the key-phrase became influential. Let the subset of the corpus that mentions a candidate key-phrase p be D_p. Intuitively, we wish to associate with (p, y) a representative document $d^* \in D_p$ which was published in year y and which was the most influential document in the subsequent development of D_p. According to the conjecture, for a bogus keyphrase, the associated d^* will achieve very poor coverage of the word content observed in documents of D_p that were published after y, while influential keyphrases will have a document that covers the associated stable word distribution very well. Following (5.4), we can model the importance of covering a word in D_p as $\theta(w, y)^p$. More precisely,

$$\theta(w, y)^p = \sum_{d \in D_p : t(d) = y} TFIDF(d, w).$$

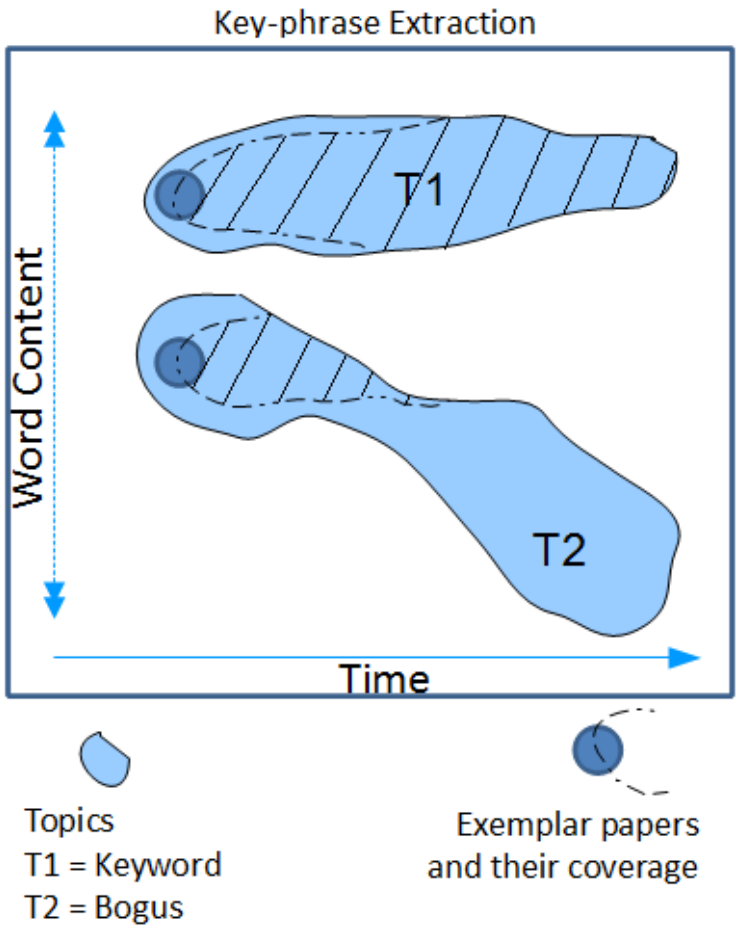


Figure 5.4: Illustrating the difference in word distributions over time between a bogus term and a genuine keyword.

$$d^*(p, y) = \underset{d \in D_p : t(d) = y}{\arg\max} \sum_w \sum_{y' > y} \theta(w, y')^p \nu(d, w).$$

With this d^* for each element in U, we can formulate the objective

$$F(S) = \sum_w \sum_y \max_{\substack{(p_i, y_i) \in S \\ y = y_i}} \nu(d^*(p_i, y_i), w) \theta(w, y)^{p_i}. \tag{5.9}$$

Again, the objective decomposes into independent sub-problems for each year, and

Algorithm 4 for greedy submodular function maximization with budget constraint.

$S^* \leftarrow \emptyset$
$A \leftarrow U = \{p_1, ...\}$
$z^* \leftarrow \underset{p \in A | C(p) < k}{\arg\max}\ F(\{p\})$
while $A \neq \emptyset$ **and** $C(S^*) < k$ **do**
  $z \leftarrow \underset{p \in A}{\arg\max} \dfrac{F(S^* \cup \{p\}) - F(S^*)}{C(p)}$
  $S^* \leftarrow S^* \cup \{z\}$
  $A \leftarrow A \backslash \{z\}$
end while
if $F(S^*) < F(z^*)$ **then**
  return z^*
else
  return S^*
end if

we can rewrite it for each year y as,

$$F_y(S) = \sum_w \max_{(p_i, y_i) \in S, y_i = y} \nu(d^*(p_i, y_i), w)\theta(w, y)^{p_i}. \tag{5.10}$$

Unlike in the previous optimization problems, we now associate a cost $C(p, y) = |\{d \in D_p : t(d) = y\}|$ with each element of S. This is done to encourage associating a key-phrase with the point in time when it begins to gain popularity. The number of documents published in a year mentioning a key-phrase is used as a proxy for the maturity of an idea. The optimization problem is,

$$S_y^* = \underset{S \subset U}{\arg\max} F_y(S)$$
$$\text{subject to} \sum_{(p,y) \in S} C(p, y) \leq K.$$

This formulation is an instance of the budgeted coverage problem with a linear cost constraint, and the greedy Algorithm 4 is $(1 - 1/\sqrt{e})$ optimal [51, 35].

5.2.5 Alternate Formulation using Global Optimization and Adaptive Budget

For simplicity, the timeline-generating optimization problems so far were decomposed into independent sub-problems, one for each year. This was possible since we imposed a cardinality constraint for each year. However, we can also define a global optimization problem across all years that constrains the maximum amount of content covered in each year. This results in a summary that will choose more documents from years that actually contain more interesting information. Formally, we change the global objective from (5.6) into

$$F(S) = \sum_{w} \sum_{y} \min\{F_y(S_y), \tau F_y(U_y)\}, \tag{5.11}$$

where the parameter τ determines how much relative word content per year we want to cover, $F_y(S_y)$ is as defined in (5.7) and U_y is the whole universe for year y. This global $F(S)$ is also monotone submodular and can be solved using the greedy Algorithm 3. Whereas earlier we had a cardinality constraint parameter k to set for each year, we have to set one global parameter τ (setting it higher results in more detailed summary) and one global k (higher values result in longer summary) now.

5.3 Evaluation

In this section, the proposed models are empirically evaluated on publicly available datasets. I first describe the datasets and then present the results of the experiments along with evaluation metrics. The experimental results show the advantage

of the approaches compared to other baselines in addition to good qualitative results.

5.3.1 Datasets

Three corpora containing research publications were used for evaluating the proposed approaches. The Neural Information Processing Systems (NIPS) corpus contains 1955 published papers over a span of 14 years. Similarly, the Association for Computational Linguistics (ACL) corpus [79] contains 18041 papers published in a number of conferences over a span of 39 years. A set of papers published in the proceedings of SIGIR and CIKM conferences over the years was collected from CiteSeer. This corpus contains 2097 papers published over a span of 18 years (the last year being 2007). In all cases, each document (paper) is associated with its publication year and the collection is limitd to 12 consecutive years ending at the year before last (the citation graph is also constrained only to those years). Since novelty of a document is computed based on the nearest neighbors from the past, the year immediately before this subset is used for this purpose in the subsequent experiments. The last year is skipped because it does not have any citations from the future. The early years in ACL and SIGIR-CIKM corpus were not used because they contain significantly less papers and citations compared to other years.

All datasets include citation graphs which are used for evaluation purposes, however the method does not require citation information and could thus be easily applied to other document collections. Note that the citation graphs are sparse as they do not include references to and from the papers outside the corpus. The NIPS collection has 1512, the ACL collection has 82892 and the SIGIR-CIKM

collection has 1750 citations between papers inside the corpus. In addition to regular research papers, NIPS corpus also contains meta documents representing volume indices. Such documents were manually removed since they are very easy to spot. Also, the words were pruned and only those words which occurred at least twice in a document and in at least three documents in the corpus were retained. This simple heuristic removed a lot of noise introduced by the OCR system, and allowed for meaningfully interpreting the influence. Every document is represented by the TFIDF score (computed on the whole corpus) of the words contained in it after pruning, and the resulting document vector is normalized to unit length. The cosine similarity between the document vectors is used to compute the nearest neighbor in the past (to determine novelty). The exact nearest neighbors are not required, and in the case of a very large corpus, approximate methods to find similar documents can be employed to sidestep the quadratic time complexity of this step.

5.3.2 Influential Documents

The word coverage approach (from Section 5.1) obtains a summary by maximizing the word coverage. Section 5.2.1 argued that influential documents have novel ideas which subsequently spread through the corpus. This experiment selects the most influential papers based on the objective (5.5) which captures novelty of a document and its sphere of influence, and compares it with those selected by the simple word coverage objective (5.1). Since there is no standard way of measuring the influence of a paper, the citation structure available in the corpora is used instead. The total citation count for the set of papers (i.e., the number of times these papers were cited by documents in the corpus) selected by any algorithm. is

used to quantitatively evaluate whether the selected papers were indeed influential. There have been several criticisms of citation-based impact measures and some effort [12, 119] addresses them. However, for a comparative study, citation counts appera to be the least biased choice in this setup.

Several baselines are considered to provide a point of comparison for the coverage based approaches. The simplest baseline is to randomly select documents until the budget is reached (*random*). Another baseline that was considered is to select the most prolific authors in the conference (in terms of number of accepted papers) and then select the required number of papers from the union of their papers (*authors*). More concretely, the authors are first ranked according to the number of papers in the collection they authored. Next, the 10 most prolific authors are selected. Finally, the same number of papers for each author are sampled uniformly at random with replacement from the set of authored papers. Selecting papers with the highest observed citation inlinks in the corpus (*bound*) is used to compute the upper bound on the total possible citation count.

In the experiment, 100 documents are selected from the NIPS, SIGIR-CIKM and ACL corpus that maximize the respective objective function. The experiments presented here are using unigrams as the elements in the universe. However, the methods can use other types of elements (e.g. bigrams formed from consecutive words, for which we observed a similar trend in the results). The results of this experiments are provided in Table 5.1. Table also also provides standard errors of these results; they were estimated from the citation count on 10 re-runs of 70% subsampling of the corpus.

From the table, it is clear that the new approach (*infl. papers*) gets significantly higher citations compared to the word-coverage approach (*word cover*) in

all corpora. Moreover, finding influential papers is computationally cheap (with running time linear in the size of the corpus multiplied by the number of selected papers if we do not count the preprocessing step of computing nearest neighbors for novelty score) and, for example, takes a few seconds for the NIPS corpus on a standard desktop computer.

method	NIPS	ACL	SIGIR-CIKM
Random	64 (4.6)	422 (41)	84 (7.9)
Authors	115 (4.0)	1097 (49)	86 (4.8)
Word Cover	92 (2.7)	799 (170)	96 (5.2)
Infl. Papers	196 (12.8)	1842 (111)	217 (14.5)
Bound	521 (11.0)	9787 (143)	815 (13.5)

Table 5.1: Total citations obtained by the papers selected for influential documents and baselines using unigrams. All results use 1-NN for novelty score. The values in parentheses indicate standard error.

5.3.3 Impact of Using Novelty Score

The approach uses the novelty score (introduced in Section 5.2.1, Eq. (5.3)) to credit a document for an idea only if it was the first one proposing it. Novelty is captured by considering k nearest neighbors in the past and subtracting their word weights (clipping at 0 to prevent negative values) from the current document. This subsection explores the impact of choosing different values of parameter k.

Results for the word coverage approach on NIPS and ACL (as examples of two slightly different behaviors) using unigrams or bigrams as elements in the universe are presented in Figure 5.5 . We would expect word coverage to improve when using novelty scores because the coverage most likely does not choose the initial (highly cited) paper but some later one with better coverage (e.g. a derivative paper that also incorporates some other ideas). This intuition is confirmed by the results

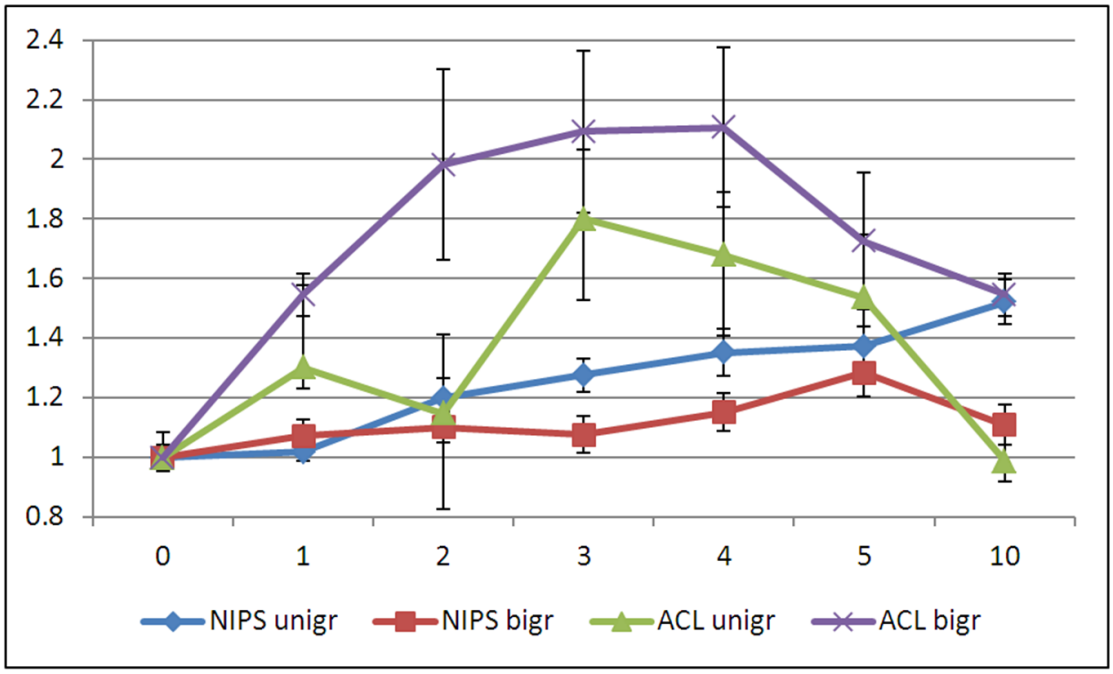


Figure 5.5: Comparison of results of word coverage approach when using different values of k (number of nearest neighbors for computing novelty score) on NIPS and ACL corpus for unigrams and bigrams. The horizontal axis represents the value of k and the vertical axis relative performance (number of citations) when compared to not using the novelty score (i.e. $k = 0$).

showing that using more neighbors improves the score as we incorporate more and more information about novelty. After a point we can see that performance starts dropping again because we are subtracting too much content.

Almost all coverage approaches benefit from using 1-NN, but increasing k only improves performance for word coverage approach. I believe that using 1-NN helps because it mimics a language background model and penalizes frequent non-content words, while increasing k above that does not bring significant benefits because we already model temporal behavior with the choice of our model.

5.3.4 Timelines of Document Influence

This sub-section evaluates the approach to create timelines of document influence and compares it against several other baselines. For each NIPS, ACL and SIGIR-CIKM corpus, 10 documents are selected for each year.

Again, we can consider the random baseline (*random*) and the 10 most prolific authors (*authors*). The authors baseline is constructed as follows: first create a union of all papers by 10 authors with the highest number of accepted papers, and for each year select 10 documents randomly from this union (with replacement) published on or before this year. The upper bound on the citations (*bound*) is computed by selecting papers with highest citation count in a given year (i.e. we count only citations occurring in that particular year) – let's call this the *current citations*.

The selections are evaluated based on the citation network as before. In the previous section, the evaluation was based on the total number of citations a paper obtained. However, in this section, it is based on the current citations. To select a timeline of influential papers, we select papers that have maximum influence in a particular year (for each year). So, to quantitatively evaluate the selections, if a paper is selected as influential in the year y, we should count the number of citations it gets in the year y (i.e. only citations from papers citing it in this year count) and then sum them across all years.

Results for this experiment are summarized in Table 5.2. We can see that *random* baseline and *authors* have inferior performance compared to described approach (*timeline*). Note that the gap between our approach (*timeline*) and the *bound* is larger than in the influential papers experiment. I believe this is due

to *timeline* being an inherently harder problem – not only do we have to find influential papers but we also have to specify exactly when were they influential (as the evaluation metric counts only citations from papers citing in that selected year). The approach to constructing timelines is fast to compute (time complexity is linear in the number of years, papers selected and corpus size) and, e.g., takes less than 3 seconds on the NIPS corpus on a standard desktop computer.

method	NIPS	ACL	SIGIR-CIKM
Random	14 (1.4)	85 (11)	11 (1.2)
Authors	14 (1.2)	84 (14)	7 (1.0)
Timeline	60 (3.0)	190 (14)	30 (1.9)
Bound	269 (3.0)	3316 (11)	367 (4.1)

Table 5.2: Current citations (i.e. number of citations from papers citing in that particular year) obtained by the papers selected for timeline and baselines using unigrams as elements of the universe. All results are for 10 re-runs of 70% subsampling and using 1-NN for novelty score. The values in parentheses indicate standard error.

5.3.5 Timelines of Author Influence

This section describes experiments based on the objective proposed in Section 5.2.3 to select timelines of Author Influence. Instead of selecting papers we now consider meta-documents describing authors and construct a timeline showing which authors were important and when. In addition to this, for each selected author we constrain the corpus to that author's papers and find the most influential ones (including their timeframe of prominence).

A visualization of the results for NIPS corpus is presented on Figure 5.6. By just looking at the plot it is easy to gain some insight into the development and content of the corpus. Features such as some authors having an influence throughout the

whole corpus (e.g. Jordan, Sejnowski) are easy to spot. We can also see that some authors have had more influence only in specific timeframes (e.g. Hinton in the early years and Smola in the later years). In addition, looking at the selection of an author's most influential papers gives us insight into what topics they usually write about.

Although there are quantitative metrics which might be used to judge the output of the system to pick influential authors (e.g. *H-index*), note that we require these metrics to be computed for the timeframe of the collection only. Given the very sparse citation graph, using only observed intra-collection citations is expected to be a noisy signal. Qualitative results clearly indicate good performance of our approach and I feel that any simple adaptation of existing measures would not give a significantly better insight.

