High Compression Rate Text Summarization

by

Satchuthananthavale Rasiah Kuhan Branavan

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author . . . .
Department of Electrical Engineering and Computer Science
April 9, 2008

Certified by . . . .
Regina Barzilay
Associate Professor
Thesis Supervisor

Accepted by . . . .
Terry P. Orlando
Chairman, Department Committee on Graduate Students

High Compression Rate Text Summarization

by

Satchuthananthavale Rasiah Kuhan Branavan

Submitted to the Department of Electrical Engineering and Computer Science
on April 9, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis focuses on methods for condensing large documents into highly concise summaries, achieving compression rates on par with human writers. While the need for such summaries in the current age of information overload is increasing, the desired compression rate has thus far been beyond the reach of automatic summarization systems.

The potency of our summarization methods is due to their in-depth modelling of document content in a probabilistic framework. We explore two types of document representation that capture orthogonal aspects of text content. The first represents the semantic properties mentioned in a document in a hierarchical Bayesian model. This method is used to summarize thousands of consumer reviews by identifying the product properties mentioned by multiple reviewers. The second representation captures discourse properties, modelling the connections between different segments of a document. This discriminatively trained model is employed to generate tables of contents for books and lecture transcripts.

The summarization methods presented here have been incorporated into large-scale practical systems that help users effectively access information online.

Thesis Supervisor: Regina Barzilay
Title: Associate Professor

Acknowledgments

I'm grateful to my advisor Regina Barzilay whose guidance and assistance was crucial in the completion of this thesis. Her energy and involvement was pivotal in the work presented here seeing the light of day.

Many are those who contributed to the papers on which this thesis is based. Chief among them are my collaborators Pawan Deshpande, Harr Chen and Jacob Eisenstein to whom I express my gratitude. I would also like to thank Michael Collins, Benjamin Snyder, Yoong Keok Lee, Daniel Roy, Luke Zettlemoyer, Terry Koo, Zoran Dzunic, Erong Chen and Cristina Sauper whose ideas, feedback, and assistance was critical to this work. I would also like to acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0415865).

I'm also indebted to Indra Dayawanse of the University of Moratuwa, Samanthie Gunasekara, Ajit Samaranayake and Hemantha Jayawardana of Millennium IT, and Saman Amarasinghe of MIT all of who along with my family were instrumental in my returning from industry to undertake graduate study.

I would like to dedicate this thesis to my wonderful family: My parents and sisters – from whom I've learnt many things about life including the process of scientific thought – Vimaladevi, Satchuthananthavale, Vithiyani and Imasala; my brother-in-law, nephew and niece, Kumaradevan, Kulaghan and Kauline; and my many uncles, aunts and cousins.

Bibliographic Notes

Portions of this thesis are based on the following papers:

“Learning Document-Level Semantic Properties from Free-text Annotations” by S.R.K. Branavan, Harr Chen, Jacob Eisenstein, Regina Barzilay. This paper will appear in the *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics (ACL)*.

“Generating a Table-of-Contents” by S.R.K. Branavan, Pawan Deshpande, Regina Barzilay. This paper appeared in the *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Contents

List of Figures

List of Tables

அரும்பயன் ஆயும் அறிவினார் சொல்லார்
பெரும்பயன் இல்லாத சொல்.

திருக்குறள், 198ம் குறள்.

The wise who weigh the worth of every utterance,
Speak only words of significance.

Thirukkural, 198th verse

Chapter 1

Introduction

"To get the right word in the right place is a rare achievement. To condense the diffused light of a page of thought into the luminous flash of a single sentence, is worthy to rank as a prize composition just by itself... Anybody can have ideas – the difficulty is to express them without squandering a quire of paper on an idea that ought to be reduced to one glittering paragraph."

– Mark Twain

While few may match Twain's eloquence, people are routinely able to present complex information as very concise summaries. We encounter particularly useful examples of such synopses on a daily basis: news headlines and summaries, tables of contents, and book extracts among many others. An impressive characteristic of these summaries is their length compared to the original material – for example, a typical table of contents might be less than $1/100^{\text{th}}$ the size in words of the book in question. This brevity is key in making such summaries helpful to users, especially given the present-day glut of information. Today, all of these synopses are created by people. However, such manual compilation becomes impractical with the advent of electronic publication, and its accessibility to an ever growing authorship. Human summarizers simply would not be able to keep up with the sheer volume of new material and the rate at which it is produced.

The problem of automatic summarization has eluded artificial intelligence researchers since the 1950's [24]. In the last decade, language technology has matured significantly and a number of practical systems have been developed [13, 28, 6, 1, 22], mostly in the domain of news summarization. By selectively extracting sentences or clauses from a document, these methods produce summaries that are around 10% the length of the original text. In other words, the systems are able to achieve a compression rate of 10%. Despite these initial successes, high-compression rate summarization on par with human abilities is still well beyond the reach of existing automatic systems. The relatively coarse units of extraction, combined with the noise inherent in the selection criteria, prevent these algorithms from scaling to new applications.

Extraction at the level of sentences, and the noise inherent in selection

In this thesis, we investigate statistical models for high compression rate summarizers. These methods are particularly effective for processing long documents, such as books and large collections of documents. We are interested in achieving compression rates in the region of 1% to 0.1%, which are currently beyond the reach of automatic methods. Developing this new class of text-processing algorithms requires some significant changes to summarization technology:

- **Moving from extraction towards abstraction** Human editors rarely create summaries by simply extracting sentences from the original document. On the contrary, by abstracting the information and rewriting it concisely, they are able to condense the essential content of a long document into a few sentences. The inability of current domain-independent methods to perform such abstraction is a key hurdle to improving their compression rates. We wish to overcome this limitation by developing summarization methods that are capable of abstraction by modelling the relationships between a document's semantic content and lexical realization.

- **Collective content selection** When summarizing a large document using only a few phrases, one needs to ensure that the summary addresses all the key topics in the text. This requires that the information content for the summary be identified in a coordinated fashion – to avoid duplicating or omitting key topics. Nevertheless, most existing approaches make decisions about each unit of information in isolation, suffering the consequent loss of summary quality. Therefore new algorithms are required that are able to make concerted global decisions in identifying the summary content.

- **Training in the presence of incomplete training data** Supervised summarization methods are most often trained using large, manually annotated corpora. Each sentence of a document in such a corpus would be marked to indicate if it should be part of the summary or not. Creating the consistent and high quality annotations required by these methods is very time-consuming – even when the documents are short. In the case of long documents and large document collections, acquiring such professional annotation becomes prohibitively expensive. Thus, we are interested in developing methods that are able to leverage freely available partial annotations created by lay users for other purposes.

The potency of the summarization methods developed in this thesis is due to their in-depth modelling of document content. We explore in particular two types of document representation which capture orthogonal aspects of text content.

The first method operates on collections of documents that are partially annotated with free-form phrases indicative of the semantic content of the text. Three different but synergistic connections are present in such documents. First is the obvious link between the lexical realization of the text and its underlying semantic content. Second is the connection between the content of the text and the free-text annotations. Due to the inconsistent and partial nature of the annotations, this relationship is somewhat tenuous. The final link is between the different paraphrases used to indicate the same underlying property in the free-text annotations of all the documents. By modelling these relationships in an integrated fashion, our method is able to make good use of the inconsistent and incomplete annotations and achieve significant results in practical applications.