5.3.6 Key-phrase Extraction

Experiments to find prominent key-phrases in each of the scientific corpora were ran with the formulation described in 5.2.4. The trigrams and bigrams that occurred in at least 0.2% of the documents in the respective corpus were considered for the set of candidate key-phrases. Moreover, if a trigram is admitted to the set of candidate key-phrases, the constituent bigrams are not considered as candidates. This is a simple heuristic that recognizes that the lexical unit for phrases is usually a trigram or bigram and greedily prefers trigrams. More sophisticated ways to determine the set of candidates are possible, say independently running a Part-Of-Speech tagger and considering only noun phrases. The number of candidate key-phrases using this heuristic rule is 3035 for the NIPS corpus, 8139 for the SIGIR-CIKM corpus and 4687 for the ACL corpus. The fewer number of candidates in the ACL corpus is

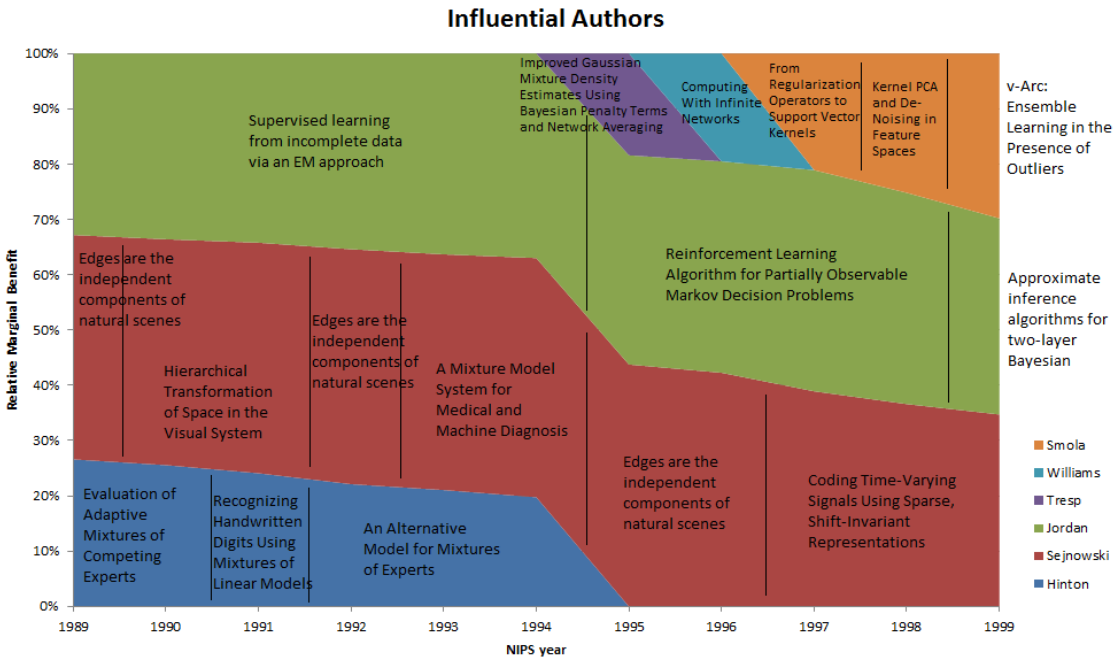


Figure 5.6: An example of applying our framework to select the most important authors and the most important papers for the given authors using our framework. In each of 11 consecutive years of NIPS three authors are selected and then that author's most influential paper is selected in a particular year. The width of the author's slice (relative marginal benefit) represents the importance relative to other selected authors (which is computed as decrease in objective score if we remove that author from the collected set).

explained by the fact that requiring the document frequency of bigrams or trigrams to be 0.2% of a much larger corpus is a more restrictive filter.

There are no ground truth key-phrases to evaluate the output of the system; also, it is hard to quantitatively judge the quality of an influential key-phrase's associated timestamp. Therefore the average number of citations in the collection of documents that mentions a key-phrase is used as a measure of its quality. Concretely, for a key-phrase p and the subset of the corpus D_p that mentions it,

$$Score(p) = \sum_{d \in D_p; d' \in D_p; d' \neq d} Cite(d \leftarrow d') / |D_p|$$

where $Cite(d \leftarrow d')$ indicates that document d is cited by d'. The objective in

Section 5.2.4 is optimized for each year with a budget of 3, and the set of all unique key-phrases is then collected. The reported scores for this approach (presented as *TimeCov* in Table 5.4) are the sum of $Score(p)$ for each unique collected key-phrase p. As a point of comparison, the number of unique key-phrases collected as *Count* is also reported. A simple baseline for this experiment would be to pick the most frequent key-phrases occurring in the corpus in each year: this approach is hindered by the frequent occurrence of redundant phrases. For instance, "neural network" in the NIPS corpus,"natural language" in the ACL corpus and "information retrieval" in the SIGIR-CIKM corpus appear in such an overwhelming majority of documents over all the years as to drown out other informative candidates. This baseline is reported as *MostFreq* in the results. Another approach we can compare with is to pick candidates that optimize the *Score* directly in each year; this can be interpreted as an upper bound for this evaluation metric. Table 5.3 also provides the collected candidates from the coverage approach and one that optimizes the $Score(t)$ directly for the SIGIR-CIKM corpus. Several informative phrases that come from diverse areas of research covered in SIGIR and CIKM get selected in the coverage approach. Furthermore, a visualization of the key-phrases over years for the NIPS corpus is shown in Fig. 5.7. Area of the shaded region corresponding to a term represents the fraction of documents observed in the corpus in that year that mention that particular term.

5.4 Summary

This chapter presented a submodular framework for temporal corpus summarization. The notion of word coverage was extended by asserting that summaries cover important concepts by covering associated words over a time interval. A timeline

Year	Coverage		CiteScore	
	Keyword	Marginal Influence	Keyword Keyword	Marginal Influence
1995	relevant document	17.240	information retrieval	20.0
	query expansion	5.464	singular value	5.0
			training set	5.0
1996	search engine	8.228	speech recognition	3.5
	semantic indexing lsi	6.034	block size	2.0
	filtering system	2.818		
1997	web search	11.853	retrieval system	5.0
	training data	7.871	test collection	5.0
	language model	6.760		
1998	language model	5.411	summarization system	8.0
	retrieval model	4.799	naive bayes	7.0
	learning algorithm	4.024	unjudged document	6.0
1999	language model	12.358	general english	10.0
			pearson correlation	5.0
2000	clustering result	4.354	cumulative gain	8.0
	document model	3.130	expansion term	4.0
	cori algorithm	2.645	event detection	4.0
2001	cross-language inf.	4.897	smoothing method	14.5
	user information	3.450	topic distillation	7.0
	hits algorithm	3.044		
2002	training example	2.099	translation disambig.	3.0
	term dependency	1.868	hoc retrieval	3.0
	input stream	1.698		
2003	query language	2.509	image feature	5.0
	feature selection	1.779	finding expert	5.0
	document clustering	1.751	novelty detection	4.0
2004	training image	1.887	regularized logistic regr.	3.0
	inverted index	1.413	label information	3.0
	element retrieval	1.100	web browser	3.0
2005	xml retrieval	3.926	existing retrieval fun.	3.0
			new system	2.0
			index construction	2.0

Table 5.3: The list of key-phrases for SIGIR-CIKM selected by the greedy algorithm solving the budgeted coverage problem with budget of 3 and by optimizing the citation score.

of influential documents, or authors, or coherent key phrases was constructed using this approach, providing concrete suggestions for further and more detailed

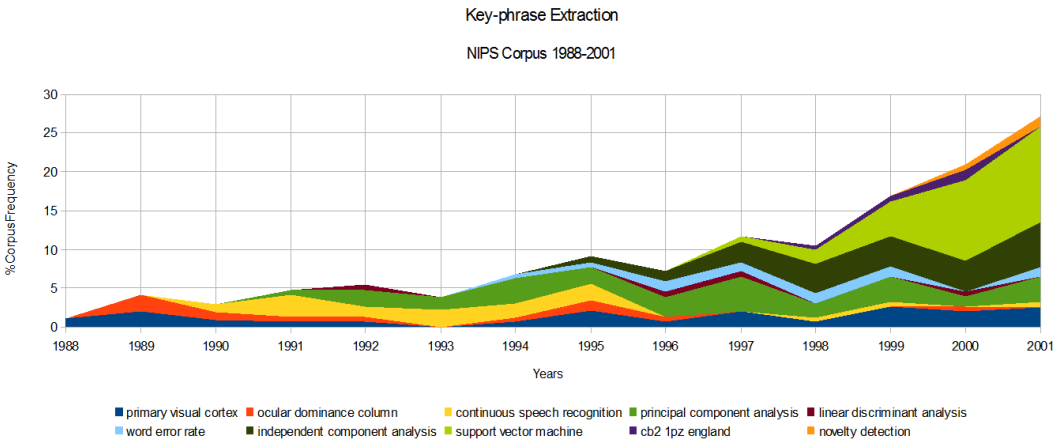


Figure 5.7: A timeline showing the evolution of the key-phrases selected by the coverage approach in the NIPS corpus.

	NIPS		ACL		SIGIR-CIKM	
Method	Count	Score	Count	Score	Count	Score
MostFreq	13	2.13	20	34.33	20	7.41
TimeCov	17	4.68	77	116.92	29	12.28
Bound	13	6.09	96	124.10	29	18.82

Table 5.4: Quantitative results of keyphrase extraction.

exploration of the corpus contents. The approach leveraged both the novelty of a document as well as its influence in the development of the corpus and relied only on word features; in particular, it does not require a citation structure to infer influence across time. Therefore it is applicable to any textual collection which provides timestamped documents. The optimization objectives used monotone submodular functions to trade-off relevance and redundancy elegantly, and were again solved using an efficient greedy algorithm with a constant factor approximation guarantee. Experiments empirically demonstrated that the approach performs better than several baselines using citation based performance measures. Results also include qualitative timelines for a few scientific corpora.

Part IV

Crowdsourced Approaches to Condensing Information

The previous chapters described how we can use summarization to condense information and thus answer some user needs. One of the challenges was specifying what is a good summary and obtaining labeled data. However, a possible alternative is to take by the users for the users approach. Let users express their wants by giving us feedback on presented content, which in turn can then be used to rank and surface good content (where "good" is defined by the users themselves) for their consumption.

The following sections first try to understand if users are giving us straightforward answers with their feedback and if not, what are the biases in their voting behavior. The second part of this chapter then shows how we can use the better understanding of the users' voting model gained form real-world data to improve the ranking algorithm.

CHAPTER 6

CONTENT RATING BEHAVIOR ON UGC

User-generated content (UGC) is now one of the primary sources of useful content on the Web. But while there is a tremendous volume of it — thanks to a lack of barrier to contribution — not all of it is equally good. This means that sorting and ranking content is essential to making UGC actually useful to a site's viewers. To this effect, most sites with user-generated content — such as reviews on Amazon, answers on online Q&A sites like StackOverflow, articles on Digg or Reddit, or comments on news articles and YouTube videos — allow viewers to rate content and use these ratings to determine the order in which to display contributions.

In an ideal world, users would respond to questions about rating — such as Amazon's "Was this review helpful to you? Yes/No" query — by judging each contribution exclusively based on its absolute merits, independent of the contribution's display context or its previous ratings. But does real user voting behavior resemble this ideal?

Addressing this question and understanding user rating behavior is important for more than one reason. First, when contributions are displayed in order of aggregate user ratings (such as the ratio of "yes" votes to total votes), whether higher quality contributions are indeed ranked higher or not depends on *what a vote actually means*: if user voting behavior is such that votes are unfavorably biased by other factors, the resulting ranking may not be the one the website seeks. Second, an accurate model of user behavior is important for designing optimal algorithms to quickly *learn* contributions' qualities from votes. In particular, the performance of learning algorithms based on user ratings will depend on correctly interpreting

the information conveyed by the votes in the first place.

This chapter address the question of how users rate content, using data collected over 5 months on 595 products from Amazon, with daily statistics on the received votes for each review in this set of products. An *absolute* rating model, where users cast votes that depend purely on judging a review's merit in isolation, is inaccurate and does not fit observed voting patterns. Instead, users appear to cast votes that reflect *relative*, rather than absolute, judgments about reviews' quality. In particular, both *how* a user rates the review, as well as *whether* a user votes on it at all, varies with the review's context — the relative quality of the surrounding reviews, as well as its current ranking due to ratings from previous voters.

Note that ratings on Amazon reviews provide a relatively "neutral" sample of user-generated content, which makes them a good environment for understand voting behavior. In other rating environments, such as comments on news articles or answers on Q&A sites, votes could (and are anecdotally known to) indicate not only content quality, but also "Agree/Disagree" or "I like/dislike your opinion". In contrast, reviews on Amazon — at least for the vast majority of non-controversial products[1] — are rated by users *before* the user has experienced the product (since users would typically read reviews on Amazon to help decide whether or not to purchase a product), so that thumbs-up/down ratings of reviews on Amazon are more likely to relate to the review's quality rather than to reflect agreement or differences of opinion with it.

[1]Some books on Amazon do appear to have highly polarized reviews.

6.1 Voting Dataset

To understand how users vote on online content, we need a dataset that contains information on how votes are cast. Publicly available datasets on user-generated content are typically snapshots of a particular site (such as Amazon) at a specific point in time, containing information about the content on the site at that point, current rankings of the content (*i.e.*, the order in which contributions are displayed), and cumulative ratings (e.g., how many users found each contribution helpful during its lifetime up until the snapshot). While this allows reasoning about, for example, how the content of a contribution influences the votes it receives, it is not sufficient to address voting behavior — when and why the contributions accumulated the votes they did.

This led to a creation of a new dataset, which is scraped from publicly viewable data on Amazon. For every product, Amazon displays a list of all its reviews, as well as the accumulated votes on each review — how many Amazon users rated that particular review helpful (or not), up-to that point. A python script was used to retrieve and parse these web pages to obtain a sequence of snapshots that contained (for every product) the list of reviews, in what order Amazon displayed these reviews (*i.e.*, their current rankings), and the number of "yes" and "no" votes for each review.

A set of 595 products[2] was chosen from the top 100 products of six Amazon "Hot New Releases" lists (Books, Video Games, Music, Movies & TV, Toys & Games, Electronics) as of 2nd October 2012. These products were tracked *daily* for a period of 5 months from October 2012 to March 2013. Over this period, the script was periodically ran to collect data on the 50 most helpful reviews (or

[2]Five items in two lists did not parse correctly and were automatically excluded.

fewer, if there are less than 50 reviews) for each of these products. The choices in data collection were driven by the following reasons. (i) First, a subset of products was chosen rather than all products because tracking all products on Amazon would be unjustifiably resource-intensive. (ii) The focus was on popular products which receive more reviews than niche products[3] to ensure adequate data points for each review. (iii) Finally, following products from 'the beginning', *i.e.*, from the time of their launch on Amazon or close to it is desired because we want to include early votes on early reviews on products in the dataset. Amazon's "Hot New Releases" lists were used as a proxy for future popularity, and to address the problem of choosing a subset of popular products which also allows observing the early voting dynamics: these lists contain new (just or soon-to-be released) products that Amazon expects to be popular enough to satisfy the criterion for (predicted) high reviewing and voting volume. The tracked products were all selected at the same time, which was at the beginning of the data collection period. Each product is restrictd to no more than 50 top reviews to politeness concerns, since there are a small number of popular products that are collecting thousands of reviews — a single pass using a reasonable delay between HTTP requests already required over an hour, even when restricted to 595 products with at most 50 reviews per product.

The interval between the script runs was one day (although the time of day when the script ran was not fixed, and changed over time). This lead to a dataset containing 150 daily snapshots, 71504 reviews and 497088 votes. The key feature of this dataset is that it allows us to study how votes were cast over time, and as a function of the context in which the review was viewed. In particular, we can see

[3]Note that this means that the observations regarding voting behavior are possibly only valid for reviews on popular products. However, this is arguably the most relevant set of reviews for which to study voting behavior, since reviews on frequently-purchased products are likely the most useful category of reviews on Amazon.

how many votes a review received on a particular day, its rank on that day, and how other reviews were ranked in relation to it.

Note that there are some important caveats about the data, however. (i) The data only consists of daily snapshots and not *individual* user interactions, so that users, and rankings, are aggregated at the daily level. Specifically, this means that if the ranking changed more often than once per day, we have a mismatch between the data and some actual users' experiences, which can blur the measurements. (ii) We do not have a way of tracking page views. This is relevant to the analysis of participation, *i.e.*, when users choose to vote; see Section6.4. (iii) Finally, there are a few instances of incorrect parsing due to changes in Amazon's webpage structure, which needed to be manually corrected for the final dataset. Nevertheless, there remain rare instances of mismatched data as a natural consequence of the temporal nature of the data we seek (since the script could not be rerun to correct any given snapshot after the corresponding day had passed).

Filtering. The dataset is filtered to focus on voting behavior on the "*average* review", eliminating all (day, review) pairs where a review received more than 10 positive or negative votes between consecutive snapshots (except for the analysis described in Section6.3.1 and Section6.4.1). This is to prevent such rare, very popular reviews from dominating and overshadowing the overall pattern of voting on reviews, since most reviews — even for the subset of popular products — receive at most a few votes (if any) between consecutive snapshots (*i.e.*, within a single day interval). Note that this means that the results are likely *not indicative* of voting behavior on the extremely popular reviews which might have interestingly different voting patterns. The focus here is on the remaining datapoints (after filtering ¿90% of datapoints and ¿60% of votes are retained) which represent the

more typical review on Amazon (for this dataset the average number of votes in a day is less than 2 even if we exclude days with no votes). Let's note, however, that the trends observed in the plots in Section6.4 and Section6.3 can still be seen when this data is not filtered out, albeit with more noise.

6.1.1 Rating Accumulation Process

Let's now discuss how reviews and ratings accumulate as a function of time elapsed following a product's release. Figure 6.1 shows that the rate at which new reviews appear remains roughly constant for almost the first 4 months of the 5-month data collection period, and remains non-negligible, although clearly lower, even in the last month. The plot does not start at zero reviews because the lists of products on "Hot New Releases" can also contain already released products, so that some very early reviews may already be present at $t = 0$. Also, since only the first 50 reviews are scraped for each product, any new reviews that appear below the 50th rank are ignored (unless and until they place in the top 50) — so this plot is essentially a lower bound on the actual total number of reviews[4].

Similarly, new votes appear throughout the collection interval (Figure 6.1). Again, the rate slows down over time but remains substantially larger than zero. The kinks in this plot arise from *cross-listing* — a jump occurs when Amazon cross-lists a review (along with its votes) from one product (not in the dataset) to another similar one (that is tracked) on that day. Figure 6.1 suggests that observed the voting patterns are not particularly influenced by the "age" of the reviews, in the sense that most of the reviewing and voting does not occur only very early in

[4]This is also the reason why we can have more than 29750 (595 products times 50) reviews in total, since all reviews that have ever been in the top 50 are included into the total, even if they are not seen in all snapshots.

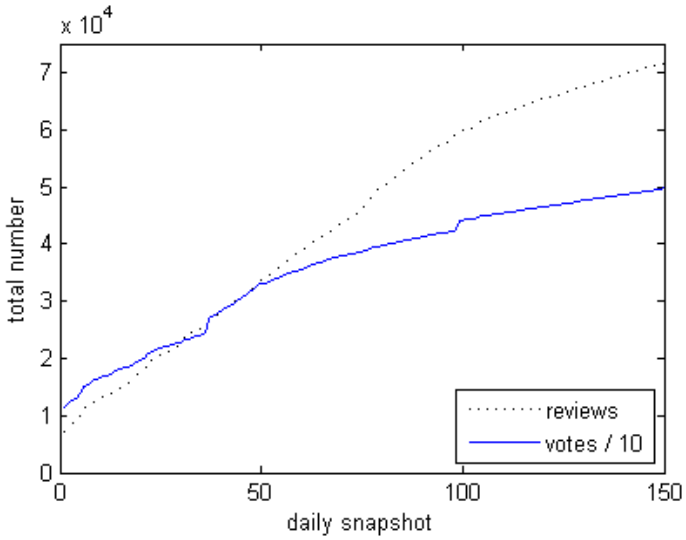


Figure 6.1: Total number of reviews and votes (y-axis) over time (x-axis in days).

the data set[5].

6.1.2 Converged Rankings

Approximately 4 months after the end of the data collection period, the full ranking of all reviews is collected for all products in the dataset (*i.e.*, not restricting it to the top 50 reviews as for the daily snapshots). The rank of a review in this final snapshot is reffered to as its *final true rank*, accounting for ties[6].

Comparing these final true ranks to the observed ranks towards the end of the data collection period shows that the top 30 positions of the rankings are largely stable by this time. In particular, Figure 6.2(b) shows that during the last 30

[5]A natural concern could be that voting stops shortly after product's introduction and this results in skewed averages when using the whole length of the data.

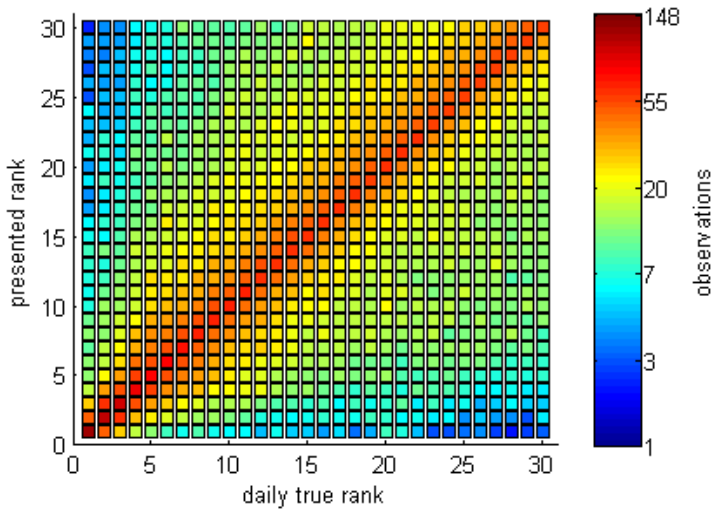
[6]Reviews with the same number of helpful and unhelpful votes are considered tied; these largely occur between reviews with 0 or 1 total votes, of which there are many in the dataset. Note that if such reviews were not considered tied at the same rank, then rankings would not stabilize since Amazon appears to continue to reorder such reviews with the same (low) number of positive and negative votes.

days of the data collection period, most reviews are already consistently close to the ranking positions observed 4 months later (after removing reviews that were published after the data collection period ended). Therefore we can conclude that the ranking process converges and that the relative ordering of reviews stabilizes.

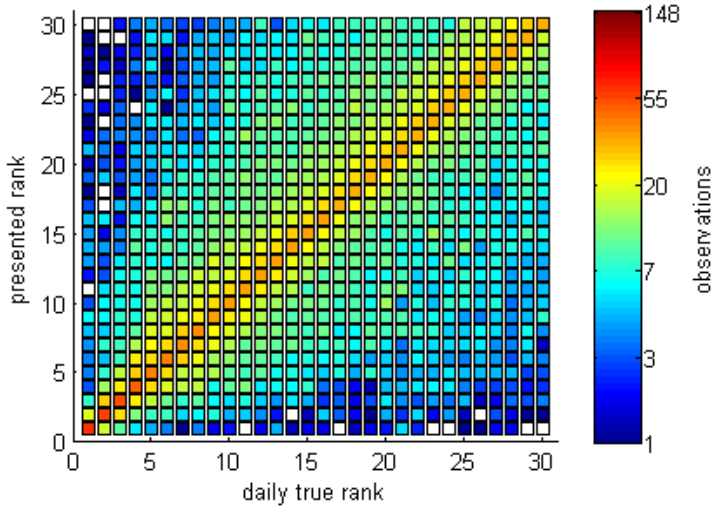
How Amazon computes a ranking in response to the helpfulness votes was empirically investigated to sanity-check that this convergence is not a pathology of the (secret) ranking algorithm that Amazon uses (e.g., Amazon may just decide to fix the ordering after a few months). In particular, a smoothed version of the ratio of "yes" votes to total votes as the ranking criterion (specifically, ranking by "yes" votes divided by total number of votes plus one) is considered. This simple ranking criterion achieves a Kendall-tau rank correlation coefficient (with ties) of 0.84 (using only reviews with at least 10 total votes and products with at least 2 reviews). While the correlation is not perfect (Amazon may use factors beyond user ratings, exploration strategies, and possibly more complex functions of the ratings themselves), we can conclude that the presented rankings correlate strongly with the observed helpfulness votes.

The final true ranks of the top 30 reviews[7] will be used as an axis for some of the plots in the analysis. Anecdotally, I believe that these top 30 reviews are reasonably ordered by review quality, and we can note that Amazon has a large commercial incentive to provide good rankings. Furthermore, only the relative ordering of the top 30 results is considered, and the top of the ranking has seen substantial attention from the users. But even if the final rankings do not reflect a perfect ordering by review quality, the voting patterns identified below reveal strong dependencies that illuminate how users cast votes.

[7]On the figures only the top 30 ranks out of 50 collected positions are used due to data sparsity.



(a) All



(b) Late

Figure 6.2: Number of times (color in log scale, with red being high) a review with particular daily true rank (x-axis) was presented at a particular rank (y-axis). Left plot (*All*) counts over all data, right plot (*Late*) only over the last 30 days.

6.2 Do User Votes Reveal the Absolute Quality of a Review?

When users are asked to rate content, the final goal is to reach some assessment of the quality of this content. Throughout the following sections, users' voting behavior when answering the question "Was this review helpful to you?", and how these votes relate to quality is empirically investigated. Let's start by considering a natural and simple model of voting behavior, which was assumed in prior work [129, 26].

Absolute Voting Model. The most natural model of how people vote is that they simply answer the question "Was this review helpful to you?" in an independent and objective way. Formally, this can be modeled as each different user u having a certain probability p_r^u of clicking "yes" for any particular review r. Over any distribution of users, the observed votes under this model follow a Bernoulli distribution, where $p_r = E[p_r^u]$ is the probability of observing a "yes" vote on review r, where the expectation is taken over the distribution of users:

$$P(yes|p_r) = p_r. \tag{6.1}$$

Under this model, p_r directly reflects the expected quality of review r in an absolute manner, since p_r becomes synonymous with quality. Furthermore, estimating p_r for each review r can simply be done by using the observed fraction of "yes" and "no" votes, which is the maximum likelihood estimator:

$$p_r = \frac{\text{number of yes votes on } r}{\text{total number of votes on } r}. \tag{6.2}$$

Let's call this model of voting behavior the *Absolute Voting Model (AVM)*. Note that this model, as well as the other models in this section, only describe the

polarity of a vote, but not the decision of *whether* to cast a vote at all or not – this participation decision is studied in Section6.4.

How accurate is this model of voting polarity and is it supported by the empirical data? Choosing an appropriate test to investigate the merit of the Absolute Voting Model is somewhat subtle, and a number of obvious tests turn out to be flawed. For example, a natural test would be splitting the data into two cases: one includes all the instances where a review is presented above its final rank and the other the rest. Then we could use e.g. a paired Student t-test with null hypothesis that votes in both cases originate from the same distribution. If voting is based purely on a review's inherent characteristics alone — as in the AVM model — then being displayed above or below one's 'correct' rank (as given, e.g., by the final converged rankings) should not change the vote's polarity. However, this test introduces a bias: we only get a paired sample when we observe instances in both bins. For example, such a test would exclude all samples where currently over-ranked reviews get even more positive votes (and thus never become under-ranked), while including over-ranked reviews that do obtain negative votes (which would falsely support a hypothesis where users vote to fix the ranking).

In order to avoid flawed tests where the split of samples into cases to study voting patterns depends on the votes themselves, the following likelihood-ratio test that uses the Absolute Voting Model as the null hypothesis is used instead. The test will identify whether presentation effects — even in their simplest form — can significantly better explain users' voting decisions.

Extended Absolute Voting Model. Consider a simple extension of the AVM, where the probability with which a user votes "yes" or "no" on a particular review r not only depends on its inherent quality (as in the AVM), but also on

the position where this review was presented in the ranked list. Specifically, the probability of a "yes" vote on a review with inherent quality parameter q_r and presented rank σ^{pres} is given by the following logistic model:

$$P(yes|q_r, \beta, \sigma^{pres}) = logit^{-1}(q_r + \beta\, \sigma^{pres}). \tag{6.3}$$

Note that the Extended AVM model has one free parameter q_r for each review and a globally shared parameter β that models the influence of the presented rank σ^{pres} (which is observed). These parameters are estimated using maximum likelihood. This maximum-likelihood objective is convex, which means that the associated optimization problem can be solve globally optimally.

Note that the AVM is a special case of the Extended AVM model with $\beta = 0$. The Extended AVM with $\beta = 0$ merely parameterizes the AVM in terms of a quality parameter q_r, which is bijectively linked to the p_r parameter of the AVM through $p_r = logit^{-1}(q_r)$. Furthermore, maximum likelihood estimation in the Extended AVM with $\beta = 0$ leads to exactly the same estimate of p_r as in Equation (6.2).

Testing the Influence of Presentation. The nested structure of the AVM and the Extended AVM enables us to perform a likelihood ratio test that has the AVM model as its null hypothesis. The test compares the likelihood of the observed data under the null hypothesis (i.e. $\beta = 0$) with the likelihood of the unrestricted model. If the improvement in likelihood is sufficiently large (*i.e.*, larger than one would expect from simply having more parameters to optimize over), then the likelihood ratio test rejects the null hypothesis.

The log-likelihood of the AVM ($\beta = 0$) model is -59267.78. The log-likelihood of the Extended AVM model is -57861.41. Both models are estimated from 136009 datapoints with 233512 total votes. The critical value according to the χ^2 statistics

for one degree-of-freedom at the 95% confidence level is 3.84, which is much smaller than the observed difference in log-likelihoods of 1406.37. We can therefore reject the AVM model in favor of the Extended AVM model with high confidence ($p < 0.001$). Clearly, users do not give independent assessments of the review quality. Simply presenting the review in a different position has a substantial effect on their voting behavior, which means that we cannot take the observed ratios of "yes" and "no" votes as an absolute measure of quality as assumed in the AVM model.

The rest of the chapter will explore improved models for how users make voting decisions. As a first step toward such models, let's look the the estimated β of the Extended AVM model, which is $\beta = 0.0722$. Somewhat surprisingly, this value is positive, indicating that users are more likely to vote "yes" (as opposed to "no") if the review is presented *lower down in the ranking.* This stands in stark contrast to other settings where endorsements are used to rank items, especially Web Search where clicks are used as endorsements. Positive endorsements in Web Search follow a strong rich-get-richer pattern, where a result gets more positive endorsements (*i.e.*, clicks) the higher it is presented in the ranking [34]. While the types of endorsement (e.g., explicit positive and negative votes vs. positive clicks only) and the timing of the endorsement (e.g., before or after consuming the content) are different in the two settings, it is nevertheless evident that the two settings require very different machine learning methods for aggregating endorsements into an optimal ranking.

6.3 How does Context Relate to Voting Polarity?

The previous experiment showed that the rank at which a review is presented is correlated with the polarity of the vote users cast. Is there a plausible model of the user's decision process that could lead to this bias in user behavior? One can conjecture a large number of factors that causally influence a user's decision, ranging from position itself having causal effect, all the way to biases involving the time of day or week[8]. It is also conceivable that the history of votes so far also changes the user's perception of a review itself, as in herding phenomena [68]. Most promising, however, we can conjecture that the *context* — the quality of surrounding reviews — might lead a user to change her opinion about the helpfulness of a given review. In particular, the following section explores whether voting polarity shows any dependence on the degree to which a review is "misordered" relative to its context.

6.3.1 Statistical Analysis

A statistical analysis was conducted to see if voting polarity depended on misorderings in the ranking. To provide the tightest amount of control against confounding factors, the focus of the statistical analysis is the voting behavior in the top three positions of the ranking — a more global analysis follows in the subsequent subsections. Since Amazon by default presents three reviews, the choice of three is natural. Let r_1, r_2 and r_3 be the best three reviews for a given product as determined by their *final ranks* (see Section6.1.2). Table 6.1 compares the polarity of the votes on r_1 and r_2 under two different conditions, namely when the reviews

[8]For example, the user population visiting the site primarily during weekends might be in a better overall mood and thus vote more positively.

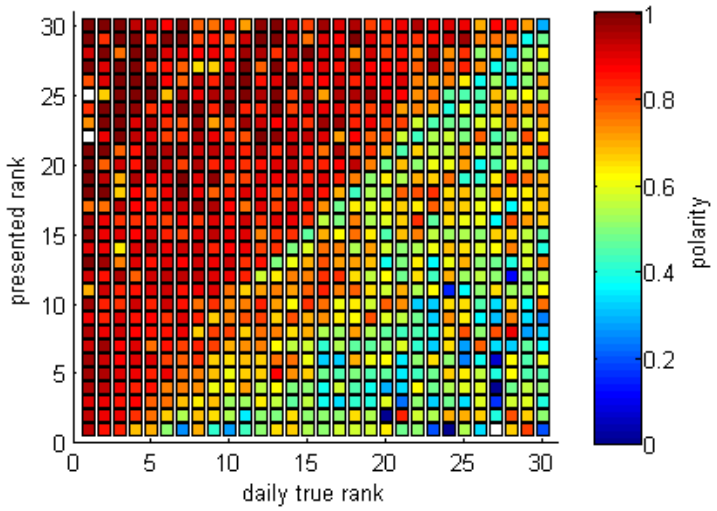
were presented in the order $r_1 - r_2 - r_3$ vs. the order $r_2 - r_1 - r_3$. The average polarity is computed for each product and each condition separately (using only votes from those snapshots where the top three reviews appear in the desired locations) and the table shows macro averages over products. Note that the set of three results is the same under both rankings, and the key difference is the switch in ordering between r_1 and r_2.

Table 6.1 shows that the polarity of votes on r_1 is more positive in the swapped condition, while the polarity of votes on r_2 is more negative. Both differences are statistically significant according to a two-tailed paired Student t-test (a pair for each product) with $p < 0.05$, as shown in the last row of the table. This is in agreement with a model of user behavior where users cast their vote to "fix" the perceived misordering in the ranking by upvoting a review that is ranked too low, and downvoting a review that is ranked too high in relation to its context. Note that this is opposite to the biases identified for clicking behavior in web search that were observed in an analogous experiment [34].

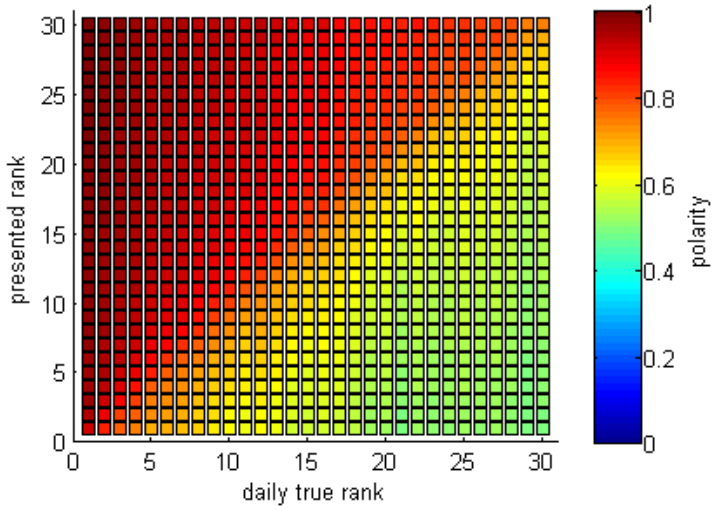
Ordering	review r_1	review r_2
$r_1 - r_2 - r_3$	0.912	0.881
$r_2 - r_1 - r_3$	0.946	0.811
p-value	0.0395	0.0183

Table 6.1: Voting polarity when the same reviews are presented in different orders.

After this microscopic study of the first three positions (where a hypothesis test showed a significant depencence between review ordering and voting polarity), let's now perform a macroscopic exploratory analysis of whether misorderings also correlate with observed changes in polarity over the whole ranking. To do this, let's consider two different measures of how misordered a ranking is, which are called "global context" and "local context".



(a) global context, data



(b) global context, model

Figure 6.3: Vote polarity: daily true rank (x-axis), context (y-axis, presented rank) and average vote polarity (color, red means higher ratio of positive votes).

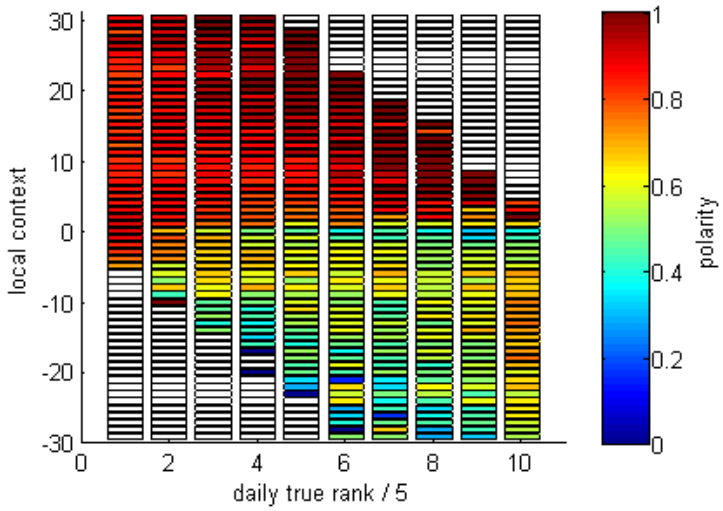


Figure 6.4: Vote polarity in data for local context: daily true rank (x-axis, with bin widths of 5), context (y-axis, positive values meaning superior) and average vote polarity (color, red means higher ratio of positive votes).

6.3.2 Exploratory Analysis: Global Context

The first measure of how misranked a particular review is relates the current position of a review to where it "should be" in a globally sorted ranking. To make this more precise, let's define a quantity called the *daily true rank* of a review as follows.

Since new reviews for any given product might appear during the course of the data collection, we need a way to compute the "true rank" of a review — the "correct" rank that review "ought" to be displayed at — among the reviews present on any particular day[9]. To this effect, reviews which were not yet published by that day are disregarded, and the already existing reviews are sorted by their final true rank. This results in the daily true rank of a review. A review is called overranked if it is presented above its daily true rank, and underranked if it is

[9]Note that this can be different from the set of all reviews present on the last day

presented below.

Figure 6.3(a) displays the average polarity of the votes (color from negative in blue to positive in red) as a function of daily true rank on the x-axis and presented rank on the y-axis. The figure shows that the more a review is presented above the rank it deserves, the more negative the polarity of the votes is. On the other hand, when a review is presented too low relative to its deserved rank, the polarity of the votes is more positive. These observations hold across a wide range of presented ranks and daily true ranks. This can be interpreted as users upvoting reviews that are rated 'too low', and downvoting reviews that are rated 'too high'. Note that the notion of converged ranks is carefully defined using rankings collected 4 months after the end of the daily snapshot collection to mitigate self-fulfilling prophecies in terms of using the same votes to understand behavior as to determine what is too low or high, as described in Section6.1.2. Furthermore, a possible bias of it correlating with the likelihood of being over-ranked (e.g., the best review can never be over-ranked) and skewing the averages (*i.e.*, positive votes from good reviews contributing mainly to under-ranked polarity average) is avoided by fixing the daily true rank (each one has its own bin on x-axis).

Voting Polarity over Time. Because the data was collected over a long timespan (a few months) there might be differences in the voting behavior between early and late periods. To explore this, analogues of Figure 6.3(a) are separately ploted using only data from the first and the last 30 days of the collection period, respectively. The resulting plots are shown in Figures 6.5(a) and 6.5(b). While the plots are more noisy due to the smaller datasets, the observed patterns are remarkably stable over time.

6.3.3 Exploratory Analysis: Local Context

Using global context to model a user's perception of how misordered the ranking is assumes that the user has a global understanding of the ranking. This clearly can only be true in an approximate sense. More likely, a user bases her voting decision on a more local view of misordering. Since Amazon displays reviews as a ranked list, a user will see any particular review in the context of the other reviews surrounding it at that time. The reviews presented immediately above and below a particular review are reffered to as the *local context* of the review. For the data analysis, the local context is defined as the 3 reviews appearing immediately above and below a review at any given time unless otherwise specified[10].

Local superiority and inferiority. The following measure is used to capture how the quality of a review relates to the 6 reviews in its local context: the daily true rank of the review under consideration is subtracted from the average of the daily true ranks of these 6 reviews. The review is *locally superior* (of higher quality compared to its surrounding reviews) when this difference is positive, and *locally inferior* when the difference is negative.

Figure 6.4 demonstrates that local context does indeed correlate with the polarity of votes received by a review. The x-axis ranges over values of the daily true rank (each bin is 5 ranks wide), and the y-axis measures the relative quality, where negative values correspond to local inferiority and positive to local superi-

[10]Presentation-induced anomalies such as those created by page breaks are ignored, *i.e.*, the fact that reviews near the top or bottom of a page are not visually surrounded by their local context on both side in the same pageview. Also, I verified that the specific choice of the number 3, as well as considering reviews appearing only above or below a review does not alter the results (since it is conceivable that reviews appearing immediately before a review would influence the vote cast on that review more than those that are read after it); the pattern of the fraction of positive votes received as a function of local context appears to be quite robust to the specifics of exactly how the local context is defined.