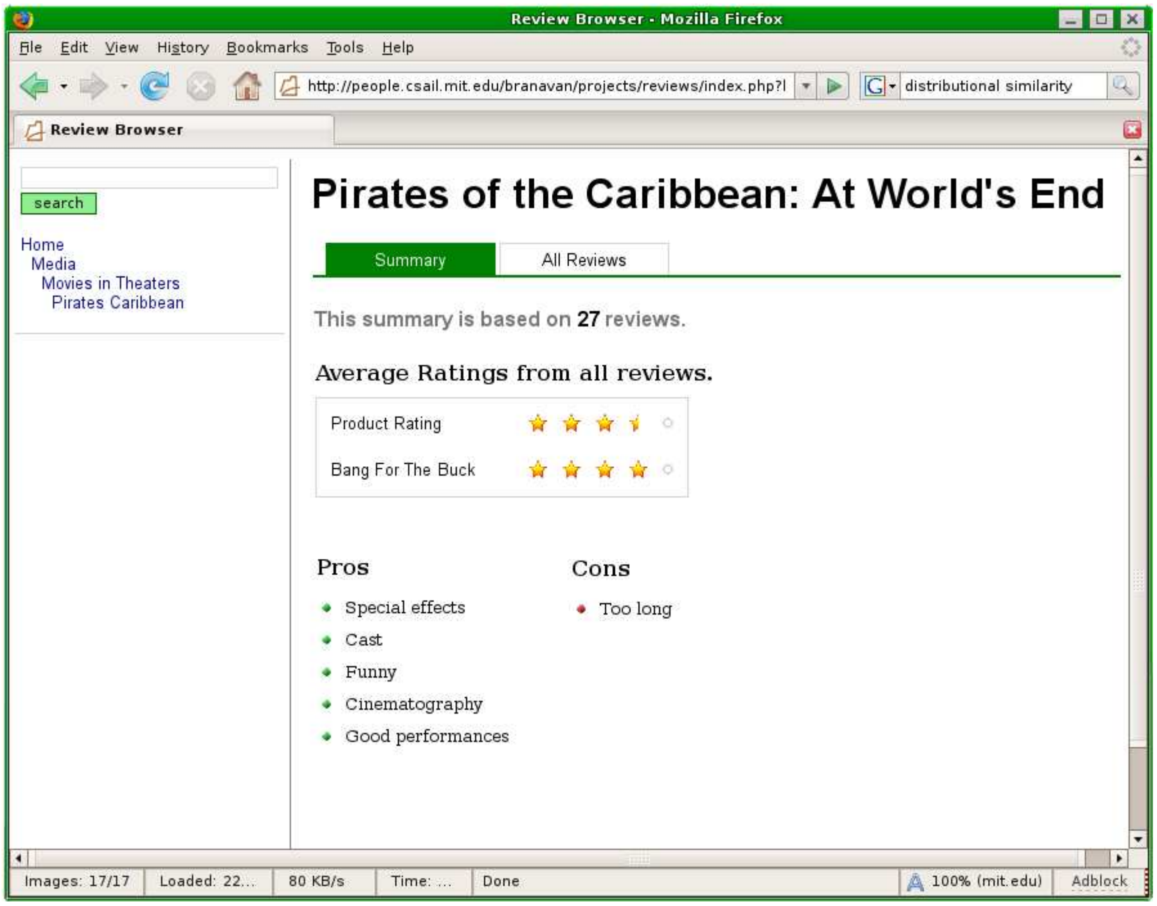
The second method is applied to the problem of summarizing long documents into tables of contents. A title in a hierarchical table of contents indicates the material presented in the corresponding section. Equally important, the title differentiates the section from its neighbours while linking it to its parent chapter. These relationships are crucial to this summarization task, and our method draws its strength from explicitly modelling them, thereby retaining key topics of the text and avoiding redundancy. As a consequence of modelling these global relationships, the space of possible summaries the model has to explore becomes exponentially large. Traversing this space naively in search of the best summary is intractable. Our method overcomes this problem by incrementally constructing its output in a manner that explores only the promising parts of the search space.

Beyond theoretical interest, these algorithms have been integrated into practical systems accessible to the general public over the internet. In both cases, the methods are used to help users access the large amounts of information available on those systems in a smart manner. The first method is applied to process a large collection of on-line consumer reviews of products and services. The system generates a concise list of a product's properties mentioned by users in the reviews. By representing a product using short phrases indicating its properties, the system allows readers to compare products and make decisions without having to read through numerous reviews. A screenshot of this system is shown in figure 1-1. The second algorithm is incorporated into the MIT online lecture browser. This website allows users to access video and text transcriptions of lectures in an interactive manner. Our model summarizes the transcriptions into tables of contents – allowing users to navigate through and quickly access the material. Figure 1-2 shows two sections of the table of contents generated by our system for an undergraduate textbook.

1.1 Summarizing Collections of Documents

Our first algorithm addresses the problem of summarizing a collection of documents by a list of semantic properties mentioned in the documents. For example, we may want to summarize multiple consumer reviews of a single product into a list of the product's properties mentioned by the authors. We are interested in automatically learning to produce such summaries based on partial free-text annotations provided for other purposes by the documents' authors.

To do so, we have to overcome two significant challenges. Firstly, authors often use different wording to express the same content - thus many of the free-text annotations would be paraphrasings of each other. This is one of the relationships within the document that


Review Browser - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://people.csail.mit.edu/branavan/projects/reviews/index.php?l
distributional similarity
Review Browser
search
Home
Media
Movies in Theaters
Pirates Caribbean
Summary
All Reviews
Images: 17/17
Loaded: 22...
80 KB/s
Time: ...
Done
100% (mit.edu)
Adblock

Pirates of the Caribbean: At World's End

This summary is based on **27** reviews.

Average Ratings from all reviews.

Product Rating	★★★½
Bang For The Buck	★★★★

Pros
- Special effects
- Cast
- Funny
- Cinematography
- Good performances

Cons
- Too long

Figure 1-1: Screenshot of the review browser system incorporating our method for summarizing large collections of documents. Accessible over the web, the system indexes over half a million consumer reviews on 50,000 different products.

- dictionary operations
  - direct address table
  - computer memory
    - worst case running time
    - searching for a given set
  - hash functions
    - real numbers and a hash function
    - creating a set of keys
    - hash functions and the number of keys
  - hash table
    - hash function
    - hash function
    - double hashing
    - hash table with load factor

- finding a shortest path
    - warshall algorithm
  - developing a shortest paths problem
    - characterizing the shortest paths and vertices
    - consisting of a shortest path from the minimum weight edges
    - taking a shortest path weights and matrices
    - computing the matrix product
  - floyd warshall algorithm to
    - vertices on shortest paths and algorithms
    - formulation of a shortest path estimates and recursive and observations
    - recurrence for the following procedure and values
    - constructing shortest path weights
    - transitive closure of a directed graph
  - sparse graphs and shortest path weights
    - showing the shortest path weights
    - paths from a source vertex
    - using the bellman ford algorithm

Figure 1-2: Sections of the table of contents generated by our method for the textbook *Introduction to Algorithms*.

we wish to explicitly model in our algorithm. However, since these annotations are often multi-word phrases, existing method of identifying synonyms are inadequate for the task. Secondly, and equally problematic, authors often mention a property either only in the text or in the annotations but not in both. Thus for example, we cannot directly use these phrases as label annotations to train a supervised classification algorithm. Any effective approach for this task will have to be able to work with these incomplete and noisy annotations.

Our approach to the task of predicting the properties mentioned in documents directly addresses both of these challenges. We handle the noisy and incomplete labels by modelling the task in a Bayesian generative framework with explicit structures to account for the label noise. In addition, we use the distributional and lexical properties of the free-text annotations to compute a semantic clustering over them. This allows us to effectively replace multiple free-text annotations with a single semantic annotation, thus practically reducing annotation noise. Furthermore, the content of the documents associated with the annotation phrases gives us additional information about the similarity between these phrases. We capture all of these relationships within a single generative model, allowing clustering and prediction tasks to positively influence each other. As evidenced by the results, this is an effective strategy.

1.2 Summarizing Long Documents

A table of contents is a particularly useful type of summary for long documents, helping readers to effectively access and navigate through large amounts of content. Our second model addresses the problem of generating a table of contents for a long text document where a hierarchical structure of chapters and sections is already available. While this may seem restrictive, conversion of documents in other formats – such as voice recordings of lectures – to text and hierarchical segmentation can be done using existing methods from the literature.

Specifically, given a document containing hierarchically structured segments, we wish to generate a tree of titles where each title is the summary of a corresponding text segment. As mentioned before, the titles are strongly interrelated – while they need to clearly differentiate their text segment from the rest of the document, titles of large chapters need to be more generic than those of smaller sections. This introduces a few significant challenges to the task. Firstly, the whole tree of titles needs to be generated in a coordinated fashion while considering the global relationships between titles. This leads to the second challenge – the global relationships makes the search space of summaries exponentially large, requiring new methods for tractable learning and inference.

We overcome these challenges with a hierarchical discriminative approach which is able to model a range of lexical and positional features of the titles, local constraints within sections, and global dependencies across the titles in the tree. This allows the model to produce coherent tables of contents composed of titles that distinguish each section from the rest of the text. We address the problem of tractability by decomposing the model into two components, one for the local dependencies, and one for the global. These nonetheless function in a joint fashion, with the global component which constructs the table of contents operating on lists of candidate titles produced by the local component. In addition, both titles and tables of contents are generated in an incremental manner, thus further improving tractability of the algorithm by considering only promising parts of the search space.

1.3 Contributions

The key contributions of this work are three-fold. From an algorithmic standpoint, we introduce novel methods for tractable global inference for the tasks of summarizing long documents and large collections of documents. These algorithms directly address the challenges inherent to the tasks.

Furthermore, by applying our methods to real world datasets, in the first case, we empirically show the feasibility of using free-text annotation of documents as a supervision signal for the task of property prediction, opening up the possibility of further applications for such annotations. In the case of generating tables of contents, our results confirm the advantages of joining modelling the local and global inference tasks, and the benefits of global constraints to summary coherence and relevance.

Finally, both methods have been incorporated into practical systems accessible through the internet enabling novel ways of accessing large collections of information for a wide variety of users.

1.4 Thesis Overview

The remainder of the thesis is organized as follows: In the next chapter we discuss related work in the areas of Summarization, Property Prediction, and Table of Contents Generation. In chapter 3 we provide formal descriptions of multiple models for learning using free-text annotations to summarize large collections of documents, and we give empirical results showing the practical value of the algorithms. In chapter 4 we describe our method of summarizing long documents into tables of contents, and provide empirical results. Chapter 5 concludes with the main ideas and contributions of this work, along with potential directions for future research.

Chapter 2

Related Work

Our work focuses on the conversion of long documents and collections of documents into succinct and structured synopses. The two specific tasks we look at require very different types of summaries – in the first, we summarize a collection of documents into a single list of semantic properties, and in the second, long documents such as books or lecture transcripts are condensed into tables of contents. While the underlying theme is of structured summarization, the two require fundamentally different approaches, and derive from different streams of related work. In the following sections we describe the prior work relevant to these two tasks.

2.1 Predicting Document Properties

Traditionally, the task of identifying the properties or topics mentioned in a document has been cast as an extraction or classification problem. The extractive methods [20, 26] use machine learning approaches or hand crafted rules to identify segments of text indicative of the document's topics. Classification methods such as [21] focus on identifying the sentences of a review which are indicative of the pros and cons of the product. Both of these approaches treat the extracted properties simply as segments of text and do not attempt to identify the semantic relationships between them.

There has also been work on applying Bayesian graphical models to learn the topics present in a document. Methods such as LDA [5] and CTM [3] associate each word in a document with one of a predefined number of latent topics. These topics maybe considered as proxies to the semantic properties mentioned in the document. However, since these models are unsupervised, they afford no method of linking the latent topics to external observed representations of the properties of interest.

Recent work [4, 30] has extended the latent topic framework to model a document's labels or numerical rankings jointly with its words. This allows these methods to use the topic modelling framework for label prediction and aspect ranking tasks.

Our work extends the latent topic framework to jointly model document words and a structure on the document labels. This allows us to compute a semantic clustering of free-text annotations associated with the documents. The document topics jointly learnt by the model are linked to these clusters. The clusters themselves are representative of the underlying semantic property. Thus in contrast to the methods mentioned above, our model is able to predict the semantic properties explicitly or implicitly mentioned in a document

In this chapter, we present background on the methods mentioned above.

2.1.1 Extractive Methods

In this section we present two different approaches to extracting a product's properties from consumer reviews. The first by Hu and Liu [20] also attempts to identify the semantic polarity of review sentences, while the second, OPINE [26], attempts to find opinion phrases about a product's properties and to identify the polarity of the phrases.

Hu and Liu [20] propose a method for the task of summarizing reviews by extracting product properties and identifying if the reviewer's opinion about the properties were positive or negative. The method first extracts the set of product properties from the reviews by finding high frequency nouns and noun phrases using *association mining*. Noun phrases that are not likely to be product properties are then removed through *compactness pruning* and *redundancy pruning*. These two steps result in the set of properties which the method uses for producing the summaries. Then, all review sentences containing one or more of the product properties are identified, and any adjectives from these sentences are extracted as *opinion words*. A set of 30 hand annotated seed adjectives and WordNet are then used to identify the polarity of the opinion words - i.e. whether the adjective has a positive or negative connotation. Finally, given an unseen review, a summary is produced by extracting all property phrases present verbatim in the document. The polarity of the sentence expressing the property is then identified as the dominant polarity of any adjectives present. In case of ties, the polarity of the adjective modifying the property noun or phrase is used. The number of sentences found of each polarity are given as an indicator of the opinions about each property.

This method was tested on consumer reviews collected from Amazon.com and Cnet.com for five different products. The authors evaluated property extraction, opinion sentence extraction and sentence polarity identification against manually annotations. The property extraction results are compared against those of the publicly available term extraction and indexing system FASTR [1], showing better recall and precision on all five products.

Popescu and Etzioni's OPINE [26] is a three step extractive process of identifying product properties, identifying opinions regarding the properties, and finally finding the polarity of the opinions. Their method first extracts frequent noun phrases from the review documents. The *Point-wise Mutual Information* between each phrase and automatically constructed *meronymy discriminators* is computed using web search engine hit counts. Meronymy discriminators are phrases such as "phone has" based on the type of product. A product's properties are distinguished from its parts using WordNet and morphological cues. In the second step, potential opinion phrases are identified by applying a set of ten hand crafted extraction rules on the syntactic dependencies produced from the document by the MINIPAR parser. The phrases whose head word has a positive or negative semantic orientation are retained as actual opinion phrases. The method uses relaxation labelling to identify the semantic orientation of words.

In the two tasks of extracting opinion phrases and identifying the polarity of opinion words, OPINE was shown to have better precision and recall as compared to Hu and Liu's system. Against PMI [32], it has better precision at the cost of recall. In opinion phrase polarity identification, OPINE has better precision and worse recall compared to both the other methods.

[1]http://www.limsi.fr/Individu/jacquemi/FASTR

In contrast to our approach, the above two extractive methods are only able to identify product properties explicitly mentioned using noun phrases in the document. Furthermore, unlike in our method, each extracted noun phrase is dealt with in isolation - no attempt is made to identify different phrases that allude to the same underlying product property. However, the output produced by either of these algorithms can be used as training data for our model.

2.1.2 Classification

Kim and Hovy focus on the problem of identifying sentences containing opinions about a product's properties and the reasons for those opinions from a review. They use consumer reviews containing pros and cons phrases in addition to the text description as training data. As a first step, their method checks each sentence in the review for the presence of the pros and cons phrases it was associated with. Sentences are labelled as "pro", "con" or "neither" on this basis. This labelled data is then used to train two maximum entropy binary classifiers - the first to identify sentences containing opinions, and the second to differentiate such sentences as "pro" or "con". Unigrams, bigrams and trigrams, sentence position features, and pre-selected opinion-bearing words are provided to the classifiers as features. Opinion bearing words are selected using WordNet, and by analysing opinion pieces such as letters and editorials, and factual reports such as news and events from a large news corpus.

The method was tested on data collected from two websites: Epinions.com and Complaints.com. On both the tasks of identifying opinion sentences and differentiating pros and cons sentences, the approach is shown to have better accuracy compared to a majority baseline.

In common with the extractive approaches, this method also does not attempt to identify paraphrases of product properties. In addition, with the focus of finding the reasons for opinions, the method extracts complete sentences and not just the properties we are interested in.

2.1.3 Bayesian Topic Modelling

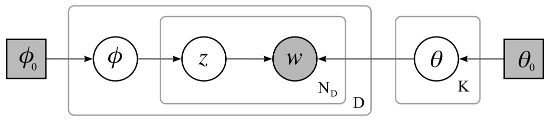
Latent Dirichlet Allocation

One of the recent Bayesian approaches to modelling documents is Latent Dirichlet Allocation (LDA) [5] by Blei et al. Their method is based on the idea that each word in a document belongs to one of several *topics* - i.e. that each document is generated from a mixture of topics. The proportions of the topics for each document are considered to be distributed as a latent Dirichlet random variable. Since the words are associated with topics, each topic also has a distinct distribution over the words. By estimating the word distributions for each topic, and the topic distributions for each document, the model reduces the documents to a low dimensional representation over topics. Figure 2-1 shows the details of this model. The parameters of LDA are intractable to compute in general, but can be estimated using a variety of approximate inference methods including Markov chain Monte Carlo, Laplace approximation and variational approximation. Blei et al. derive a convexity-based variational inference method in their work.

By training the model on 16,000 documents from the TREC AP corpus [17], the authors qualitatively show that the word distributions seem to capture some of the underlying topics of the documents. LDA was also shown to perform better (i.e. has lower perplexity) than comparable latent variable models such as pLSI [19] on tasks such as document modelling and collaborative filtering. The authors also test an SVM classifier trained on the lower dimensional representation produced by LDA against the same classifier trained on the full document on a binary document classification task. The evaluations show that the classifier trained on LDA's output performs better in almost all cases, with little loss of accuracy in the rest.

Correlated Topic Models

One of the shortcomings of topic modelling as discussed above is that the topics are assumed to be independent of each other. This is obviously not true in general - for example, a single review of a restaurant is highly unlikely imply that the food is both good and bad. Or more subtly, a restaurant with bad staff is unlikely to have good service. Given



N_d	–	Number of words in document d
ϕ_0	–	Dirichlet prior on topic model ϕ
ϕ	–	document word topic model
z	–	word topic assignment
θ	–	language models of each topic
w	–	document words

$$N_d \sim \text{Poission}(\xi)$$

$$\phi_d \sim \text{Dirichlet}(\phi_0)$$

$$z_{n,d} \sim \text{Multinomial}(\phi_d)$$

$$\theta_z \sim \text{Dirichlet}(\theta_0)$$

$$w_{n,d} \sim \text{Multinomial}(\theta_z)$$

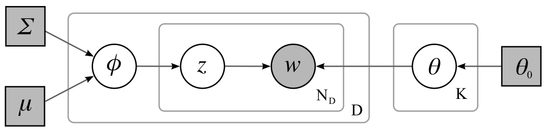
Figure 2-1: Plate diagram and sampling equations for the Latent Dirichlet Allocation model.

that the link between topics can potentially be very strong, learning these correlations can potentially be beneficial - resulting in better topic models. The Correlated Topic Model (CTM) [3] of Blei et al. extends LDA based on this intuition. In LDA, the assumption of topic independence is made in modelling the proportions of topics as being drawn from a Dirichlet distribution. CTM replaces the Dirichlet by a logistic normal distribution whose covariance matrix models the correlation between the topics. While the logistic normal increases the expressivity of the model and allows it to learn the correlation between topics, it has the significant disadvantage of not being conjugate to the multinomial distribution - thus complicating inference. Blei et al. describe a fast variational inference algorithm for approximate inference which overcomes this problem.