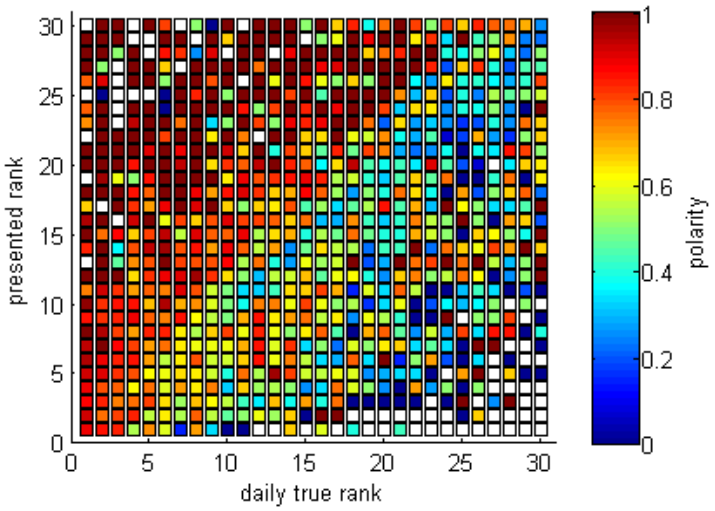
ority. The color of a point (x, y) is the polarity of votes received by a review when it has daily true rank x and local context y. If local context did not correlate with voting patters, there should be no gradient in color along the y-axis. However, Figure 6.4 displays a noticeable increase in the fraction of positive votes when a review is locally superior, and vice versa when the review is locally inferior.

Note that the effect of global and local context on voting polarity closely resemble each other: a review receives more positive votes when it is under-ranked (a global measure) as well as when it is locally superior, and fewer positive votes when it is over-ranked as well as when it is locally inferior. That is, the interpretation that users vote to correct misorderings in the ranking is not sensitive to this particular measure for misordering, and in fact leads to similar qualitative observations for two different measures of misordered rankings.

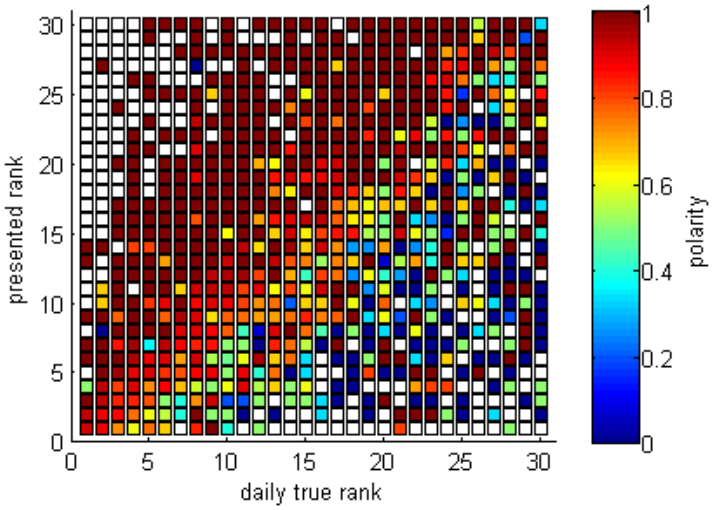
6.3.4 A Model of Voting Polarity

This section abstracts the observed voting polarity patterns into a general model. The reason for this is twofold. First, such models of voting polarity are needed for designing content-ranking algorithms that make optimal use of the votes they elicit. Second, a formal model can be tested and verified also in other content-rating settings.

Voting polarity is modeled using a traditional form of logistic regression (with two variables capturing insights gained in the previous sections). Each review r has an inherent quality q_r that is unknown. In addition to inherent quality, the model considers the context δ^{ctxt} of a review at the time a user casts a vote. A positive δ^{ctxt} means that review r is better than its context, a negative δ^{ctxt} means



(a) Early



(b) Late

Figure 6.5: Vote polarity: daily true rank (x-axis), presented rank (y-axis) and vote polarity (color, red being positive). Only using the first 30 days (left) and last 30 days (right) of data.

that it is worse. The strength with which δ^{ctxt} influences polarity is captured by the parameter β:

$$P(yes|q_r, \beta, \delta^{ctxt}) = logit^{-1}(q_r + \beta\, \delta^{ctxt}). \tag{6.4}$$

Both global context (*i.e.*, the difference between presented rank and daily true rank) and local context as defined above can be used in approximations of context. However, it is unreasonable to assume a linear relationship between δ^{ctxt} and these rank-based measures. Instead, the following transfer function is used and de-emphasizes the impact of large rank differences. Let δ^{ctxt}_{rank} be either the local or the global context in terms of rank, then

$$\delta^{ctxt} = sign\left(\delta^{ctxt}_{rank}\right) log\left(1 + \left|\delta^{ctxt}_{rank}\right|\right). \tag{6.5}$$

Using this transfer function and global context as a proxy for δ^{ctxt}, the fitted model has a log-likelihood of -57051 with parameter $\beta = 0.415$ (under-ranked reviews have more positive δ^{ctxt} which in turn means more positive polarity due to a positive β). Figure 6.3(b) plots the fitted model, which can be compared directly to Figure 6.3(a). In particular, Figure 6.3(b) is produced by replaying the observed rankings that produced Figure 6.3(a) and then, for each cell, averaging the predicted probabilities of the model instead of the observed vote polarities. Overall, the model captures the key trends in the data, including a decrease in voting polarity with rank on the diagonal, and the increase in voting polarity for reviews that are ranked too low.

6.4 How does Context Relate to Participation?

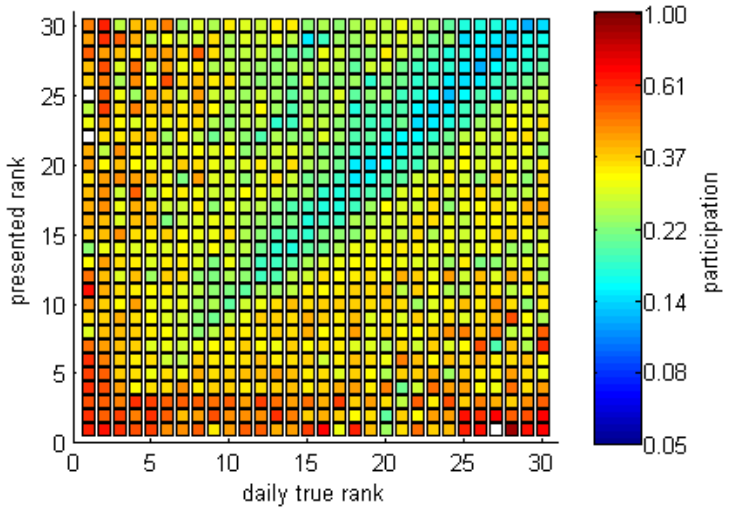
So far explored user voting behavior was only explored in terms of *polarity* (*i.e.*, how the users cast "yes" vs. "no" votes). A second, equally important dimension

is participation — *when* does a review receive votes? A first guess would be that the number of votes a review receives depends largely on its presented rank, corresponding to an attention bias whereby more users read, and therefore vote on, reviews that are displayed at higher rather than lower ranks. However, this is not the entire story — participation also depends on context, as we show below.

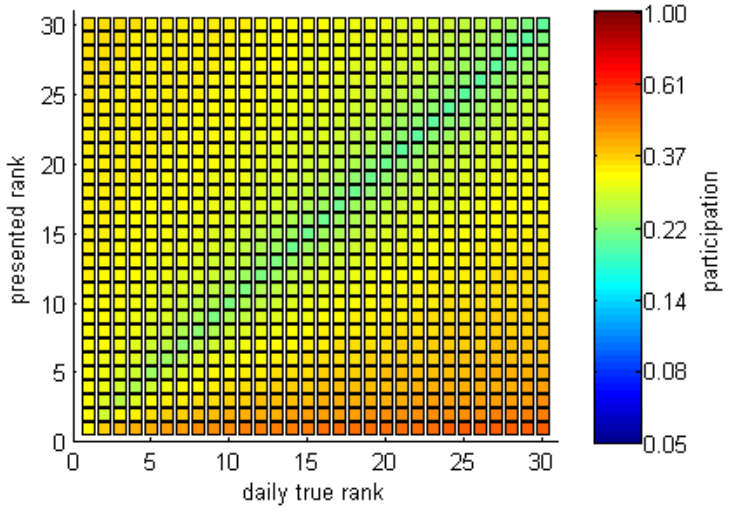
Ideally, participation would be measured as the ratio of the total number (positive plus negative) of votes on a review to the number of views it receives. However, since the data does not include information about pageviews, we have to assume that there is a constant number of pageviews (for each position) in each period between snapshots (note that the actual number of such pageviews does not matter since we are only interested in relative comparisons). An additional issue is sparsity of data — most reviews receive no or one vote on most days. To deal with these issues, participation is measured using the following statistic. Consider a particular bin (x, y) defined by some value of the x and y variables (daily true rank and local/global context). As a measure of participation, we can use the ratio of the number of intervals (between two consecutive snapshots) where at least one new vote was cast on reviews in this bin to the total number of observed intervals in the bin.

6.4.1 Statistical Analysis

To investigate whether there is a dependence of participation on context similar to polarity, let us start by considering the same experiment setup as in Section6.3.1. Table 6.2 shows participation at rank 1 and rank 2 conditioned on the ordering of the top two results. When r_1 is correctly ordered before r_2, there is a strong decay in participation, as one would expect from an attention bias (see [34] for



(a) global context, data



(b) global context, model

Figure 6.6: Participation: daily true rank (x-axis), context (y-axis, presented rank) and average participation (color, log scale, red being high).

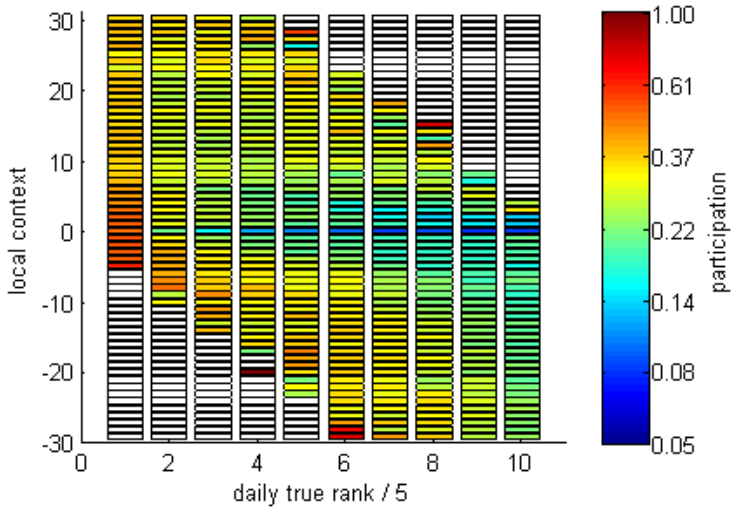


Figure 6.7: Participation in data for local context: daily true rank (x-axis, using bin widths of 5), context (y-axis, positive values meaning superior) and average participation (color, log scale, red being high).

similar attention biases in web search). Once r_1 gets misordered into position 2, however, voting participation on position 2 significantly (two-tailed paired Student t-test, $p < 0.05$) increases compared to the correct ordering. It appears that users are motivated to participate once they see that the ranking is misordered. Voting participation on rank 1 does not change significantly. A plausible explanation is that users typically do not go "back" up the ranking to vote once they have realized that there was a misordering.

	rank 1	rank 2
$r_1 - r_2 - r_3$	4.34	2.33
$r_2 - r_1 - r_3$	3.71	4.19
p-value	0.395	0.004

Table 6.2: Participation when the same reviews are presented in different orders.

Similar to the argument for polarity, let's now explore how far this microscopic finding about participation extends more macroscopically.

6.4.2 Exploratory Analysis: Context

First, let's investigate how participation varies as a function of global context. Figure 6.6(a) plots participation (from low participation in blue to high participation in red) as a function of presented rank and daily true rank. As expected, there is an attention bias. Amongst reviews that are correctly ranked (*i.e.*, presented at their daily true rank), reviews with higher ranks are voted on more often (note that the color scale is logarithmic). This can be observed from the diagonal elements in Figure 6.6(a). Furthermore, note that there is a special attention bias for the top three presented ranks, which receive much larger participation. This is because Amazon presents the top 3 reviews on the product page, while an additional click is required to display more of the ranking.

Beyond attention bias, Figure 6.6(a) shows that context affects participation as well — reviews receive more votes when they are incorrectly ranked. Comparing the off-diagonal elements in Figure 6.6(a) with the diagonal ones, we see that reviews get voted on more often when they are not in their correct position. The upper triangular portion of Figure 6.6(a) demonstrates the same effect that was already observed in previous subsection's experiment — a good review that is ranked too low receives more votes. Unlike in the statistical analysis of the top three positions, however, the lower triangular portion of Figure 6.6(a) shows that participation at lower presented ranks also increases when a review is presented too high. A plausible explanation is that for reviews at lower presented ranks, the user has already formed a reliable expectation about the review quality of the next review, upon which the user can judge misordering.

Similar trends also emerge in the analogous Figure 6.7 for local context. Reviews that are locally superior or inferior (non-zero bins on y-axis) get voted on

more often.

Voting Participation over Time. Similarly to polarity (Section6.3.2), temporal consistency is observed in voting participation. Both Figure 6.8(a) and Figure 6.8(b) are analogous to Figure 6.6(a), but are restricted to the first 30 days and the last 30 days of the data collection period, respectively. Both plots show the same general v-shaped pattern, indicating that participation follows a stable pattern over time.

6.4.3 A Model of Participation

The following proposes a model of participation that is analogous to the one derived in Section6.3.4 for polarity. For each position p, we model the "normal" amount of attention a review at this rank gets using the parameter z_p. Variable δ^{ctxt} is the context of review r as defined for polarity, and we use the same transfer function from Equation 6.5 to connect δ^{ctxt} to the rank-based measures of global and local context. However, since participation is symmetric in δ^{ctxt}, we use its absolute value. The parameter α models the influence of context:

$$P(vote|z_p, \alpha, \delta^{ctxt}) = logit^{-1}\left(z_p + \alpha \left|\delta^{ctxt}\right|\right). \tag{6.6}$$

This participation model is fit directly to the observed participation frequencies from Figure 6.3(a) using maximum likelihood, where each cell receives the same weight in the maximum likelihood estimate. This is done because the number of observations in the first few presented ranks otherwise dominates the likelihood and biases the parameters towards a fit to only those cells. The parameters z_p are smoothed to lie on a curve $z_p = \gamma_0 + \gamma_1/p^{\gamma_2}$, where γ_0, γ_1, and γ_2 are fitted parameters. This step is added to produce smoother estimates of the z_p despite

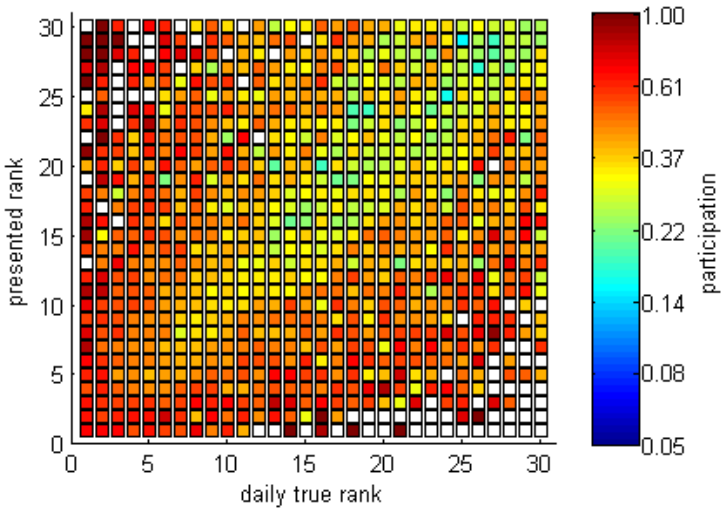
data sparsity at low ranks, but it is not essential for capturing the general trends.

The resulting model is plotted in Figure 6.6(b), and it is analogous to the plot of the observed data in Figure 6.6(a). The parameter α of the fitted model is 0.228. This means that the more misranked a review is according to $|\delta^{ctxt}|$, the higher the participation. While the model underestimates participation at the top presented ranks (which is due to the equal weighting of the cells during maximum likelihood estimation), the plot overall resembles the patterns in the observed data.

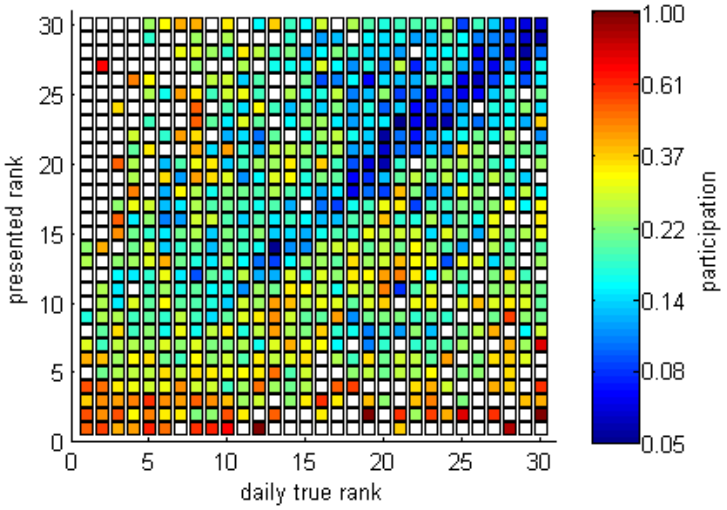
6.5 Discussion of the Analysis

Previous sections analyzed user voting patterns on Amazon reviews, over time and as a function of the context in which the review was rated by the user. Using a dataset of daily snapshots of reviews and their ratings, we can observe that voting *polarity* (*i.e.*, whether to assign a positive or negative helpfulness vote to a review) as well as *participation* (*i.e.*, whether to vote at all) depend not only on the inherent quality of a review, but also on the context in which it is presented at the time of voting. In particular, a hypothesis test provided evidence that the observed connection between context and voting behavior cannot be captured by an absolute voting model (Section6.2), where users make absolute and independent judgments about helpfulness.

As an alternative to the absolute voting model, models of voting incorporating context in addition to a review's inherent quality were described, and they provide a much closer fit to the observed data. Notably, voting polarity becomes more positive/negative, if a review is better/worse than its context. Furthermore, voting participation generally increases if a review is misranked. These patterns are sub-



(a) Early



(b) Late

Figure 6.8: Participation: daily true rank (x-axis), presented rank (y-axis) and participation on log scale (color, red being high). Only using the first or last 30 days of data.

stantially different from other setting where endorsements are used to rank items, most prominently Web Search, which means that methods for learning rankings in one setting cannot be naively transferred to another.

6.5.1 Limitations

As the previous sections pointed out, the study design and analysis have some limitations which are further discussed below. These provide several interesting directions for further work.

(i) *Observational Data.* The analysis, which is based on observational rather than experimental data, comes with the limitations typically associated with deriving models from observational data. While non-uniformity in the presented ranks provided natural experiments which allowed us to investigate correlations, only experiments with controlled interventions can ultimately validate the causal reliability of the conclusions and models.

(ii) *Snapshot Granularity.* Recording one snapshot per day is too infrequent to capture all the voting events that occur on Amazon. In particular, there are reviews that receive more than a hundred votes in a single day, and rankings can also change very fast. A separate dataset containing snapshots at intervals of about one hour was collected over the course of one day; even here there were products where review rankings changed between snapshots. However, the observations on this dataset showed that rankings do not change drastically (e.g., often only reviews with the same number of votes swapping places), so that using daily snapshots

does provide an acceptable approximation for studying voting patterns on typical reviews on Amazon. Nevertheless, studying a more fine-grained dataset can potentially lead to new insights on very frequently read reviews.