The authors compare CTM against LDA by computing the log probabilities of the resulting models on held-out data. A better model would assign a higher probability to the unseen documents. Testing with ten-fold cross validation on a collection of 1,452 documents from the JSTOR [2] on-line archive, CTM was shown to achieve higher log-probability on held-out documents compared to LDA.

Both LDA and CTM described above focus only on modelling the words of a document as generated from a mixture of topics. While they are able to reduce a document to a low dimensional representation over these topics, the topics themselves in these models

[2]www.jstor.org



Σ, μ	–	Logistic normal prior on ϕ
ϕ	–	document word topic model
z	–	word topic assignment
θ	–	language models of each topic
w	–	document words

$$\eta_d \sim \text{Multivariate Normal}(\mu, \Sigma)$$
$$\phi_d = \frac{e^{\eta}}{\sum_i e^{\eta_i}}$$
$$\theta_z \sim \text{Dirichlet}(\theta_0)$$
$$z_{n,d} \sim \text{Multinomial}(\phi_d)$$
$$w_{n,d} \sim \text{Multinomial}(\theta_z)$$

Figure 2-2: Plate diagram and sampling equations for the Correlated Topic model.

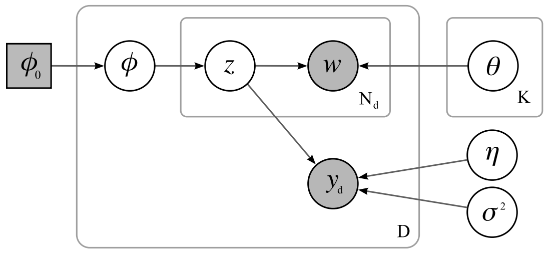
are abstract, being defined simply by distributions over words. Neither of the models is designed to learn an association between the topics and other features of a document such as the product properties mentioned in it. As such, they cannot directly be applied to our task.

Supervised Topic Model

Supervised latent Dirichlet allocation (sLDA) [4], a statistical model of labelled documents, is an attempt at overcoming the above limitations of LDA and CTM. The goal here is to infer latent topics that are predictive of the required *response*. In general, the response may be a label or numerical ranking associated with the document.

sLDA models the words of a document in the same manner as LDA. In addition, when the response associated with the documents is an unconstrained real value, sLDA models this response as being drawn from a normal linear model. By allowing the response to be drawn from a *generalized linear model* [25] the method is also shown to be capable of handling other types of responses such as positive real values, ordered or unordered labels and non-negative integers. Blei and McAuliffe describe a variational expectation-maximization procedure for approximate maximum-likelihood estimation of the model's parameters.

sLDA was tested on two tasks. The first is to identify the ranking given to a movie by a review author in terms of the number of stars. The second task is to predict web page popularity on Digg.com in terms of the number of “diggs” a page gets from users. In both tasks, sLDA was shown to be better than applying linear regression on the topics identified in a document by unsupervised LDA. sLDA was also compared against *lasso* a L_1-regularized least-squares regression method and shown to produce moderate improvements.



ϕ_0	–	Dirichlet prior on topic model ϕ
ϕ	–	document word topic model
z	–	word topic assignment
θ	–	language models of each topic
w	–	document words
y_d	–	document response variable

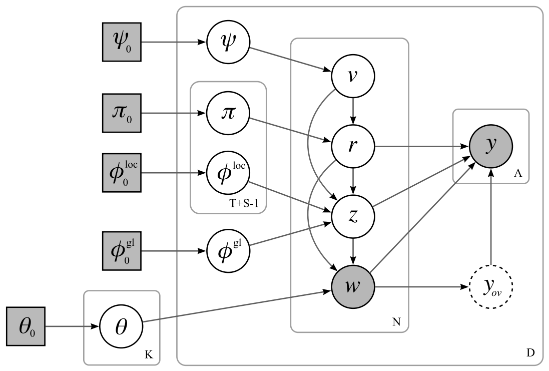
$$\begin{aligned}
\phi_d &\sim \text{Dirichlet}(\phi_0) \\
\theta_z &\sim \text{Dirichlet}(\theta_0) \\
z_{n,d} &\sim \text{Multinomial}(\phi_d) \\
w_{n,d} &\sim \text{Multinomial}(\theta_z) \\
y_d &\sim \mathcal{N}(\eta^{\mathsf{T}}\overline{z}_d, \sigma^2) \\
\overline{z}_d &= \frac{1}{N}\sum_{n=1}^{N} z_{n,d}
\end{aligned}$$

Figure 2-3: Plate diagram and sampling equations for the Supervised Latent Dirichlet Allocation model for an unconstrained real-valued response.

sLDA extends the topic modelling framework to include the annotations associated with a document. While similar in this respect, our work is distinguished in modelling a structure over the annotations in addition to the annotations themselves. This allows our model to cluster free-text annotations into semantic classes, thereby learning document topics representative of these classes.

Multi-Aspect Sentiment Model

In the Multi-Aspect Sentiment model [29] (MAS), Titov and McDonald focus on the task of identifying textual mentions in a document that are relevant to a rateable aspect. MAS, shown in figure 2-4 is a joint statistical model of sentiment ratings and document text, and is an extension of the Multi-Grain LDA (MG-LDA) model [30] As with MG-LDA, MAS models the words of a document as being generated either from a mixture of topics global to the given document, or from a mixture of topics local to the neighbourhood of the word. Titov and McDonald define this neighbourhood as a sliding window covering T adjacent sentences in the document, and thus a single word may be generated by one of many windows. The sentiment ratings associated with the document are modelled as being drawn from a log-linear distribution parameterized by the document words, the word topics, and the topic distribution (local or global) from which the words were drawn.



v	–	Sliding window defining a word's neighbourhood
ψ	–	categorical distribution over windows v
ϕ^{gl}	–	topic model global to document
ϕ_v^{loc}	–	topic model local to window v
r	–	random variable identifying ϕ_v^{loc} or ϕ^{gl} as source of word
π	–	distribution over r
w	–	document words
z	–	word topic assignment
θ	–	language models of each topic
y_{ov}	–	overall sentiment rating of document (not directly modelled)
y_d	–	individual aspect ratings associated with document d

$$\psi_d \sim \text{Dirichlet}(\psi_0)$$
$$v_{d,w} \sim \text{Categorical}(\psi_d)$$
$$\pi_{d,v} \sim \text{Beta}(\pi_0)$$
$$r_{d,w} \sim \text{Bernoulli}(\pi_{d,v})$$
$$\phi_{d,v}^{loc} \sim \text{Dirichlet}(\phi_0^{loc})$$
$$\phi^{gl} \sim \text{Dirichlet}(\phi_0^{gl})$$
$$z \sim \begin{cases} \text{Multinomial}(\phi^{gl}) & \text{if } r = gl \\ \text{Multinomial}(\phi_{d,v}^{loc}) & \text{if } r = loc \end{cases}$$
$$\theta_{r,z} \sim \text{Dirichlet}(\theta_0)$$
$$w \sim \text{Multinomial}(\theta_{r,z})$$
$$y \sim P(y \mid \mathbf{w}, \mathbf{r}, \mathbf{z})$$

where $P(y \mid \mathbf{w}, \mathbf{r}, \mathbf{z})$ is a log-linear distribution over y dependent of features derived from the words, their topics, and the distributions from which the topics were drawn.

Figure 2-4: Plate diagram and sampling equations for the Multi-Aspect Sentiment model.

The authors use Gibbs sampling to estimate the parameters of the MG-LDA part of the model, and stochastic gradient ascent for the parameters of the log-linear distribution on the sentiment ratings. Testing on 10,000 reviews downloaded from the TripAdvisor.com website, MAS is shown to infer topics corresponding to the aspects of the sentiment ratings. The model is also able to accurately identify the text fragments of the documents relevant to a given aspect.

MAS uses the estimated topic distributions associated with the aspect ratings to extract text segments in documents that are relevant to a given aspect. Unlike our model, MAS does not attempt to learn the semantic relationships between these phrases. However, as with the extractive methods mentioned earlier, the phrases identified by MAS can be used as training information for our model.

2.2 Summarizing Documents into Tables of Contents

While a variety of approaches have been developed for text summarization, much of this work has been on processing short documents such as news articles. These methods use approaches such as shortening individual sentences [22], and as such are unable to achieve the high compression rates required for the long documents that we wish to summarize. Longer documents have typically been handled using highly domain specific methods [13, 28], where strong assumptions could be made regarding the structure of the input. Alternatively, approaches such as [6, 1] take advantage of the topic structure of a text to produce representative summaries while processing long documents.