(iii) *No Independent Assessments of True Quality.* Ideally, one would have liked to use independent assessments to determine the true quality (*i.e.*, true rank) of each review where this quantity is used in the analysis. However, this was infeasible since even with access to grading resources, it would have been unclear how to judge which reviews truly are helpful to the (unknown) user population. A possible concern with an analysis that uses final ranks as a proxy for true quality is that the data used to analyze voting behavior also contributes to defining the true qualities (namely the final ranks): for example, a review with a given final rank k might have been ranked lower, *i.e.*, been underranked, at some earlier point in the dataset, and then received an above-average share (relative to the entire set of votes that contribute to the final rank) of positive votes after this time to climb up to its final rank. This can lead to a self-fulfilling prophecy where 'underranked' reviews receive an above-average share of positive votes, and 'overranked' reviews a below-average share. The following points help mitigate concerns regarding such self-fulfilling prophecies: (a) The final true ranks are computed based on a snapshot recorded 4 months after the last sample in the data used for the analysis of voting patterns. That is, the final ranks also include a large number of votes cast over a long interval that does not overlap with the interval corresponding to the votes in the empirical analysis (see Section6.1.2); (b) The statistical analysis in Section6.2 does not make use of final ranks at all, but still shows that voting is not based purely on inherent review quality.

(iv) *Aggregate vs. Individual Voting Models.* The models do not claim to describe the actions of any individual user, but merely the aggregate behavior of a user population. For example, we cannot ask whether each individual user modifies polarity based on rank (even though that is a plausible conjecture), but only observe that the population of users displays this aggregate behavior. The same aggregate behavior, however, could also be explained by other factors, such as heterogeneity in user populations (different types of users may have different baseline polarities and may explore rankings to different depths). Investigating the effect of such alternate hypotheses is an interesting open direction.

(v) *Macroscopic Participation and Polarity.* The model for participation predicts that a review displayed at its correct rank should receive fewer votes. This means once rankings converge to the right ordering of reviews, participation should globally decrease. This is indeed observed in the data. Figure 6.2(b) shows that rankings stabilize after a few months, and Figure 6.1 shows that voting also decreases at this time. Regarding voting polarity, Figure 6.9 shows that the average vote polarity becomes more positive with time. This is to be expected. Since converged rankings have 'good' reviews at the top (with high attention bias), these reviews have to maintain a high fraction of yes votes (or regain that ratio once their rank has dropped). Trends in participation and polarity could also have other causes, however. For example, participation could be explained by a product becoming outdated and consequently fewer users reading and voting on reviews. Similarly, an alternate explanation for polarity is that users continue to give positive votes to reviews they like even when rankings are accurate, but do not downvote reviews unless necessary to correct the ranking.

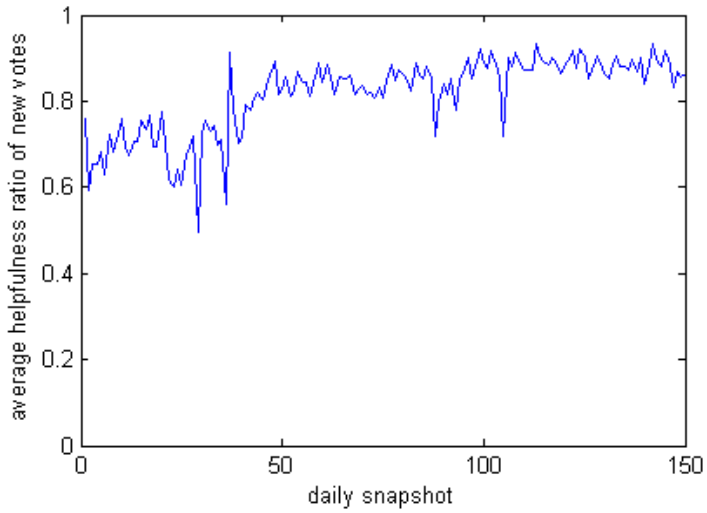


Figure 6.9: Average ratio of positive to total votes (y-axis) over time (x-axis in days).

(vi) *Perception versus Action.* A final intriguing question that is not addressed is the mechanism by which context affects voting: does a review's context cause a user to actually change her perception of its helpfulness, or merely her action? One possibility is that a user may truly value a review less if it is presented next to an even better review. A different possibility is that each user might have some absolute quality capturing each review's helpfulness, and vote in order to 'correct' the current vote ratio (of current "yes" to current total votes) towards her opinion of the review's quality. Preliminary explorations indicate that both effects (modified perception and modified action) might be supported by the data. An experimental study, or analysis of a more detailed dataset, could lead to clearer conclusions on this question.

CHAPTER 7

CONTENT RANKING ALGORITHMS FOR UGC

Not all of user-generated content is of equal quality, which poses a *quality estimation and ranking* problem in order to present the best content to viewers. Since the massive scale of UGC makes it infeasible for most websites to manually rate their content, sites often crowdsource this evaluation problem to their audience by allowing viewers to vote on quality. An example is Amazon asking users "Was this review helpful to you? Yes/No", and other websites employ similar feedback mechanisms.

A straighforward estimator of quality — and one that appears most commonly used among these websites — is the ratio of positive to total votes received (or possibly a small modification thereof). This estimator, while simple, does have a clean theoretical basis in a world where each vote is a Bernoulli coin toss with success probability q, where $q \in [0, 1]$ is some measure of the contribution's underlying quality. Under this model, the ratio of positive to total received votes is an unbiased estimate of q. Previous sections suggest, however, that user voting behavior in real websites is more complicated than described by this simple 'absolute voting' model. In particular, users do not make absolute, independent judgements of contributions' qualities; rather, a model of voting behavior that incorporates the *context* in which a contribution was viewed yields a much better fit to observed voting data. What are the implications of this refined model of voting behavior on estimating content quality and ranking content?

The following sections investigate two questions to this effect. The first question regards an *analysis* of current practice: how does the basic widely-used algorithm that ranks contributions according to the ratio of positive to total votes — the

maximum likelihood estimator for a absolute voting model — perform when its data instead comes from a more realistic contextual model of user voting behavior? In particular, one may be worried that this mismatch between the model that justifies the estimator and the actual data generating process could lead to poor ranking performance. Fortuitously, quite the contrary happens. The complex voting behavior displayed by real users turns out to actually be *more* informative than if users were voting according to the simpler and more natural Bernoulli model. Specifically, the basic ranking algorithm converges faster, and to a more accurate ranking, with voting behavior described by a contextual voting model than when votes are drawn according to the absolute voting model with Bernoulli votes (Section7.3).

The second question regards *design*: can one converge to the good ranking even faster by using a more sophisticated estimator designed specifically for such a contextual voting model? A new algorithm is proposed that employs an improved estimator that outperforms the conventional ranking algorithm, both in terms of accuracy and speed of convergence (Section7.4). Then, how much improvement in performance results from using the sophisticated estimator relative to the naive estimator for a range of parameter values in the contextual voting model is investigated in Section7.4.3. Finally, the robustness of this algorithm to the parameters of the data for which it is designed in explored in Section7.4.4.

7.1 Models of User Voting Behavior

In order to design and analyze methods for estimating quality estimation based on user feedback, it is necessary to have an accurate model of how users issue this feedback. In the following, we describe two possible models of how users vote

on content. The first is the Absolute Voting Model (AVM), which assumes that users make independent voting decisions for each piece of content and was already partially introduced in 6.2. The second is the Relative Voting Model (RVM) based on description in Section 6.3.4, which models both the polarity of the vote, as well as the user's decision to participate in voting, as dependent on the context in which the review was presented. The RVM model was shown to fit observed voting behavior on Amazon reviews significantly and substantially better than the conventional AVM [93].

7.1.1 Absolute Voting Model

A natural first model of voting behavior on user-generated content [129, 26] is a naive IID Bernoulli voting model, whereby viewers independently cast Bernoulli votes on each contribution. We call this the Absolute Voting Model (AVM). Under this model, the observed votes directly reveal a cardinal score describing the contribution's inherent 'quality', where the quality of a contribution is an unobserved abstract parameter that might depend on, for example, its actual textual content, its 'opinion' content (such as the star rating it gives to a product on Amazon), the identity of the author, social features, and so on. This model was already described and discussed in 6.2. Note that if votes are cast according to this model, the ratio of positive to total votes is the maximum likelihood estimator of the *true* quality p_r for each contribution. This estimator is unbiased.

Participation. The Bernoulli voting process only describes the *polarity* of the votes cast by each viewer, *conditional* on a vote being cast. But not all contributions might receive votes from each viewer; we therefore also need a model describing participation, *i.e.*, the number of votes that a contribution will receive.

Again, a simple model of participation is that the number of viewers, and therefore the number of votes cast, drops off with the rank p at which a contribution is displayed:

$$P(vote|z_p) = z_p. \tag{7.1}$$

Note that the probability of a vote depends only on the presented rank, but not on the quality of the content itself. To model the drop-off in attention, we use the following decay model, where $\gamma \geq 0$ is a parameter that controls the speed of attention decay:

$$z_p = p^{-\gamma}. \tag{7.2}$$

The drop-off is steep for the top few ranks, but there is still some non-negligible amount of voting even on the low ranks. This reflects observed voting behavior from Amazon (based on data from [93]) with best $\gamma = 0.8$. Note that Section 6.4.3 already touches on this subject, but here we even further simplify the attention bias model while still retaining a good fit to the observed data. Figure 7.1 shows users' participation (y-axis) in relation to presented rank (x-axis) and we can see that the chosen polynomial model has a good fit (and it gives qualitatively better fit than exponential decay used for e.g. web search [15]). The probability of participation on the first rank is rescaled to be equal to 1, because data [93] uses arbitrary temporal units (and thus the scale does not matter, it only changes the time interval between pageviews).

7.1.2 Relative Voting Model

An analysis of voting data from Amazon reviews [93] suggests that users, when asked the question "Was this review helpful to you? Yes/No", answer it with votes that cannot be explained by the Cardinal Voting Model described above. A

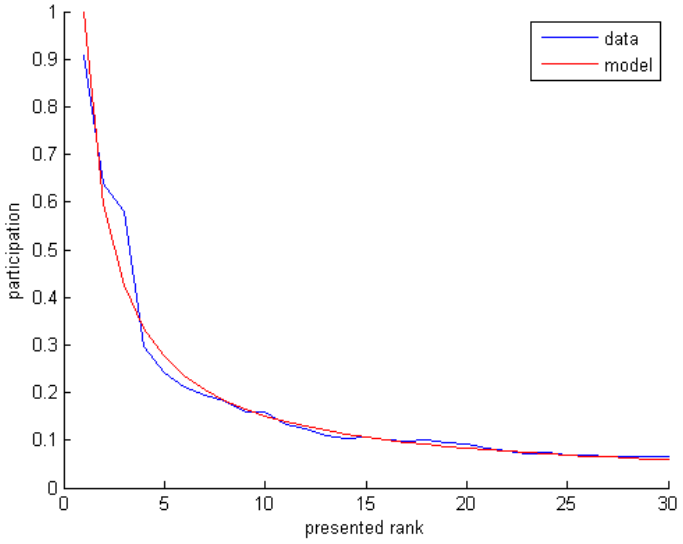


Figure 7.1: Attention bias for participation with presented ranks on x-axis and probability of participation on y-axis. Values obtained from real-world dataset are in blue and the model fit is in red color.

Relative Voting Model (RVM) [93], however, that incorporates the instantaneous *context* in which the contribution was viewed at the time of voting yields a much better fit to the observed voting patterns in the data.

The Relative Voting Model refines the Absolute Voting Model, recognizing that the voting behavior on a piece of content may be influenced by the quality of the other content items. In particular, the RVM introduces the notion of a *context* that captures how "misranked" a particular piece of content is. Section 6.3.4 already presents one such context-dependent model for voting polarity. RVM is almost identical to Eq. 6.4, except for the form of parameter β which here becomes easier to interpret, and is captured by the following probabilistic model. Let q_i be the true (unknown) quality of content i, then a positive (negative) value of context $\delta_{i,t}^{ctxt}$ indicates that it is better (worse) than its context at time t. True quality q_i and current context $\delta_{i,t}^{ctxt}$ are then combined using a binomial logit link function to

model the polarity of a vote:

$$P(yes|q_i, \beta, \delta_{i,t}^{ctxt}) = logit^{-1}((1-\beta)q_i + \beta\, \delta_{i,t}^{ctxt}). \tag{7.3}$$

Parameter $\beta \in [0, 1]$ is a tradeoff between quality and context-based votes. Note that for $\beta = 0$ the RVM is isomorphic to the AVM, and that maximum-likelihood estimation of the q_i results in a model that is identical to using the estimator in (6.2). The following simulation experiments use a local notion of context to define the value of $\delta_{i,t}^{ctxt}$ (see Section 6.3.3), using only one review above and below it, and with identity for the transfer function (instead of Eq. 6.5).

Participation. For the participation part, RVM uses the same form as was introduced in Section 6.4.3 with attention bias falloff using the same shape as AVM in the previous subsection.

7.2 Voting Simulation Setup

This section describes the simulation setup used to address the two questions we are primarily interested in—first, investigating the performance of the naive ratio-of-votes ranking algorithm when votes are cast according to the Relative, rather than the Absolute voting model, and second, comparing the performance of an estimation and ranking method designed for the RVM to that of the naive model, which is an optimal estimator for the AVM. Addressing either of these questions satisfactorily via evaluation on 'real data' is quite hard—an empirical study with existing voting data from any real website (which uses a particular ranking algorithm and corresponds to one specific voting pattern) makes comparisons infeasible, while an experimental study (using e.g. crowdsourcing) would face issues related to selection biases in the participant population, and concerns about the validity

of conclusions regarding 'real' user behavior on real websites. This leads to use of simulations, using votes that are drawn from models of real user behavior fitted to empirically observed data, as a means to addressing these questions.

7.2.1 Workflow

The simulations follow the high-level structure outlined in Algorithm 5, and are meant to mimic the voting and ranking process on a real user-generated content website. Each *run* of the experiment evaluates the performance of a given ranking algorithm on a set of products[1], each with reviews whose qualities are randomly drawn from some specified distribution, when votes are cast according to a given model of voting behavior.

Each *iteration* in a run of the experiment can be interpreted as corresponding to a single user, who chooses which reviews to vote on (participation) and what vote to cast (polarity) according to one of the AVM or RVM models, again with some specified parameters. Note that multiple votes can be cast during a single iteration, but at most one vote is cast per review. An alternative view is that each iteration represents a fixed time interval (chosen so that we get at most one vote on each review) during which different users cast votes on any review in accordance with attention bias.

The ranking algorithm under consideration recomputes the ranking of reviews for each product after one round of votes, *i.e.*, once after each user has finished casting votes. This is a reasonable design choice—recomputing the ranking after

[1]Note that simulating multiple products, say n products, is equivalent in this setup to repeating one run of the experiment n times on a single product (*i.e.*, with a fresh set of reviews each time) to lower the variance in the results.

each single vote is expensive, and also does not correspond to a believable user experience where a user sees the rankings of displayed content change *while* she is browsing through the reviews for a single product. Note that this workflow setting is general enough so that it can be easily modified to allow for performing multiple iterations (instead of just one) between recalculating the ordering to lower computational costs if necessary.

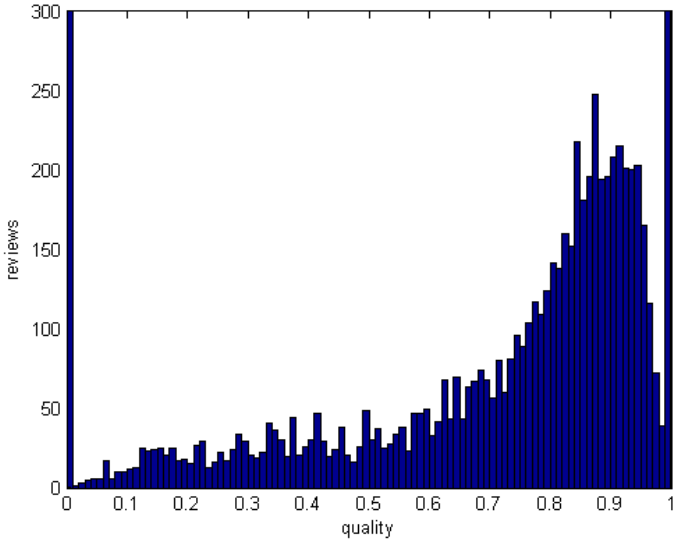
Algorithm 5 Simulation workflow.

```
for all products do
  generate reviews
  for all iterations do
    for all reviews do
      if decide to participate then
        cast helpfulness vote
      end if
    end for
    estimate model parameters
    rerank items
  end for
end for
```

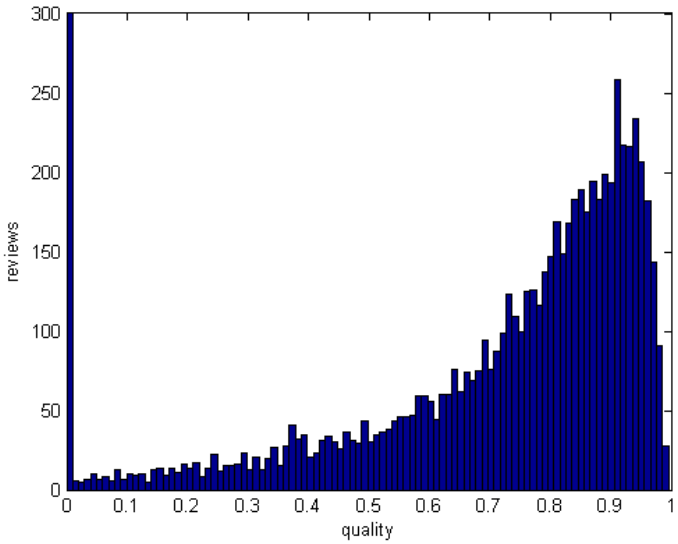
7.2.2 Parameter Settings

In the following sections, Algorithm 5 will be used to compare the performance of different ranking algorithms (in particular, estimators of content quality) under different user voting models. The settings for the parameters in Algorithm 5 that are common across all the simulations are presented in Table 7.1; the rationale behind some of these choices is discussed below.

Number of products. The number of products, or equivalently the number of simulation reruns, was chosen as a tradeoff between the size of error bars and computation time: increasing the number of products decreases the standard error



(a) observed data



(b) lognormal distribution

Figure 7.2: Review quality distributions.

(due to larger sample size) but increases computational cost. Therefore a value was chosen that is large enough to get sufficiently small standard error (for making performance significance comparisons between different experiments) while still allowing simulations to terminate in a reasonable amount of time.

Parameter	Value
Number of products	100
Number of reviews	30
Number of iterations	100
Comparison point	200 votes
α	0.228
β	0.415
γ	0.8
Mean and variance of normal	$\mu = 0.8$, $\sigma = 0.1$
Mean and variance of lognormal	$\mu = 0$, $\sigma = 1$

Table 7.1: Simulation settings used for all our experiments unless otherwise noted.

Number of reviews. Simulation runs on 30 reviews because this is equivalent to more than one page of reviews presented on Amazon (they show 10 reviews per page) and because [93] used this number (products on Amazon have overall enough reviews to get enough data about the top 30).

Number of iterations. The number of iterations should be chosen to allow adequate time for the rankings to converge, so that the steady-state error/performance of algorithms can be measured and compared, as well as the time to convergence.

Varying number of votes. Note that two experiments with the same number of iterations, *i.e.*, users arriving at the website, can still have a different number of actual accumulated votes, for two reasons: (i) first, changing the parameters in the model of user voting participation can lead to a smaller or larger number of *expected* votes cast, even if the number of users who vote does not vary (for example, if there is a sharper attention bias falloff, each user will vote on fewer reviews), and (ii) the decision of whether to vote on any given review or not—participation—is a random variable, so that the actual number of cast votes may differ across iterations even if the participation parameters, and therefore the expected number of votes, remain the same across runs.