In this section we briefly describe the methods mentioned above.

2.2.1 Sentence Compression

Knight and Marcu [22] consider two different approaches to the summarization of single sentences or *sentence compression* - the first approach using a probabilistic noisy-channel model, and the second using a decision based deterministic model. In the noisy channel framework, the long sentence to be compressed is considered to have been produced by the addition of noise to a short sentence. The method operates on the parse trees of the sentences in question rather than directly on the text, and attempts to find the most likely short sentence based on probabilities defined over the parse trees and over operations on the parse trees. In the second approach, the authors use four basic operations to "rewrite" the long sentence into a short one. Based on a set of training pairs of long and summary sentences, a decision tree is learnt for rewriting a given sentence.

While both these methods were shown to be successful, the sentence compression approach is unable to achieve the high compression rates required for our task.

2.2.2 Summarizing Long Documents

Elhadad and McKeown [13] describe a method for summarizing medical journal articles into a single short document containing information relevant to a given patient. Their method involves multiple domain and corpus specific steps where handcrafted rules or templates are used to identify relevant sentences and to extract information from them. This information is then used to generate a summary after filtering based on the patient's details.

In a similar vein, Teufel and Moens [28] make use of rhetorical structure specific to scientific articles to achieve high compression rates. The authors train a naïve Bayes classifier using documents annotated with a set of predefined rhetorical categories and with sentence salience information. In addition to standard lexical and positional features, the classifier is also provided with information specific to scientific articles such as the presence of citations. Once trained, the classifier is used to identify sentences to be extracted for the summary.

While they are able to handle long documents, both of these methods are highly domain specific, and in the first case corpus specific - severely restricting their general applicability.

2.2.3 Using Document Structure for Higher Compression Rates

Summarization by stitching together text fragments extracted from a long document as in the methods above can result in problems such as loss of coherence, loss of readability, and thematic under-representation [6]. Boguraev and Neff attempt to address some of these issues using discourse segmentation information. In addition to lexical features for sentence salience, information from an automatic topical segmentation algorithm and hand crafted rules are used to encourage the summary to be representative of all topics in a long document. The authors empirically show that adding segmentation information improves summary quality under certain conditions such as high compression rates.

Also making use of document structure, Angheluta et al. [1] use a layered topic segmentation method and term extraction to summarize a document into a structured table of contents. The authors use language specific heuristics to identify the main topic word or word group of each sentence. The distribution of these topic terms across the document's segments is used to locate it in a hierarchical structure, thus generating a table of contents.

While these methods, like our approach, make use of the document's structure, they select the summary term for each section of the document in isolation. In contrast, our algorithm is able to leverage the relationships between the summary items to enforce global constraints, thus reducing redundancy.

2.2.4 Title Generation

The task of generating titles for individual documents has seen extensive study. Banko et al. [2] view the task as analogous to statistical machine translation, and propose a model that jointly performs content selection and surface realization.

Taking an alternative approach, Dorr et al. [12] generate headlines for news articles by condensing the first sentence of the text. The use of the first sentence is based on the observation that it contains most of the words of the article's original headline. The sentence is trimmed by identifying and removing constituents from its parse tree using linguistically motivated heuristics.

Both of these methods are shown by their authors to be effective for generating headlines for newspaper reports. Besides being specific to news reports, these approaches focus on separately generating a single title for each individual article. Therefore, they do not need to contend with the issues unique to our task: the hierarchical generation of multiple titles, and the global relationships between those titles.

2.3 Summary

Prior work on predicting the properties or topics mentioned in a document have focused on either extracting text segments from the documents, or in supervised classification methods. The classification methods are based on features computed either over the text, or over some lower dimensional representation of the text such as that produced by latent topic models. All of these methods however treat the properties as independent labels or phrases. While our method also learns from phrase annotations associated with the documents, in contrast to the other approaches, it automatically learns a structure over the phrases - clustering them into semantically similar classes. By learning this clustering jointly with the property prediction task, our model leverages both sources of information showing improved performance on both tasks. In the following chapter we discuss this joint model in detail.

Chapter 3

Summarizing Reviews

3.1 Introduction

A central problem in language understanding is transforming raw text into structured representations. Learning-based approaches have dramatically increased the scope and robustness of automatic language processing, but they are typically dependent on large expert-annotated datasets, which are costly to produce. In this chapter, we show how novice-generated free-text annotations available online can be leveraged to automatically infer document-level semantic properties.

More concretely, we are interested in determining properties of consumer products and services from reviews. Often, such reviews are annotated with *keyphrase* lists of pros and cons. We would like to use these keyphrase lists as training labels, so that the properties of unannotated reviews can be predicted. However, novice-generated keyphrases lack consistency: the same underlying property may be expressed many ways, *e.g.*, “reasonably priced” and “a great bargain.” To take advantage of such noisy labels, a system must both uncover their hidden *clustering* into properties, and learn to predict these properties from review text.

This paper presents a model that attacks both problems simultaneously. We assume that both the review text and the selection of keyphrases are governed by the underlying hidden properties of the review. Each property indexes a language model, thus allowing reviews that incorporate the same property to share similar features. In addition, each pros/cons: *great nutritional value*

... combines it all: an amazing product, *quick and friendly service*, cleanliness, great nutrition ...

pros/cons: *a bit pricey, healthy*

... is an awesome place to go if you are health conscious. They have some really great low calorie dishes and they publish the calories and fat grams per serving.

Figure 3-1: Excerpts from online restaurant reviews with pros/cons phrase lists. Both reviews discuss healthiness, but use different keyphrases.

observed keyphrase is associated with a property; keyphrases that are associated with the same property should have similar distributional and surface features.

We link these two ideas in a joint hierarchical Bayesian model. Keyphrases are clustered based on their distributional and orthographic properties, and a hidden topic model is applied to the review text. Crucially, the keyphrase clusters and hidden document topics are linked, and inference is performed jointly. This increases the robustness of the keyphrase clustering, and ensures that the inferred hidden topics are indicative of salient semantic properties.

Our method is applied to a collection of reviews in two distinct categories: restaurants and cell phones. During training, lists of keyphrases are included as part of the reviews by the review authors. We then evaluate the ability of our model to predict review properties when the keyphrase list is hidden. Across a variety of evaluation scenarios, our algorithm consistently outperforms alternative strategies by a wide margin.

3.2 The Method

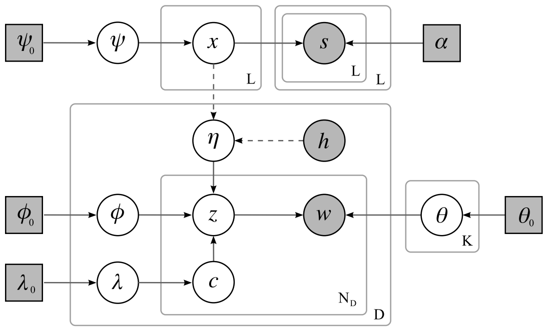
3.2.1 Problem Formulation

We formulate our problem as follows. We assume a dataset composed of documents with associated keyphrases. Each document may be marked with multiple keyphrases that express semantic properties. Across the entire collection, several keyphrases may express the same property. The keyphrases are also incomplete – review texts often express properties that are not mentioned in their keyphrases. At training time, our model has access to both text and keyphrases; at test time, the goal is to predict which properties a previously unseen document supports, and by extension, which keyphrases are applicable to it.

3.2.2 Document Property Model

Our approach leverages both keyphrase clustering and distributional analysis of the text in a joint, hierarchical Bayesian model. Keyphrases are drawn from a set of clusters; words in the documents are drawn from language models indexed by a set of topics, where the topics correspond to the keyphrase clusters. Crucially, we bias the assignment of hidden topics in the text to be similar to the topics represented by the keyphrases of the document, but we permit some words to be drawn from other topics not represented by the document's keyphrases. This flexibility in the coupling allows the model to learn effectively in the presence of incomplete keyphrase annotations, while still encouraging the keyphrase clustering to cohere with the topics supported by the document text. The plate diagram for our model is shown in Figure 3-2.

We train the model on documents annotated with keyphrases. During training, we learn a hidden topic model from the text; each topic is also associated with a cluster of keyphrases. At test time, we are presented with documents that do not contain keyphrase annotations. The hidden topic model of the review text is used to to determine the properties that a document as a whole supports. For each property, we compute the proportion of the document's words assigned to it. Properties with proportions above a set threshold (tuned on a development set) are predicted as being supported.