Comparison point. When we want to compare two traces of runs we need a point of comparison (*i.e.*, at which timepoint do we consider the difference in the loss of utility). Using a mesure that directly compares two traces (by e.g. norm of the difference in performance between all points in traces) can be biased by including more or less of the converged state. Therefore 200 votes were chosen becuase this gives the algorithm some time to collect information[2] but is well before convergence (*i.e.*, we do not care so much about how good a ranking is after very long time has passed because users moved on to other products).

Choice of model parameters (α, β, γ). Parameter γ used in siumations was obtained by fitting the model to real-world data [93]. Models in [93] using α and β are not exactly the same (although they do have the same functional form) as the ones we use in this paper (we use context defined by qualities, while the published parameter values correspond to global context defined by logarithm of rank difference). However, we do know that they should be non-zero and choosing small numbers falls into pessimistic case (we get better performance when context plays larger role as discussed later in the paper).

Distribution of content quality. The polarity of the votes cast on a review in both the absolute and relative voting models depends on the review's quality. It is, therefore, quite possible that the performance of an algorithm—which depends on the the computed rankings at each step, which in turn depends on the vector of cast votes—might depend strongly on the distribution from which review qualities are drawn. Therefore the performance comparsions are repeated across multiple distributions—the uniform, the normal, and the lognormal.

[2]To completely specify a ranking of 30 review we need at least 187 votes. Expecting users to optimally select reviews to cast votes on is unrealistic. Therefore we should not expect convergence before a lot more votes than this are cast.

The parameters of the normal distribution were chosen such that it is not symmetric with respect to good and bad content (*i.e.*, the mean is not at quality of 0.5) and has noticeably narrow peak (to contrast with uniform distribution). In addition to this we can see by looking at (transformed) lognormal distribution shown on Figure 7.2(b) that it qualitatively appears very similar to what we observed in the data (Figure 7.2(a)). The observed distribution of qualities was obtained from data in [93] (using their rank-based global context model). To obtain the desired shape the following transformations are performed: the x-axis is rescaled by factor 0.2, the direction is reversed using $1 - x$ and the values are clamped to $[0, 1]$ interval.

7.2.3 Metrics for Performance

The primary quantity of interest when evaluating an algorithm is the 'goodness' of the ranking it produces. While all described algorithms also do compute estimates of reviews' true qualities, the focus is on the accuracy of the overall rankings rather than the accuracy of individual quality estimates, since it is the ranking returned by the algorithm, rather than the quality estimates, which determine the utility of the presented content to a viewer on the website.

There is a range of different measures to capture the 'goodness' of a ranking; the measure used here is the loss of utility defined as the sum of squared differences between item's true quality (known due to siumlated setting) and its eastimate from the model. Another possible measue to use is the number of bubble sort swaps necessary to correct the presented ranking into the correct one. This measure also results in qualitatively similar figures.

7.3 The Ratio-Sort Algorithm with Contextual Voting

This section analyzes the performance of the widely used ratio-sort algorithm which ranks contributions in decreasing order of the ratio of positive to total received votes, when votes are drawn from the contextual voting model. The ratio of the positive to total received votes is the minimum-variance unbiased estimator as well as the maximum-likelihood estimator of a review's true quality q *if* the polarity of cast votes are drawn IID Bernoulli(q), *i.e.*, if users vote according to the absolute voting model (AVM). Ranking according to this ratio, therefore, is a natural choice for a website that wants to display higher quality content at higher ranks—provided the AVM accurately describes user behavior, which, as shown in [93], it does not. How does this 'optimal' estimator for the AVM perform when its input comes from 'real' users who cast context-dependent votes instead?

First, we need to fully specify the ratio-sort algorithm by defining what happens to items with no votes, and how we resolve ties between items with the same ratio. To do this, the algorithm is modified slightly to sort according to the ratio of positive votes to total votes plus one (7.4), with a small added random perturbation ϵ for tiebreaking:

$$p_r = \frac{\text{number of yes votes on } r}{\text{total number of votes on } r+1} + \epsilon. \tag{7.4}$$

Sorting according to this ratio instead of the ratio of positive to total votes does not change the ordering of items, except for those with no votes (which previously had an undefined score)—these now have a score of zero and are therefore placed at the bottom of the rankings. The additive perturbation of ϵ implements random tie-breaking between items with the same vote ratio. Note that depending on the magnitude of ϵ, this additive random noise can also reverse the ordering between

items simulating exploration. To minimize effect of this and emphasise only tie-breaking a small random $\epsilon \in [0, 0.001]$ is chosen.

7.3.1 Does Context Bias in Votes Help or Hurt?

Let's now use the simulation setup described in Section7.2 to analyze the performance of the ratio-sort algorithm, which could be interepreted as designed for absolute voting, under a contextual voting model. We can now compare the performance of the ratio-sort algorithm when votes are drawn according to the RVM (*i.e.*, where context biases the votes cast on a review, which is a better model of observed user voting patterns) against that when votes are cast based purely on content quality according to the AVM. We can conduct two runs of the simulation which are identical in all parameters except users' voting participation and polarity models, with votes drawn from the AVM in one run and from the RVM in another.

The results of this comparison is shown on Figure 7.3. The x-axis corresponds to time and measures the cummulative number of received votes over all reviews; as described in Section7.2, the y-axis is the sum-of-squares distance between the quality vectors corresponding to the current and true rankings. Data points are plotted at each iteration with error bars corresponding to standard error over all products.

The ratio-sort algorithm achieves similarly poor utility in its first few iterations on both sets of inputs. This is likely due to the randomness in the initial rankings—at first, the context of each contribution is essentially random and does not significantly alter voting relative to the AVM, and becomes relevant only in later iterations where users strive to fix the ranking where necessary (and convey

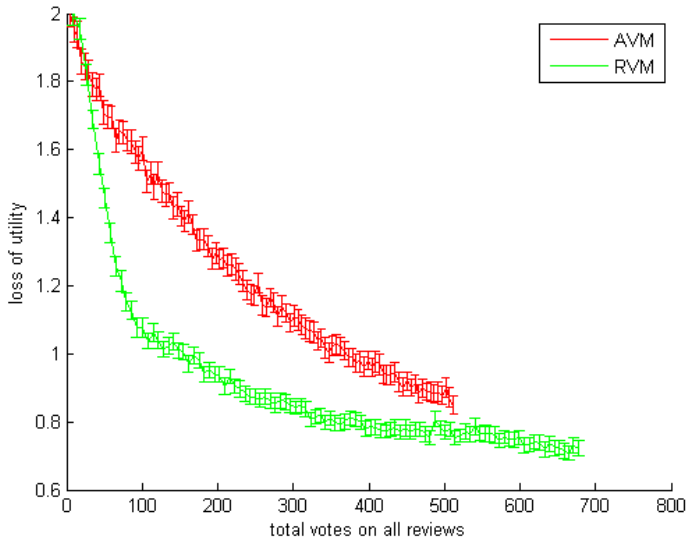


Figure 7.3: Comparison of AVM (red) and RVM (green) when using the ratio-sort ranking algorithm.

more information through their votes than with independent quality-based votes). The more interesting dynamics occur after the initial phase—the algorithm with input votes from the RVM suffers much lower utility losses at the same number of votes collected over all reviews (*i.e.*, the amount of user feedback) compared to when votes are cast according to the AVM; also the ratio-sort algorithm with inputs from the RVM achieves the same level of performance with far fewer votes than with inputs from the AVM. There is no visible convergence towards zero loss on figures becuase the variance of cast votes is high relative to differences in true qualities between reviews (meaning that we need a large number of votes to correctly order the reviews that are very close in quality).

The higher performance after any given number of votes as well as the smaller time required to achieve any given performance in Figure 7.3 suggests that the ratio-sort algorithm, designed to be optimal for the absolute voting model, in fact performs even better with context-dependent votes—that is, the particular kind

of contextual voting displayed by users is 'more informative' than independently drawn votes that are based only on a contribution's quality.

Informativeness of contextual votes. We saw that contextual votes (RVM) carry more information (*i.e.*, we can compute a better ranking with the same amount of received user feedback) than pure quality based voting. Why is that? From the theoiretical perspective we can consider the following case. Let's focus on ratio-sort algorithm and assume that we do not have a correct estimate (*i.e.*, the ratio of positive votes differs from the item's true quality value) but think that the quality is lower than the truth (*i.e.*, item is under-ranked). What is the expected number of new votes we need (added to the ones received in the past) to be able to obtain the correct estimate (up to some precision)? Because of the contextual term in RVM, we know that the probability of a positive vote will be slightly higher (due to being under-ranked) than in the AVM case (which cares only about the inherent quality). Therefore, the expect number of votes needed to correct/increase our current estimate/ratio will be lower for the RVM case (due to it having a higher probability of a positive vote).

Another case we can analyze is comparison between AVM and deterministic upvotes/downvotes. The later is defined as RVM with $\beta = 1$ (*i.e.*, purely contextul) and deterministic votes: if the item is under-ranked it always gets a positive vote and vice versa. The AVM case (votes are n independent Bernoulli random variables) can be analyzed by applying Hoeffding's inequality: the variance of the quality estimate drops with $1/\sqrt{n}$.

For the detrministic upvote/downvote case we can use induction to derive the bound on the estimate. Base case is a single vote, either 0/1 or 1/1, which is within 1/1 of the true quality. Assuming that in the previous induction step we had ratio

d/n (*i.e.*, d positive out of n total votes) which was within $1/n$ of the true quality. After one more positive vote we have $(d+1)/(n+1)$ for the under-ranked case (the other one is similar). In the worst case (the previous estimate was only ϵ below the true quality) we move at most $(d+1)/(n+1) - d/n = 1/(n+1)$ into over-ranked scenario (otherwise we improve the estimate for $1/(n+1)$ which brings us within $1/(n^2+n) \leq 1/(n+1)$ of the true value). Therefore the estimate of the quality falls within $1/n$ (*i.e.*, it is better than AVM case).

7.4 A Ranking Algorithm that Models Context Dependent Votes

The previous section looked only at what happens in the case that users vote differently than we expected when we are using the helpfulness vote ratio to estimate content quality. We saw that votes depending on context help even if the ranking function isn't aware of it. But, can we do better if we do know about it?

Lets use this additional knowledge to improve the results. Given a model of users voting behavior we can design an estimator of content quality tailored to the votes we are receiving. This way we create a consistent setting where users' model agrees with web page's expected model and thus allows us to extract more from the same feedback.

7.4.1 Context Aware Ranking Algorithm

Instead of using the ratio of positive to total votes to rank we want to use the true qualities to sort the items. In the case of RVM the accumulated votes do not always directly reflect the item's true quality – the decision about which ranking to present to the user can affect the polarity of received votes (e.g. under-ranked items get more positive votes).

Assuming the RVM, we have an explicit parameter q_r representing true quality of item r. To obtain these we need to compute the parameters for the new model by fitting it to the data. Here we can use maximum likelihood estimation to fit the model. If we assume prior knowledge of trade-off parameters α and β then RVM becomes traditional logistic regression and can be efficiently solved. By doing this we obtain item qualities (as the fitted parameters of the model) and can use them for sorting to create new ranking.

In this section we assume that trade-off parameters α and β are constant, known and provided to us. This is a reasonable assumption because they can be obtained by fitting a large amount of historical data and we would expect them to remain more or less constant – we can conjecture that they depend mainly on web site design (e.g. wording of the question eliciting the feedback, target demographic etc). Also, we do not need very accurate estimates of these parameters as is discussed later on in subsection 7.4.4 on robustness. Another option is to estimate these two parameters at the same time as we estimate the rest of the model during each iteration. However, the model is not traditional logistic regression anymore and requires more work to estimate.

As for the first experiment we can again use simulations to explore this new

algorithm. The overall workflow stays the same as in the previous section, but now the ratio-sort ranking algorithm is replaced with the context aware one. The model is refit to the collected data (from all previous iterations within the same simulation run) after each iteration of voting. The model is independent between products (a possible exception to this would be cross-listed reviews[3] which is not considered in the simulations) and can therefore be independently fitted. When necessary for computational reasons, one could consider incremental updates (instead of fitting the whole model from scratch during each iteration) or re-estimating the qualities only every k iterations instead of each one (in this case we trade computational cost for possible loss of utility due to not immediately exploiting all the available information).

This iterative approach of collecting votes and refitting the model, however, has the following caveat: the samples used in maximum likelihood estimation are not IID. Votes collected up until now determine the next presented ranking which in turn influences the new votes – ranking controls the context and context modifies polarity of votes and participation. However, in the experiments this does not seem to have a noticeable determential effect. Comparisons with (a) holding the ranking fixed for a multiple iterations (and thus improving IID-ness of samples) and (b) cheating (*i.e.*, computing loss on the estimated ranking but presenting a random one to the simulated user resulting in IID samples) do not show a noticeable difference.

[3]For example, Amazon sometimes lists the same review for multiple different products.

7.4.2 Does the New Algorithm Learn Faster?

Previous comparison between AVM and RVM was performed by changing users' behavior but keeping the ranking algorithm the same (naive ratio based). Now we want to see if we can further improve on this by changing the algorithm. Switching to context aware ranking algorithm (described in previous subsection) should give us better results than the naive one due to it being tailored to the received feedback.

Figures 7.4 and 7.5 show the comparison between ratio-sort (green) and context aware (blue) algorithm when users vote according to RVM. There is not much difference during the initial few votes because the context is still random (*i.e.*, the ranking starts as random ordering and thus has meaningless context). After that the new algorithm manages to recognize contextual votes and accounts for them when estimating the quality (*i.e.*, contextual votes represent user's desire to change the position of an item and do not directly describe the absolute quality). This leads to better performance (as soon as context becomes informative, *i.e.*, non-random). The ratio-sort algorithm incorrectly interprets contextual votes and converges more slowly due to higher variance in the estimates of quality.

Another interesting detail can be observed if we focus on the first few iterations. We see that the new algorithm appears to be doing worse for a bit before it overtakes the ratio-sort one. This could be explained by the fact that it is trying to fit a more complex model with a very small amount of data – we are trying to estimate attention bias (z_p for all ranks p), item qualities (q_r for all items r) and interactions with context using only a handful of votes, while on the other hand naive algorithm only cares about quality estimates. However, as soon as we get enough data we can form better estimates based on knowledge of the voting process.

7.4.3 How does Improvement Scale with Strength of Context Bias?

We saw that in case of RVM we obtain more information from the feedback and achieve better ranking performance. However, we did not yet address the question of what role do α in β play. Do we get the same improvement for all possible choices of these two parameters (hint: obviously no, setting them to 0 results in special case of AVM)? And, why are we even asking this question – these two parameters are constant and given to us? First, we want to know if the approach is applicable across a broad range of websites – is is unrealistic to expect that they all have the same value of there parameters. And secondly, if we know in which case we get the best results we can then try and design websites in such a way that they elicit feedback in the desired range of this spectrum.

In line with previous experiments simulations are used but this time with varying degrees of contextual influence. Note that robustness will be evaluated separately in the next subsection. The following experiment is performed. For a given choice of α and β simulations are run with RVM and both algorithms. Then the performance is compared at the total of 200 votes point. This number was chosen to be after the initial volatile part but before a large number of votes is collected (200 votes spread over 30 reviews is less than 10 per on average). The first is necessary to get low variance in comparisons and later because we do not care (that much) about utility after a long time has passed (for example, having perfect rankings on old products does not help if users moved to new ones – which now have bad rankings due to being new).

Results are presented on Figure 7.6(a). Parameters $\alpha = \beta$ are changed simulta-

neously within the $[0, 1]$ interval (shown on x-axis). Then we can compare ratio-sort and new algorithm by loss of utility (y-axis). When both $\alpha = \beta = 0$ we are in special case where the context-aware algorithm becomes the same as the ratio-sort one (and votes follwo AVM); so the same performance is expected. On the other extreme we have purely context based votes – they are only up and down votes with no explicit information about quality. If we measure expected loss of utility (difference between the estimated quality and the true quality in the simulation) then even if we get correct ordering in the ranking (from the observed votes) we can still have incorrect estimates of quality – so we should not realistically expect zero-loss when using this performance measure.

From the figure we can see that increase in contextual part of votes leads to better performance for both algorithms. However, having algorithm designed for the voting model results in better rankings throughout the spectrum. From this we can conclude that as long as we have additional knowledge about the voting model we should exploit it – the gains might vary from application to application, but the benefit can be obtained in all cases.

A possibly important difference across websites (in addition to amount of contextual information in the votes) could be the steepness of the attention bias drop-off for participation captured in the models by γ. We should ask ourselves how well can we perform when users look only at the top results, or perhaps tend to look through most of the content. Parameter γ is intrinsic to a given site and used in the simulations, but note that it is not necessary for the model estimation part (*i.e.*, we do not need to know its value to be able to estiamte model parameters z_p).

Figure 7.6(b) looks at the behavior associated with varying γ in the interval

$[0, 1.5]$ (larger values correspond to more pronounced attention bias). In this case too the context-aware algorithm outperforms other choices and we see smooth degradation in performace as we get higher attention bias (reviews lower in the ranking get less votes and thus we have a poorer estimate of their quality).

7.4.4 Robustness

The new algorithm and RVM itself result in gains across widely ranging amounts of contextual information included in the votes. Up until now it was assumed that we know the correct values of parameters α and β (*i.e.*, they are provided to us, e.g. based on historical data). How robust is this approach to less accurate estimates of these parameters? For example, will this approach still work if we have bad estimates or if they change over time; or are we better off playing safe and using the ratio-sort algorithm?

Robustness is explored on Figure 7.7(a). The belief about true α and β is fixed and the actual values are varied in data via simulations. If the approach is robust then we should see smooth and slow degradation of performance as data moves further away from the belief of the algorithm. All points on figure use $\alpha = \beta = 0.5$ in the algorithm, while data spans a large interval. We can observe that performance does not change drastically. Another important observation is that as we move further towards higher influence of the context the performance steadily increases (and reverse for the opposite direction) despite the parameter estimates being more and more wrong. This is because the results in this plot show the joint effect of incorrect parameter estimates and higher informativeness of contextual votes. Because the latter turns out to be a larger factor we can observe gradient in performance on this figure that is aligned with the parameter

values.

To get a better sense of what is actually good or bad, we can comapare the results to using ratio-sort algoritm as a baseline. Figure 7.7(b) shows differences in the performance, with positive values meaning that the context-aware algorithm performs better than the baseline ratio-sort approach. Note that the standard error is approximately 0.05 when drawing conclusions about the differences. We see that context-aware algorithm performs better for the most part, except for being about equal when assuming high amount of contextual information while the data is mostly from AVM.

7.4.5 New Reviews over Time

Another important case we should consider (becuase it is closer to real-world scenarios) is new reviews arriving over time (instead of all already being present at the beggining as in the previous experiments). We can modify the framework to start with a single review and then add new ones over time.

Figure 7.8(a) shows results for the case of adding one new review each iteration (until we reach the full set of 30 reviews). Performance is measured only on the reviews existing at that timepoint. Note that total loss of utility increases with larger number of reviews (due to how it is defined) even if the average per review quality estimates have the same or lower variance. This is why we see increase in loss until iteration 30, after which all reviews are present. In this experiment too, we can observe the same performance characteristics as before (*i.e.*, the context-aware algorithm outperforms ratio-sort and RVM gives more information than AVM).

If new reviews do not arrive every iteration but at longer regular intervals we get traces as seen on Figure 7.8(b). Here, a new review gets added every 10 iterations. Afterwards, its quality estimate rapidly improves over the nnext few iterations. Ratio-sort algorithm is significantly worse in this case too.