ψ –	keyphrase cluster model
x –	keyphrase cluster assignment
s –	keyphrase similarity values
h –	document keyphrases
η –	document keyphrase topics
λ –	probability of selecting η instead of ϕ
c –	selects between η and ϕ for word topics
ϕ –	document topic model
z –	word topic assignment
θ –	language models of each topic
w –	document words

$$\psi \sim \text{Dirichlet}(\psi_0)$$

$$x_\ell \sim \text{Multinomial}(\psi)$$

$$s_{\ell,\ell'} \sim \begin{cases} \text{Beta}(\alpha_=) & \text{if } x_\ell = x_{\ell'} \\ \text{Beta}(\alpha_{\neq}) & \text{otherwise} \end{cases}$$

$$\eta_d = [\eta_{d,1} \dots \eta_{d,K}]^{\mathrm{T}}$$

where

$$\eta_{d,k} \propto \begin{cases} 1 & \text{if } x_\ell = k \text{ for any } l \in h_d \\ 0 & \text{otherwise} \end{cases}$$

$$\lambda \sim \text{Beta}(\lambda_0)$$

$$c_{d,n} \sim \text{Bernoulli}(\lambda)$$

$$\phi_d \sim \text{Dirichlet}(\phi_0)$$

$$z_{d,n} \sim \begin{cases} \text{Multinomial}(\eta_d) & \text{if } c_{d,n} = 1 \\ \text{Multinomial}(\phi_d) & \text{otherwise} \end{cases}$$

$$\theta_k \sim \text{Dirichlet}(\theta_0)$$

$$w_{d,n} \sim \text{Multinomial}(\theta_{z_{d,n}})$$

Figure 3-2: The plate diagram for our model. Shaded circles denote observed variables, and squares denote hyper parameters. The dotted arrows indicate that η is constructed deterministically from $\mathbf{x}$ and $\mathbf{h}$.

Keyphrase Clustering

One of our goals is to cluster the keyphrases, such that each cluster corresponds to a well-defined document property. While our overall model is generative, we desire the freedom to use any arbitrary metric for keyphrase similarity. For this reason, we represent each distinct keyphrase as a vector of similarity scores computed over the set of observed keyphrases; these scores are represented by s in Figure 3-2. We then explicitly generate this similarity matrix, rather than the surface form of the keyphrase itself. Modelling similarity scores rather than keyphrase words affords us the flexibility of clustering the keyphrases using more than just their word distributions. We assume that similarity scores are conditionally independent given the keyphrase clustering. Models that make similar assumptions about the independence of related hidden variables have previously been shown to be successful [31]. In section 3.2.3 we describe a model which removes the need for this assumption by applying principal component analysis to the similarity matrix.

We compute the similarity between keyphrases using a linear interpolation of two metrics. The first is the cosine similarity between keyphrase word vectors. The second is based on the co-occurrence of keyphrases in the review texts themselves. While we chose these two metrics for their simplicity, our model is inherently capable of using other sources of similarity information. For a discussion of similarity metrics, see [23].

Document-level Distributional Analysis

Our analysis of the document text is based on probabilistic topic models such as LDA [5]. In the LDA framework, each word is generated from a language model that is indexed by the word's topic assignment. Thus, rather than identifying a single topic for a document, LDA identifies a distribution over topics.

Our word model operates similarly, identifying a topic for each word, written as z in Figure 3-2. However, where LDA learns a distribution over topics for each document, we deterministically construct a document-specific topic distribution from the clusters represented by the document's keyphrases – this is η in the figure. η assigns equal probability to all topics that are represented in the keyphrases, and zero probability to other topics.

Generating the word topics in this way ties together the keyphrase clustering and language models.

As noted above, sometimes properties are expressed in the text even when no related keyphrase is present. For this reason, we also construct another document specific topic distribution ϕ. The auxiliary variable c indicates whether a given word's topic is drawn from the set of keyphrase clusters, or from this topic distribution.

Generative Process

In this section, we describe the underlying generative process more formally.

First we consider the set of all keyphrases observed across the entire corpus, of which there are L. We draw a multinomial distribution ψ over the K keyphrase clusters from a symmetric Dirichlet prior ψ_0. Then for the ℓ^{th} keyphrase, a cluster assignment x_ℓ is drawn from the multinomial ψ. Finally, the similarity matrix $\mathbf{s} \in [0,1]^{L \times L}$ is constructed. Each entry $s_{\ell,\ell'}$ is drawn independently, depending on the cluster assignments x_ℓ and $x_{\ell'}$. Specifically, $s_{\ell,\ell'}$ is drawn from a Beta distribution with parameters $\alpha_=$ if $x_\ell = x_{\ell'}$ and $\alpha_\neq$ otherwise. The parameters $\alpha_=$ linearly bias $s_{\ell,\ell'}$ towards one (Beta$(\alpha_=) \equiv$ Beta$(2,1)$), and the parameters $\alpha_\neq$ linearly bias $s_{\ell,\ell'}$ towards zero (Beta$(\alpha_\neq) \equiv$ Beta$(1,2)$).

Next, the words in each of the D documents are generated. Document d has N_d words, and the topic for word $w_{d,n}$ is written as $z_{d,n}$. These latent topics are drawn either from the set of clusters represented by the document's keyphrases, or from the document's topic model ϕ_d. We deterministically construct a document-specific keyphrase topic model η, based on the keyphrase cluster assignments $\mathbf{x}$ and the observed keyphrases $\mathbf{h}$. The multinomial η_d assigns equal probability to each topic that is represented by a phrase in h_d, and zero probability to other topics.

As noted earlier, a document's text may support properties that are not mentioned in its observed keyphrases. For that reason, we draw a document topic multinomial ϕ_d from a symmetric Dirichlet prior ϕ_0. The binary auxiliary variable $c_{d,n}$ determines whether the word's topic is drawn from the keyphrase model η_d or the document topic model ϕ_d. $c_{d,n}$ is drawn from a weighted coin flip, with probability λ; λ is drawn from a Beta distribution with prior λ_0. We have $z_{d,n} \sim \eta_d$ if $c_{d,n} = 1$, and $z_{d,n} \sim \phi$ otherwise. Finally, the word $w_{d,n}$

$$
\begin{aligned}
p(x_\ell \mid \ldots) &\propto p(x_\ell \mid \psi) p(\mathbf{s} \mid x_\ell, \mathbf{x}_{-\ell}, \alpha) p(\mathbf{z} \mid \eta, \psi, \mathbf{c}) \\
&\propto p(x_\ell \mid \psi) \left[\prod_{\ell' \neq \ell} p(s_{\ell,\ell'} \mid x_\ell, x_{\ell'}, \alpha) \right] \left[\prod_d^D \prod_{c_{d,n}=1} p(z_{d,n} \mid \eta_d) \right] \\
&= \text{Multinomial}(x_\ell; \psi) \left[\prod_{\ell' \neq \ell} \text{Beta}(s_{\ell,\ell'}; \alpha_{x_\ell, x_{\ell'}}) \right] \left[\prod_d^D \prod_{c_{d,n}=1} \text{Multinomial}(z_{d,n}; \eta_d) \right]
\end{aligned}
$$

Figure 3-3: The resampling equation for the keyphrase cluster assignments.

is drawn from the multinomial $\theta_{z_{d,n}}$, where $z_{d,n}$ indexes a topic-specific language model. Each of the K language models θ_k is drawn from a symmetric Dirichlet prior θ_0.

Parameter Estimation

Ultimately, we need to compute the model's posterior distribution given the training data. Doing so analytically is intractable due to the complexity of the model. In these cases, standard sampling techniques can be used to estimate the posterior. Our model lends itself to estimation via a straightforward Gibbs sampler, one of the more commonly used and simpler approaches to sampling.

By computing conditional distributions for each hidden variable given the other variables, and repeatedly sampling each of these distribution in turn, we can build a Markov chain whose stationary distribution is the posterior of the model parameters [15]. Other work in natural language processing that employs sampling techniques includes [14, 16]. We now present sampling equations for each of the hidden variables in Figure 3-2.

The prior over keyphrase clusters ψ is sampled based on hyperprior ψ_0 and keyphrase cluster assignments $\mathbf{x}$. We write $p(\psi \mid \ldots)$ to mean the probability conditioned on all the other variables.

$$\begin{aligned} p(\psi \mid \ldots) &\propto p(\psi \mid \psi_0) p(\mathbf{x} \mid \psi), \\ &= p(\psi \mid \psi_0) \prod_{\ell} p(x_\ell \mid \psi) \\ &= \text{Dirichlet}(\psi; \psi_0) \prod_{\ell} \text{Multinomial}(x_\ell; \psi) \\ &= \text{Dirichlet}(\psi; \psi'), \end{aligned}$$

where ψ'_i is $\psi_0 + \text{count}(x_\ell = i)$. This update rule is due to the conjugacy of the multinomial to the Dirichlet distribution. The first line follows from Bayes' rule, and the second line from the conditional independence of similarity scores $\mathbf{s}$ given $\mathbf{x}$ and α, and of word topic assignments $\mathbf{z}$ given η, ψ, and $\mathbf{c}$.