7.5 Summary

Focusing on user generated content (e.g. product reviews) and the need to promote good content (by e.g. collecting user feedback on helpfulness of contributions), This chapter explored interactions between user voting models and ranking algorithms. Two competing models were discussed: Absolute Voting Model (where users vote according to content quality) and Relative Voting Model (where context in which an item is presented plays a role). Analysis of data suggests that real-world votes are not cast according to AVM, but include context bias. Based on simulations we see that votes cast according to RVM carry more information and allow ratio-sort based ranking algorithm (*i.e.*, quality of a contribution is estimated by ratio of positive to all votes) to converge faster despite it not being aware of mismatch between the RVM votes and its assumption of the model (AVM).

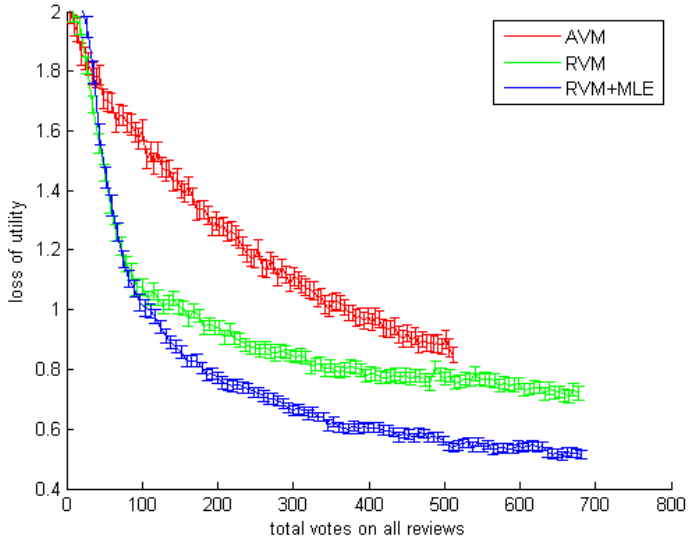
Knowing more about the user voting model allows us to tailor the ranking algorithm to the actual users' behavior. To improve upon ratio-base algorithm, MLE was used for obtaining content qualities based on RVM. Again, using a simulated setting we see that this does indeed further improve the performance. Furthermore, this improvement scales smoothly with the increase in the amount of contextual information in the votes.

The approach is robust to varying amounts of contextual influence. If the

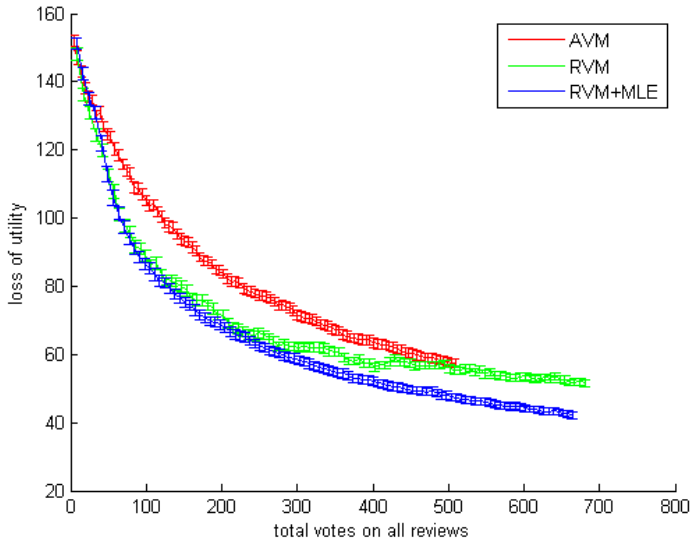
ranking algorithm assumes a certain amount of contextual information in the votes, good performance (*i.e.*, better than using ratio-sort in the interesting part of the parameter space) is achieved even if data does not come from the exactly same distribution.

Let's note here a related line of work: bandit algorithms. A classical multiarmed bandit setting [4] is remeniscent of AVM in combination with ratio-sort algorithm, where we try to discern the true values of content items and attan higest user satisfaction by presenting best content in the most viewed spots. The purely contextual version of RVM (at $\beta = 1$) is in broad view related to dueling bandits [124], where we look at the preferences between items. The approach described here in its entirety, however, deals with a mixture of both and allows for receiving multiple (possibly dependent) pieces of feedback at the same time. Furthermore, the approach specifies how to estimate content quality but does not prevent us from incorporating an additional exploration strategy on top of it (e.g. it could be combined with [80]).

To sum up, this chapter showed presence of context bias in observed data and then demonstrated that votes including contextual information (in addition to item quality) carry more information and result in faster ranking convergence even if the ranking algorithm is not aware of the change in the model of users' behavior. Moreover, even better performance can be achieved by exploiting model knowledge and estimating content quality in accordance with users' voting model.

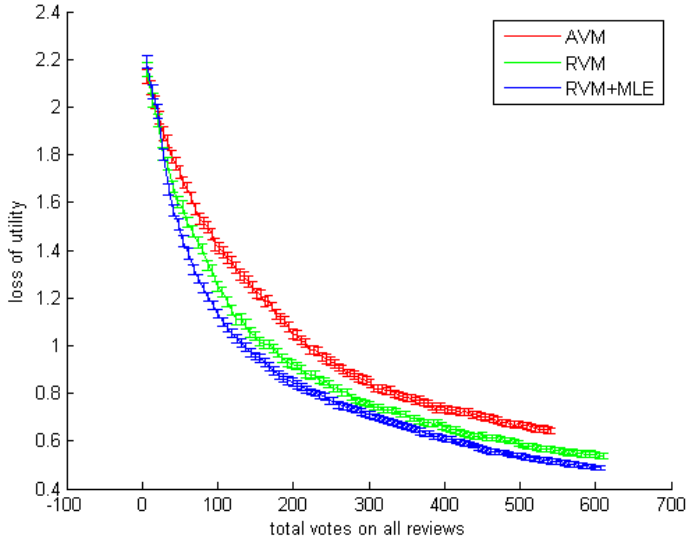


(a) lognormal, utility loss

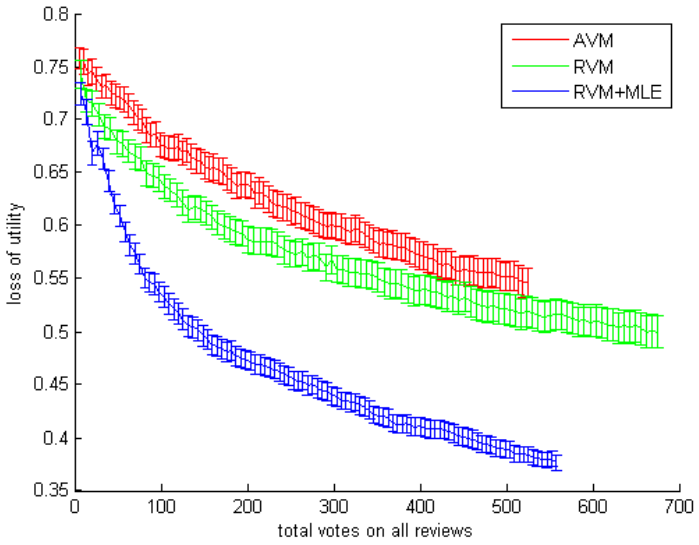


(b) lognormal, swaps

Figure 7.4: Comparison between AVM (red) and RVM (green) using ratio-sort versus RVM with MLE algorithm (blue) using different distributions of review qualities and different performance measures.

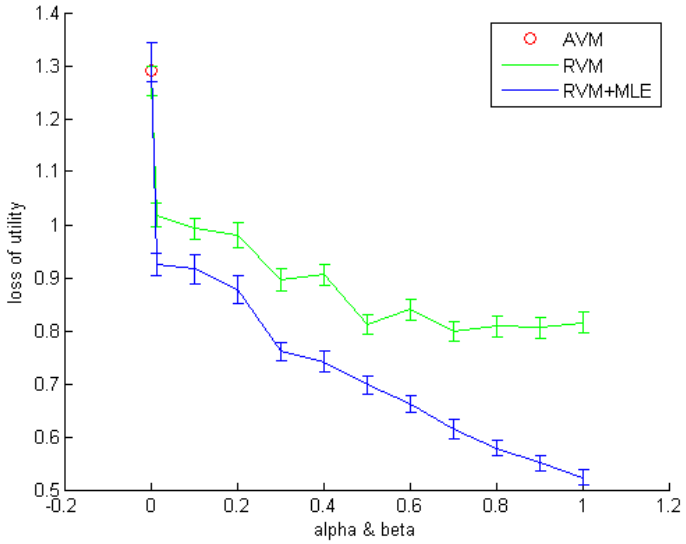


(a) uniform, utility loss

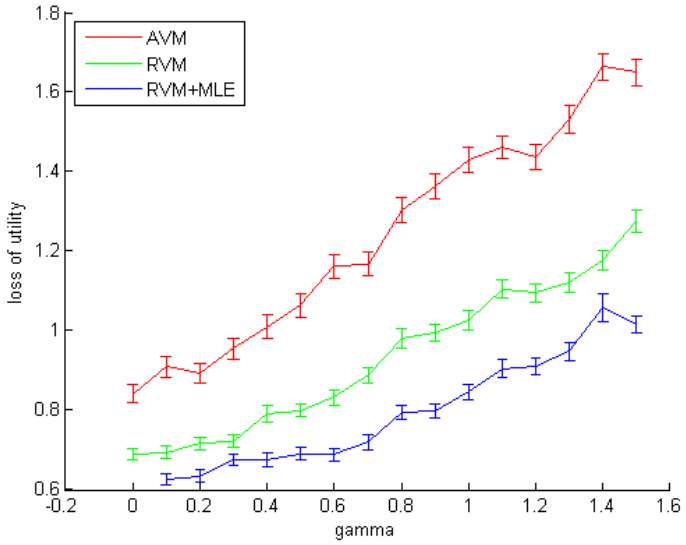


(b) normal, utility loss

Figure 7.5: Comparison between AVM (red) and RVM (green) using ratio-sort versus RVM with MLE algorithm (blue) using different distributions of review qualities and different performance measures.

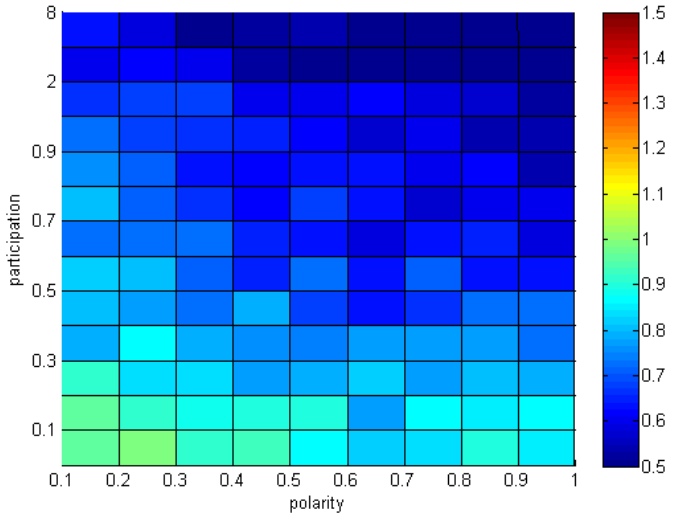


(a) context trade-off $\alpha = \beta$

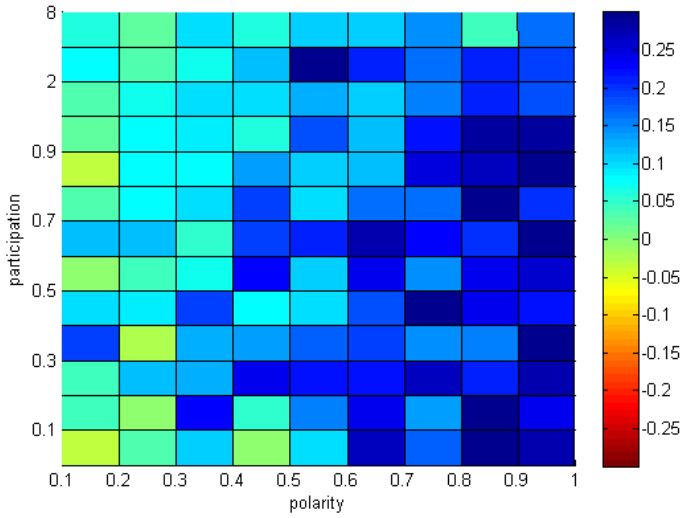


(b) attention bias γ

Figure 7.6: Sweep of the main model parameters showing smooth changes in performance and better performance of the new algorithm in all cases.

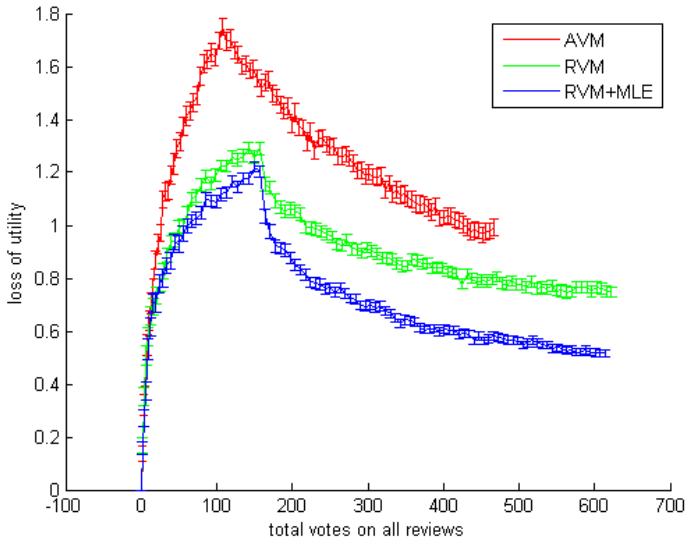


(a) RVM with MLE

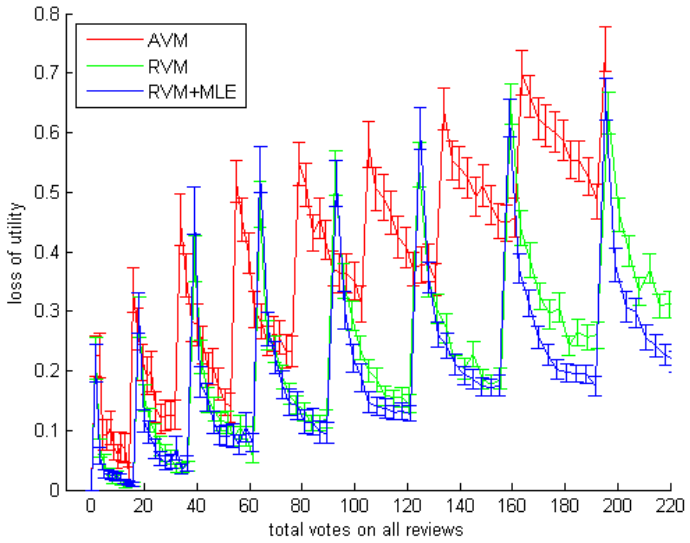


(b) comparison with baseline

Figure 7.7: Robustness for fixed assumed parameters α and β by the algorithm, while the data varies. Comparison plot shows how much better is the context-aware algorithm compared to the ratio-sort.



(a) new review each iteration



(b) new review every 10 iterations

Figure 7.8: A setting where new reviews arrive over time.

Part V

Conclusion

CHAPTER 8

CONCLUSIONS

Vast and rapidly growing amounts of digital data are a big contributor to accelerating technology progress we are all experiencing. New developments are fueled by accumulated information and shared knowledge. On the other side of the coin, the growth is outpacing our own abilities to filter, process and assimilate desired information. To counteract this, we have also been building information retrieval and other systems on the parallel tract to data collection. For example, we already have very sophisticated (and effective) keyword-based web search engines. There is also past work that covers other user needs by e.g. condensing information distilling large amounts of data into small packages that contain all relevant information, avoid redundancy, give high-level insight and are easily consumable by us in a limited timeframe. This thesis strives to add another piece to this by expanding the toolbox with novel automated approaches covering as-of-yet unaddressed user needs.

The first half of this thesis focuses on condensing information by summarizing. Starting with a large amount of data, one can construct a brief summary that contains relevant information for the user who can then in a short amount of time obtain the desired insight. Summarization in the sense of information condensation is not limited to plain text but can take other forms to address specific user needs (e.g. summary can be a timeline). The latter half focuses on crowdsourcing. The main idea is to collaborate with the users and use freely provided user feedback to help surface the good content (and thus distill larger collections into quality content).

Starting with improvements to approaches for traditional summarization task,

Chapter 3 presented a supervised learning approach to extractive multi-document summarization using structured output prediction based on structual SVMs, which models summaries as multivariate objects and can thus capture dependencies within them. The learning method applies to all submodular scoring functions, ranging from pairwise-similarity models to coverage based approaches. Submodular functions have the desirable property of diminishing returns and have also been applied to many other summarization-like tasks. The learning problem was formulated as a convex quadratic program and was then solved approximately using a cutting-plane method. The submodular objective can be optimized greedily and has a constant factor approximation guarantee. In an empirical evaluation, the structural SVM approach significantly outperforms conventional hand-tuned models. A key advantage of the learning approach is its ability to handle large numbers of features, providing substantial flexibility for building high-fidelity summarization models. Furthermore, it shows good control of overfitting, making it possible to train models even with only a few training examples.

The framework of submodular scoring functions is flexible and can be applied to a different information condensation task. Chapter 4 presents an approach to selecting pairs of snippets from reviews in a way that creates a summarizing product comparison. The scoring function strives to select aligned pairs (both snippets are about the same aspect) with good coverage of important aspects and low redundancy. The objective function is submodular and thus efficient to optimize with a constant factor approximation. The experiments show that it outperforms a naive baseline even with the uniform weights model. Using a supervised learning approach it can achieve generalization across different product pairs by using user feedback on the presented pairs.

Adding temporal dimension and using unsupervised setting we can define another task that helps the user gain insight into evolution of a given corpus over time. Chapter 5 presents a submodular framework for temporal corpus summarization which is an extension of objective functions used in the previous chapters. The notion of word coverage is extended so that the summaries cover important concepts by covering associated words over a time interval. A timeline of influential documents, or authors, or coherent key phrases can be constructed using this approach, thus providing concrete suggestions for further and more detailed exploration of the corpus contents. The approach leveraged both the novelty of a document (i.e. document introduces novel ideas into the corpus) as well as its influence (i.e. documents at the later dates expand upon its ideas) in the development of the corpus and relied only on word features; in particular, it does not require a citation structure to infer influence across time in a corpus of scientific publications. Therefore it is applicable to any textual collection which provides timestamped documents. The optimization objectives is again monotone submodular which can be efficiently solved. The evaluation demonstrates that the approach performs better than several baselines using citation based performance measures.

To leverage the crowdsourcing, we should first understand the user behavior and their feedback. Chapter 6 provides an analysis of user voting patterns on Amazon reviews, over time and as a function of the context in which the review was rated by the user. Using a newly collected dataset of daily snapshots of reviews and their ratings, we can observe that voting polarity (i.e., whether to assign a positive or negative helpfulness vote to a review) as well as participation (i.e., whether to vote at all) depends not only on the inherent quality of a review, but also on the context in which it is presented at the time of voting. In particular, there is evidence that the observed connection between context and voting behavior cannot be captured

by absolute and independent judgments about helpfulness.

An alternative descriptive model of the behavior that incorporates the context provides a better fit to the observed data. Notably, voting polarity becomes more positive/negative, if a review is better/worse than its context. Furthermore, voting participation generally increases if a review is misranked. The discovered patterns are substantially different from other setting where endorsements are used to rank items (e.g. Web Search).