Resampling equations for ϕ_d and θ_k can be derived in a similar manner:

$$p(\phi_d \mid \ldots) \propto \text{Dirichlet}(\phi_d; \phi'),$$
$$p(\theta_k \mid \ldots) \propto \text{Dirichlet}(\theta_k; \theta_{k'}),$$

where $\phi'_i = \phi_0 + \text{count}(z_{n,d} = i \wedge c_{n,d} = 0)$ and $\theta'_{k,i} = \theta_0 + \sum_d \text{count}(w_{n,d} = i \wedge z_{n,d} = k)$. In building the counts for ϕ'_i, we consider only cases in which $c_{d,n} = 0$, indicating that the topic $z_{n,d}$ is indeed drawn from the document topic model ϕ. Similarly, when building the counts for θ'_k, we consider only cases in which the word $w_{d,n}$ is drawn from topic k.

To resample λ, we employ the conjugacy of the Beta prior to the Bernoulli observation likelihoods, adding counts of c to the prior λ_0.

$$p(\lambda \mid \ldots) \propto \text{Beta}(\lambda; \lambda'),$$

where $\lambda' = \lambda_0 + \begin{bmatrix} \text{count}(c_{d,n} = 1) \\ \text{count}(c_{d,n} = 0) \end{bmatrix}$.

The keyphrase cluster assignments are represented by $\mathbf{x}$, whose sampling distribution depends on ψ, $\mathbf{s}$, and $\mathbf{z}$, via η. The equation is shown in Figure 3-3. The first term is the prior on x_ℓ. The second term encodes the dependence of the similarity matrix $\mathbf{s}$ on the cluster assignments; with slight abuse of notation, we write $\alpha_{x_\ell, x_{\ell'}}$ to denote $\alpha_=$ if $x_\ell = x_{\ell'}$, and $\alpha_{\neq}$ otherwise. The third term is the dependence of the word topics $z_{d,n}$ on the topic distribution η_d. We compute the final result of Figure 3-3 for each possible setting of x_ℓ, and then sample from the normalized multinomial.

The word topics $\mathbf{z}$ are sampled according to the keyphrase topic distribution η_d, the document topic distribution ϕ_d, the observed words $\mathbf{w}$, and the auxiliary variable $\mathbf{c}$:

$$\begin{aligned}
&p(z_{d,n} \mid \ldots) \\
&\quad \propto \; p(z_{d,n} \mid \phi_d, \eta_d, c_{d,n}) p(w_{d,n} \mid z_{d,n}, \theta) \\
&\quad = \; \begin{cases} \text{Multinomial}(z_{d,n}; \eta_d)\text{Multinomial}(w_{d,n}; \theta_{z_{d,n}}) & \text{if } c_{d,n} = 1 \\ \text{Multinomial}(z_{d,n}; \phi_d)\text{Multinomial}(w_{d,n}; \theta_{z_{d,n}}) & \text{otherwise.} \end{cases}
\end{aligned}$$

As with x, each $z_{d,n}$ is sampled by computing the conditional likelihood of each possible setting within a constant of proportionality, and then sampling from the normalized multinomial.

Finally, we sample the auxiliary variables $c_{d,n}$, which indicates whether the hidden topic $z_{d,n}$ is drawn from η_d or ϕ_d. $\mathbf{c}$ depends on its prior λ and the hidden topic assignments $\mathbf{z}$:

$$\begin{aligned}
&p(c_{d,n} \mid \ldots) \\
&\quad \propto \; p(c_{d,n} \mid \lambda) p(z_{d,n} \mid \eta_d, \phi_d, c_{d,n}) \\
&\quad = \; \begin{cases} \text{Bernoulli}(c_{d,n}; \lambda)\text{Multinomial}(z_{d,n}; \eta_d) & \text{if } c_{d,n} = 1 \\ \text{Bernoulli}(c_{d,n}; \lambda)\text{Multinomial}(z_{d,n}; \phi_d) & \text{otherwise.} \end{cases}
\end{aligned}$$

Again, we compute the likelihood of $c_{d,n} = 0$ and $c_{d,n} = 1$ within a constant of proportionality, and then sample from the normalized Bernoulli distribution.

3.2.3 Modelling Using a PCA Transform of the Similarity Matrix

One of the assumptions made in the *Document Property Model* is that the values in the similarity matrix are conditionally independent given the clustering of the keyphrases. We can remove this assumption by performing transformations such as principal component analysis (PCA) on the similarity matrix. PCA is useful in this case since it reduces the dimensionality of the data, and produces a matrix of independent, normally distributed values. We provide the first A principal components of the result as input to the model.

Each row of the resulting matrix corresponds to a keyphrase we wish to cluster. We assume that each of the A scores in this row is generated by separate independent normal distributions indexed by the cluster of the keyphrase. Each cluster therefore has A normal distributions associated with it. Figure 3-5 shows the plate diagram of the model modified to work off the PCA transform of the similarity matrix.

Generative Process

The generative process for all parameters in the model except for s, μ and σ are identical to that of the Property Prediction model. We draw the mean μ of each of the A distribution over s from a univariate Normal with prior mean μ_0 and prior variance κ_0. The variance σ of the distributions are drawn from a scaled inverse Gamma distribution with prior shape parameter ν_0 and prior scale σ_0.

Next, the PCA transformed similarity matrix s $\in [0, 1]^{L \times A}$ is generated. Each row of s, s_ℓ corresponds to keyphrase ℓ, and $s_{\ell,i}$ is drawn from a normal distribution with mean $\mu_{x_\ell,i}$ and variance $\sigma_{x_\ell,i}$.

Parameter Estimation

The change to the model impacts the resampling equations for μ and σ and the equations for sampling keyphrase cluster assignments.

The conjugacy of the Normal distribution to the Normal prior on its mean and the scaled inverse Gamma prior on its variance leads to the following resampling equations for μ and

σ:

$$\begin{aligned}\mu &\sim \text{Normal}(\mu_n, \kappa_n)\\ \sigma &\sim \text{Scaled-Inv-Gamma}(\nu_n, \sigma_n)\end{aligned}$$

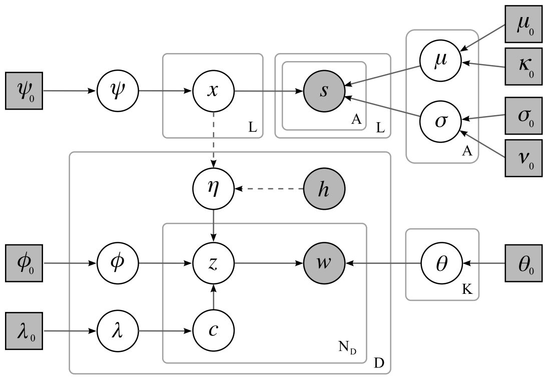
where

$$\begin{aligned}\mu_n &= \frac{\kappa_0\mu_0 + n\overline{s}}{\kappa_0 + n}\\ \kappa_n &= \kappa_0 + n\\ \nu_n &= \nu_0 + n\end{aligned}$$

$$\begin{aligned}p(x_\ell \mid \ldots) &\propto p(x_\ell \mid \psi)p(\mathbf{s} \mid x_\ell, \mu, \sigma)p(\mathbf{z} \mid \eta, \psi, \mathbf{c})\\ &\propto p(x_\ell \mid \psi)\left[\prod_{i=1}^{A} p(s_{\ell,i} \mid \mu_{x_\ell,i}, \sigma_{x_\ell,i})\right]\left[\prod_{d}^{D}\prod_{c_{d,n}=1} p(z_{d,n} \mid \eta_d)\right]\\ &= \text{Mult}(x_\ell; \psi)\left[\prod_{i=1}^{A}\text{Normal}(s_{\ell,i} \mid \mu_{x_\ell,i}, \sigma_{x_\ell,i})\right]\left[\prod_{d}^{D}\prod_{c_{d,n}=1}\text{Mult}(z_{d,n}; \eta_d)\right]\end{aligned}$$

Figure 3-4: The resampling equation for the keyphrase cluster assignments.

The sampling distribution of the keyphrase cluster assignments $\mathbf{x}$, depends on ψ, s, and $\mathbf{z}$, via η. The equation is shown in Figure 3-4. The first term is the prior on x_ℓ. The second term encodes the dependence of the similarity matrix $\mathbf{s}$ on the cluster assignments. The third term is the dependence of the word topics $z_{d,n}$ on the topic distribution η_d. We compute the final result of Figure 3-4 for each possible setting of x_ℓ, and then sample from the normalized multinomial.