After gaining some insight into the user behavior, we can try to exploit this additional knowledge to improve ranking of the content (and thus help the user by surfacing the good content). Chapter 7 explores interactions between user voting models and ranking algorithms. Two competing models: Absolute Voting Model (where users vote according to content quality) and Relative Voting Model (where context in which an item is presented plays a role) are compared using simulations. Votes cast according to RVM carry more information and allow ratio-sort based ranking algorithm (i.e., quality of a contribution is estimated by ratio of positive to all votes) to converge faster despite it not being aware of mismatch between the RVM votes and its assumption of the model (AVM). Knowing more about the user voting model allows us to tailor the ranking algorithm to the actual users' behavior. The ranking convergence speed can be further improved by using MLE instead of the nave ratio-sort algorithm for estimating review qualities based on votes cast according to RVM. This improvement scales smoothly with the increase in the amount of contextual information in the votes.

This thesis presented an improved way of summarizing that combined submodular objective functions (which can be efficiently optimized) with supervised learning to achieve new state-of-the-art performance. The same framework can

be applied to different tasks, ranging from traditional summarization to temporal and comparative summaries, thus covering different user needs by automatically condensing information in different ways. An important resource is also user feedback. The second part of this thesis asks how can we condense information and bring good content to the user in the crowdsourced setting where we get free user feedback. An analysis of a real-world data shows that votes cast by users can be biased by the context in which they were cast. By building upon the descriptive models of the observed behavior we can improve ranking algorithms and achieve faster convergence of the ranking of the content and thus bring good and relevant content to the user.

BIBLIOGRAPHY

[1] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 183–194, New York, NY, USA, 2008. ACM.

[2] James Allan, Rahul Gupta, and Vikas Khandelwal. Temporal summaries of new topics. In *SIGIR*, pages 10–18, New York, NY, USA, 2001. ACM.

[3] Georgios Askalidis and Greg Stoddard. A theoretical analysis of crowd-sourced content curation. In *The 3rd Workshop on Social Computing and User Generated Content*, 2013.

[4] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002.

[5] Philip Beineke, Trevor Hastie, Christopher Manning, and Shivakumar Vaithyanathan. Exploring sentiment summarization. In *AAAI Spring Symposium*, 2004.

[6] T. Berg-Kirkpatrick, D. Gillick, and D. Klein. Jointly learning to extract and compress. In *ACL*, 2011.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, 1998.

[9] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization*, IPCO '07, pages 182–196, Berlin, Heidelberg, 2007. Springer-Verlag.

[10] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, New York, NY, USA, 1998. ACM.

[11] Chien Chin Chen and Meng Chang Chen. Tscan: a novel method for topic

summarization and content anatomy. In *SIGIR*, pages 579–586, New York, NY, USA, 2008. ACM.

[12] P. Chen, H. Xie, S. Maslov, and S. Redner. Finding scientific gems with google's pagerank algorithm. *Journal of Informetrics*, 1(1):8 – 15, 2007.

[13] Jaehoon Choi, Donghyeon Kim, Seongsoon Kim, Junkyu Lee, Sangrak Lim, Sunwon Lee, and Jaewoo Kang. Consento: a new framework for opinion based entity search and summarization. In *CIKM*, pages 1935–1939, New York, NY, USA, 2012. ACM.

[14] J. M. Conroy and D. P. O'leary. Text summarization via hidden markov models. In *SIGIR*, 2001.

[15] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 87–94, New York, NY, USA, 2008. ACM.

[16] Cristian Danescu-Niculescu-Mizil, Gueorgi Kossinets, Jon Kleinberg, and Lillian Lee. How opinions are received by online communities: A case study on amazon.com helpfulness votes. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 141–150, New York, NY, USA, 2009. ACM.

[17] Harold Charles Daume, III. *Practical Structured Learning Techniques for Natural Language Processing*. PhD thesis, Los Angeles, CA, USA, 2006. AAI3337548.

[18] Loretti I. Dobrescu, Michael Luca, and Alberto Motta. What makes a critic tick? Connected authors and the determinants of book reviews. *Journal of Economic Behavior & Organization*, 96(C):85–103, 2013.

[19] Khalid El-Arini and Carlos Guestrin. Beyond keyword search: discovering relevant scientific literature. In *KDD*, pages 439–447, New York, NY, USA, 2011. ACM.

[20] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *KDD*, pages 289–298, New York, NY, USA, 2009. ACM.

[21] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as

salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

[22] E. Filatova and V. Hatzivassiloglou. Event-based extractive summarization. In *ACL Workshop on Summarization*, 2004.

[23] Wei Gao, Peng Li, and Kareem Darwish. Joint topic modeling for event summarization across news and social media streams. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1173–1182, New York, NY, USA, 2012. ACM.

[24] Alexander F. Gelbukh, Mikhail Alexandrov, Ales Bourek, and Pavel Makagonov. Selection of representative documents for clusters in a document collection. In Antje Dsterhft and Bernhard Thalheim, editors, *NLDB*, volume 29 of *LNI*, pages 120–126. GI, 2003.

[25] Anindya Ghose and Panagiotis G. Ipeirotis. Designing novel review ranking systems: Predicting the usefulness and impact of reviews. In *Proceedings of the Ninth International Conference on Electronic Commerce*, ICEC '07, pages 303–310, New York, NY, USA, 2007. ACM.

[26] Arpita Ghosh and Preston McAfee. Incentivizing high-quality user-generated content. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 137–146, New York, NY, USA, 2011. ACM.

[27] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz. Multi-document summarization by sentence extraction. In *NAACL-ANLP*, 2000.

[28] Aria Haghighi and Lucy Vanderwende. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 362–370, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[29] F. Maxwell Harper, Daphne Raban, Sheizaf Rafaeli, and Joseph A. Konstan. Predictors of answer quality in online Q&A sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 865–874, New York, NY, USA, 2008. ACM.

[30] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[31] He Huang and Chunping Li. A unified graph model for chinese product review summarization using richer information. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, WISDOM '12, pages 2:1–2:9, New York, NY, USA, 2012. ACM.

[32] Xiaojiang Huang, Xiaojun Wan, and Jianguo Xiao. Comparative news summarization using linear programming. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 648–653, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[33] Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1189–1190, New York, NY, USA, 2007. ACM.

[34] T. Joachims, L. Granka, Bing Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), April 2007.

[35] Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1), 1999.

[36] Hyun Duk Kim and ChengXiang Zhai. Generating comparative summaries of contradictory opinions in text. In *CIKM*, pages 385–394, New York, NY, USA, 2009. ACM.

[37] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 423–430, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[38] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46:604–632, 1999.

[39] A. Kulesza and B. Taskar. Learning determinantal point processes. In *UAI*, 2011.

[40] J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *SIGIR*, 1995.

[41] Theodoros Lappas and Dimitrios Gunopulos. Efficient confident search in large review corpora. In *ECML PKDD*, pages 195–210, Berlin, Heidelberg, 2010. Springer-Verlag.

[42] Kevin Lerman and Ryan McDonald. Contrastive summarization: An experiment with consumer reviews. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 113–116, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[43] Jingxuan Li, Lei Li, and Tao Li. Multi-document summarization via submodularity. *Applied Intelligence*, 37(3):420–430, October 2012.

[44] L. Li, Ke Zhou, G. Xue, H. Zha, and Y. Yu. Enhancing diversity, coverage and balance for summarization through structure learning. In *WWW*, 2009.

[45] Xuan Li, Liang Du, and Yi-Dong Shen. Graph-based marginal ranking for update summarization. In *SDM*, pages 486–497. SIAM / Omnipress, 2011.

[46] Jung-Min Lim, In-Su Kang, Jae-Hak Bae, and Jong-Hyeok Lee. Sentence extraction using time features in multi-document summarization. In *Information Retrieval Technology*, volume 3411 of *Lecture Notes in Computer Science*, pages 82–93. Springer Berlin / Heidelberg, 2005.

[47] C. Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL*, 2003.

[48] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[49] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *ACL-HLT*, 2011.

[50] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *2010*, NAACL-HLT.

[51] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *HLT*, pages 912–920, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[52] Jingjing Liu, Yunbo Cao, Chin Y. Lin, Yalou Huang, and Ming Zhou. Low-Quality Product Review Detection in Opinion Summarization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 334–342, 2007.

[53] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. Modeling and predicting the helpfulness of online reviews. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 443–452, 2008.

[54] Chong Long, Min-Lie Huang, Xiao-Yan Zhu, and Ming Li. A new approach for multi-document update summarization. *J. Comput. Sci. Technol.*, 25(4):739–749, July 2010.

[55] Yue Lu, Panayiotis Tsaparas, Alexandros Ntoulas, and Livia Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 691–700, New York, NY, USA, 2010. ACM.

[56] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 131–140, New York, NY, USA, 2009. ACM.

[57] Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. Topic-driven reader comments summarization. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 265–274, New York, NY, USA, 2012. ACM.

[58] Walid Magdy, Ahmed Ali, and Kareem Darwish. A summarization tool for time-sensitive social media. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2695–2697, New York, NY, USA, 2012. ACM.

[59] Milind Mahajan, Patrick Nguyen, and Geoffrey Zweig. Summarization of multiple user reviews in the restaurant domain. Technical Report MSR-TR-2007-126, Microsoft Research, 2007.

[60] Takuya Makino, Hiroya Takamura, and Manabu Okumura. Balanced coverage of aspects for text summarization. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1742–1746, New York, NY, USA, 2012. ACM.

[61] Juha Makkonen and Helena Ahonen-Myka. Utilizing temporal information in topic detection and tracking. In Traugott Koch and Ingeborg Slvberg, editors, *ECDL*, volume 2769 of *Lecture Notes in Computer Science*, pages 393–404. Springer, 2003.

[62] F. T. Martins and N. A. Smith. Summarization with a joint model for sentence extraction and compression. In *ACL Workshop on Integer Linear Programming for Natural Language Processing*, 2009.

[63] R. McDonald. A study of global inference algorithms. In *Lecture Notes in Computer Science*, 2007.

[64] Xinfan Meng and Houfeng Wang. Mining user reviews: from specification to summarization. In *ACL-IJCNLP Short Papers*, pages 177–180, Stroudsburg, PA, USA, 2009. ACL.

[65] D. Metzler and T. Kanungo. Machine learned sentence selection strategies for query-biased summarization. In *SIGIR*, 2008.

[66] R. Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *ACL on Interactive poster and demonstration sessions*, 2004.

[67] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *EMNLP*, 2004.

[68] Lev Muchnik, Sinan Aral, and Sean J. Taylor. Social influence bias: A randomized experiment. *Science*, 341(6146):647–651, 2013.

[69] Susan M. Mudambi and David Schuff. What makes a helpful online review? A study of customer reviews on amazon.com. *MIS Q.*, 34(1):185–200, March 2010.

[70] Makoto Nakayama and Yun Wan. An exploratory study: "Blind-testing" consumers how they rate helpfulness of online reviews. In *Conf-IRM*, 2012.

[71] Ramesh Nallapati, Ao Feng, Fuchun Peng, and James Allan. Event threading within news topics. In *CIKM*, pages 446–453, New York, NY, USA, 2004. ACM.

[72] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approxima-

tions for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[73] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions–i. *Mathematical Programming*, 14(1):265–294, 1978.

[74] Ani Nenkova and Kathleen McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233, 2011.

[75] T. Nomoto and Y. Matsumoto. A new approach to unsupervised text summarization. In *SIGIR*, 2001.

[76] Michael J. Paul, ChengXiang Zhai, and Roxana Girju. Summarizing contrastive viewpoints in opinionated text. In *EMNLP*, pages 66–76, Stroudsburg, PA, USA, 2010. ACL.

[77] V. Qazvinian, D. R. Radev, and A. Özgür. Citation summarization through keyphrase extraction. In *COLING*, 2010.

[78] Dragomir R. Radev, Hongyan Jing, Malgorzata Styś, and Daniel Tam. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938, November 2004.

[79] Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. The ACL anthology network corpus. In *Proceedings, ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*, Singapore, 2009.

[80] Filip Radlinski and Thorsten Joachims. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 570–579, New York, NY, USA, 2007. ACM.

[81] K. Raman, T. Joachims, and P. Shivaswamy. Structured learning of two-level dynamic rankings. In *CIKM*, 2011.

[82] Karthik Raman, Thorsten Joachims, and Pannanaga Shivaswamy. Structured learning of two-level dynamic rankings. In *CIKM*, 2011.

[83] Gobaan Raveendran and Charles L.A. Clarke. Lightweight contrastive sum-

marization for news comment mining. In *SIGIR*, pages 1103–1104, New York, NY, USA, 2012. ACM.

[84] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Cornell University, Ithaca, NY, USA, 1987.

[85] Shilad Sen, F. Maxwell Harper, Adam LaPitz, and John Riedl. The quest for quality tags. In *Proceedings of the 2007 international ACM conference on Supporting group work*, GROUP '07, pages 361–370, New York, NY, USA, 2007. ACM.

[86] Dafna Shahaf and Carlos Guestrin. Connecting the dots between news articles. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 623–632, New York, NY, USA, 2010. ACM.

[87] Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. Trains of thought: Generating information maps. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 899–908, New York, NY, USA, 2012. ACM.

[88] B. Shaparenko and T. Joachims. Information genealogy: Uncovering the flow of ideas in non-hyperlinked document databases. In *KDD*, pages 619–628, 2007.

[89] Benyah Shaparenko and Thorsten Joachims. Identifying the original contribution of a document via language modeling. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 696–697, New York, NY, USA, 2009. ACM.

[90] Chao Shen and Tao Li. Multi-document summarization via the minimum dominating set. In *COLING*, pages 984–992, Stroudsburg, PA, USA, 2010. ACL.

[91] D. Shen, J. T. Sun, H. Li, Q. Yang, and Z. Chen. Document summarization using conditional random fields. In *IJCAI*, 2007.

[92] Stefan Siersdorfer, Sergiu Chelaru, Wolfgang Nejdl, and Jose San Pedro. How useful are your comments?: Analyzing and predicting youtube comments and comment ratings. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 891–900, New York, NY, USA, 2010. ACM.

[93] Ruben Sipos, Arpita Ghosh, and Thorsten Joachims. Was this review helpful to you?: It depends! context and voting patterns in online content. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 337–348, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.

[94] Ruben Sipos and Thorsten Joachims. Generating comparative summaries from reviews. In *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, pages 1853–1856, New York, NY, USA, 2013. ACM.

[95] Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 224–233, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[96] Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. Temporal corpus summarization using submodular word coverage. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 754–763, New York, NY, USA, 2012. ACM.

[97] Josef Steinberger and Karel Jeek. Using latent semantic analysis in text summarization and summary evaluation. In *In Proc. ISIM 04*, pages 93–100, 2004.

[98] Ilija Subašić and Bettina Berendt. From bursty patterns to bursty facts: The effectiveness of temporal text mining for news. In *ECAI*, pages 517–522, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[99] A. Swaminathan, C. V. Mathew, and D. Kirovski. Essential pages. In *WI-IAT, IEEE Computer Society*, 2009.

[100] Ashwin Swaminthan, Cherian Metthew, and Darko Kirovski. Essential pages. In *Technical Report, MSR-TR-2008-15*, Microsoft Research, 2008.

[101] Russell Swan and James Allan. Automatic generation of overview timelines. In *SIGIR*, pages 49–56, New York, NY, USA, 2000. ACM.

[102] Hiroya Takamura and Manabu Okumura. Text summarization model based on maximum coverage problem and its variant. In *EACL*, pages 781–789, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[103] Roberto Torres, Sean M. McNee, Mara Abel, Joseph A. Konstan, and John Riedl. Enhancing digital libraries with techlens+. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '04, pages 228–236, New York, NY, USA, 2004. ACM.

[104] Panayiotis Tsaparas, Alexandros Ntoulas, and Evimaria Terzi. Selecting a comprehensive set of reviews. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 168–176, New York, NY, USA, 2011. ACM.

[105] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[106] Oren Tsur and Ari Rappoport. Revrank: A fully unsupervised algorithm for selecting the most helpful book reviews. In Eytan Adar, Matthew Hurst, Tim Finin, Natalie S. Glance, Nicolas Nicolov, and Belle L. Tseng, editors, *ICWSM*. The AAAI Press, 2009.

[107] X. Wan, J. Yang, and J. Xiao. Collabsum: Exploiting multiple document clustering for collaborative single document summarizations. In *SIGIR*, 2007.

[108] Xiaojun Wan, Houping Jia, Shanshan Huang, and Jianguo Xiao. Summarizing the differences in multilingual news. In *SIGIR*, pages 735–744, New York, NY, USA, 2011. ACM.

[109] Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL*. The Association for Computational Linguistics, 2007.

[110] Dingding Wang and Tao Li. Document update summarization using incremental hierarchical clustering. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 279–288, New York, NY, USA, 2010. ACM.

[111] Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 307–314, New York, NY, USA, 2008. ACM.

[112] Dingding Wang, Mitsunori Ogihara, and Tao Li. Summarizing the differences from microblogs. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1147–1148, New York, NY, USA, 2012. ACM.

[113] Dingding Wang, Shenghuo Zhu, Tao Li, and Yihong Gong. Comparative document summarization via discriminative sentence selection. *ACM Trans. Knowl. Discov. Data*, 7(1):2:1–2:18, March 2013.

[114] Hongning Wang, Yue Lu, and Chengxiang Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *SIGKDD*, pages 783–792, New York, NY, USA, 2010. ACM.

[115] Xuerui Wang and Andrew McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 424–433, New York, NY, USA, 2006. ACM.

[116] René Witte and Sabine Bergler. Next-Generation Summarization: Contrastive, Focused, and Update Summaries. In *International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, Borovets, Bulgaria, September 27–29 2007.

[117] Fang Wu and BernardoA. Huberman. How public opinion forms. In Christos Papadimitriou and Shuzhong Zhang, editors, *Internet and Network Economics*, volume 5385 of *Lecture Notes in Computer Science*, pages 334–341. Springer Berlin Heidelberg, 2008.

[118] Mingli Wu, Wenjie Li, Qin Lu, and Kam-Fai Wong. Event-based summarization using time features. In *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '07, pages 563–574, Berlin, Heidelberg, 2007. Springer-Verlag.

[119] Erjia Yan and Ying Ding. Weighted citation: An indicator of an article's prestige. *Journal of the American Society for Information Science and Technology*, 61(8):1635–1643, 2010.

[120] Rui Yan, Xiaojun Wan, Mirella Lapata, Wayne Xin Zhao, Pu-Jen Cheng, and Xiaoming Li. Visualizing timelines: Evolutionary summarization via iterative reinforcement between text and image streams. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 275–284, New York, NY, USA, 2012. ACM.

[121] Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *SIGIR*, pages 745–754, New York, NY, USA, 2011. ACM.

[122] Jung-Yeon Yang, Jaeseok Myung, and Sang-goo Lee. The method for a summarization of product reviews using the user's opinion. In *Proceedings of the 2009 International Conference on Information, Process, and Knowledge Management*, EKNOW '09, pages 84–89, Washington, DC, USA, 2009. IEEE Computer Society.

[123] Y. Yue and T. Joachims. Predicting diverse subsets using structural svms. In *ICML*, 2008.

[124] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *J. Comput. Syst. Sci.*, 78(5):1538–1556, September 2012.

[125] Yisong Yue and T. Joachims. Predicting diverse subsets using structural SVMs. In *ICML*, pages 271–278, 2008.

[126] ChengXiang Zhai, Atulya Velivelli, and Bei Yu. A cross-collection mixture model for comparative text mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 743–748, New York, NY, USA, 2004. ACM.

[127] Jiaming Zhan, Han Tong Loh, and Ying Liu. Gather customer concerns from online product reviews - a text summarization approach. *Expert Syst. Appl.*, 36(2):2107–2115, March 2009.

[128] Richong Zhang and Thomas T. Tran. Probabilistic modeling of user-generated reviews. In *Web Intelligence*, pages 171–175, 2010.

[129] Zhu Zhang and Balaji Varadarajan. Utility scoring of product reviews. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 51–57, New York, NY, USA, 2006. ACM.

[130] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 43–50, New York, NY, USA, 2006. ACM.