ψ – keyphrase cluster model
x – keyphrase cluster assignment
s – keyphrase similarity values
μ – mean of normal distribution on s
σ – variance of normal distribution on s
h – document keyphrases
η – document keyphrase topics
λ – probability of selecting η instead of ϕ
c – selects between η and ϕ for word topics
ϕ – document topic model
z – word topic assignment
θ – language models of each topic
w – document words

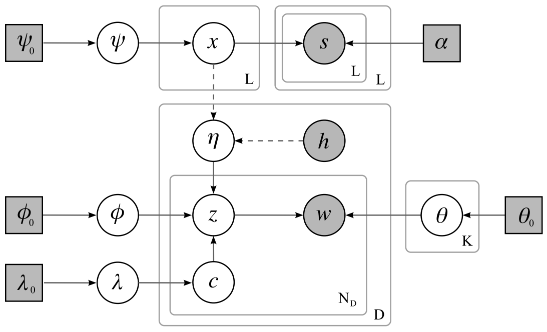
$$\psi \sim \text{Dirichlet}(\psi_0)$$
$$x_\ell \sim \text{Multinomial}(\psi)$$
$$\mu \sim \text{Normal}(\mu_0, \kappa_0)$$
$$\sigma \sim \text{Scaled-Inv-Gamma}(\nu_0, \sigma_0)$$
$$s_{\ell,i} \sim \text{Normal}(\mu_{x_\ell,i}, \sigma^2_{x_\ell,i})$$
$$\eta_d = [\eta_{d,1} \ldots \eta_{d,K}]^{\mathrm{T}}$$
where
$$\eta_{d,k} \propto \begin{cases} 1 & \text{if } x_\ell = k \text{ for any } l \in h_d \\ 0 & \text{otherwise} \end{cases}$$
$$\lambda \sim \text{Beta}(\lambda_0)$$
$$c_{d,n} \sim \text{Bernoulli}(\lambda)$$
$$\phi_d \sim \text{Dirichlet}(\phi_0)$$
$$z_{d,n} \sim \begin{cases} \text{Multinomial}(\eta_d) & \text{if } c_{d,n} = 1 \\ \text{Multinomial}(\phi_d) & \text{otherwise} \end{cases}$$
$$\theta_k \sim \text{Dirichlet}(\theta_0)$$
$$w_{d,n} \sim \text{Multinomial}(\theta_{z_{d,n}})$$

Figure 3-5: The plate diagram for the model using PCA. Shaded circles denote observed variables, and squares denote hyper parameters. The dotted arrows indicate that η is constructed deterministically from $\mathbf{x}$ and $\mathbf{h}$.

3.2.4 Global Background Property Model

One potential variation on the *Document Property Model* is to replace the document specific topic distributions ϕ_d with a single global background topic distribution ϕ that is common across all the documents. While this does not simplify training or inference, it does explore the effects of learning the overall probability of document topics. It is important to note that since the topic model is global to all documents, it encourages the topic distributions of the words of a particular document to follow the overall norm of all documents. Therefore, this model is potentially less flexible than the document topic model.

Figure 3-6 shows the plate diagram of this model.



ψ	–	keyphrase cluster model
x	–	keyphrase cluster assignment
s	–	keyphrase similarity values
h	–	document keyphrases
η	–	document keyphrase topics
λ	–	probability of selecting η instead of ϕ
c	–	selects between η and ϕ for word topics
ϕ	–	background word topic model
z	–	word topic assignment
θ	–	language models of each topic
w	–	document words

$$\begin{aligned}
\psi &\sim \text{Dirichlet}(\psi_0) \\
x_\ell &\sim \text{Multinomial}(\psi) \\
s_{\ell,\ell'} &\sim \begin{cases} \text{Beta}(\alpha_=) & \text{if } x_\ell = x_{\ell'} \\ \text{Beta}(\alpha_{\neq}) & \text{otherwise} \end{cases} \\
\eta_d &= [\eta_{d,1} \dots \eta_{d,K}]^{\text{T}} \\
&\quad \text{where} \\
&\quad \eta_{d,k} \propto \begin{cases} 1 & \text{if } x_\ell = k \text{ for any } l \in h_d \\ 0 & \text{otherwise} \end{cases} \\
\lambda &\sim \text{Beta}(\lambda_0) \\
c_{d,n} &\sim \text{Bernoulli}(\lambda) \\
\phi &\sim \text{Dirichlet}(\phi_0) \\
z_{d,n} &\sim \begin{cases} \text{Multinomial}(\eta_d) & \text{if } c_{d,n} = 1 \\ \text{Multinomial}(\phi) & \text{otherwise} \end{cases} \\
\theta_k &\sim \text{Dirichlet}(\theta_0) \\
w_{d,n} &\sim \text{Multinomial}(\theta_{z_{d,n}})
\end{aligned}$$

Figure 3-6: The plate diagram for the global background property model. Shaded circles denote observed variables, and squares denote hyper parameters. The dotted arrows indicate that η is constructed deterministically from $\mathbf{x}$ and $\mathbf{h}$.

Generative Process

The changes to the generative process from this variation are minor. Specifically, the topic z of a word w is now drawn either from the document-specific keyphrase topic model η or the global background topic distribution ϕ depending on the value of the binary auxiliary variable $c_{d.n}$. The background topic distribution ϕ is drawn from a symmetric Dirichlet prior ϕ_0.

Parameter Estimation

The derivations of the resampling equations are similar to that of the Property Prediction model, the only changes being to the equations of ϕ. In the previous model, ϕ_d was document specific, and thus it's resampling depended on the topics of the words in the relevant document. Now since ϕ is global to all documents, it is resampled based on the topics of the words of all documents:

$$p(\phi \mid ...) \propto \text{Dir}(\phi; \phi')$$

where

$$\phi' = \phi_0 + \sum_d \text{count}(z_{n,d} = i \wedge c_{n,d} = 0)$$

As before, Gibbs sampling is used to estimate the parameters of the model.

3.3 Experiments

3.3.1 Experimental Setup

Corpora Details

We evaluate our system on reviews from two categories, restaurants and cell phones. These reviews were downloaded from the popular Epinions[1] website. Users of this website evaluate products by providing both a textual description of their opinion, as well as concise lists of keyphrases (pros and cons) summarizing the review. The statistics of this dataset are provided in Table 3.1. For each of the categories, we randomly selected 50%, 15%, and 35% of the documents as training, development, and test sets, respectively.

Manual analysis of this data reveals that authors often omit properties from the list of keyphrases that are mentioned in the text. To obtain a complete gold standard, we annotated a subset of the reviews from the restaurant category manually. The annotation effort focused on eight properties that were commonly mentioned by the authors. These included properties underlying keyphrases such as "pleasant atmosphere" and "attentive staff." Two annotators performed this task, annotating collectively 160 reviews. 30 reviews were annotated by both. The Cohen's kappa, a measure of interannotator agreement that ranges from zero to one, is 0.78 on this joint set, indicating high agreement [7]. Each review was annotated with 2.56 properties on average.

	Restaurants	Cell Phones
Number of reviews	3883	1112
average review length	916.9	1056.9
average keyphrases / review	3.42	4.91

Table 3.1: Statistics of the reviews dataset by category.

Training Details

Our model needs to be provided with the number of clusters K. We set K large enough for the model to learn effectively on the development set. For example, in the restaurant

[1]http://www.epinions.com/

category, where the gold standard has eight clusters, we set K to 20. In the cell phone category, it was set to 30.

As mentioned before, we use Gibbs sampling to estimate the parameters of our model. To improve the model's convergence rate, we perform two initialization steps. In the first step, Gibbs sampling is done only on the keyphrase clustering component of the model, ignoring document text. The second step fixes this keyphrase clustering and samples the rest of the parameters in the model. These initialization steps are run for 5,000 iterations each. The full joint model is then sampled for 100,000 iterations. Inspection of the parameter estimates confirms model convergence. On a 2GHz dual-core desktop machine, model training as implemented in C++ with multi-threading takes about two hours.

The final point estimate used for testing is an average (for continuous variables) or a mode (for discrete variables) over the last 1,000 Gibbs sampling iterations. Averaging is a heuristic that is applicable in our case because our sample histograms are unimodal and exhibit low skew. The model usually works equally well using one-sample estimates, but is more prone to estimation noise.

As previously mentioned, we convert word topic assignments to document properties by examining the proportion of words supporting each property. A proportion threshold is set for each property via the development set.

Baselines

To the best of our knowledge the task of simultaneously identifying and predicting multiple properties has not been addressed in the literature. We therefore consider five baselines that allow us to explore the properties of this task and our model.

Random: Each keyphrase is supported by a document with probability of one half. This baseline's results are computed (in expectation) rather than actually run. This method is expected to have a recall of 0.5, because in expectation it will select half of the correct keyphrases. Its precision is the proportion of supported keyphrases in the test set.

Phrase in text: A keyphrase is supported by a document if it appears verbatim in the text. Precision should be high whereas recall will be low, because of the strict requirements for a keyphrase to be supported.

Cluster in text: A keyphrase is supported by a document if it or any of its paraphrases appears in the text. Paraphrasing is based on our model's clustering of the keyphrases. The use of paraphrasing information enhances recall at the potential cost of precision, depending on the quality of the clustering.

Phrase classifier: A separate discriminative classifier is trained for each keyphrase. Positive examples are documents that are labelled by the author with the keyphrase; all other documents are negative examples. A keyphrase is supported by a document if that keyphrase's classifier returns positive.

Cluster classifier: A separate discriminative classifier is trained for each cluster of keyphrases. Positive examples are documents that are labelled by the author with any keyphrase from the cluster; all other documents are negative examples. All keyphrases of a cluster are supported by a document if that cluster's classifier returns positive. Keyphrase clustering is based on our model.

Phrase classifier and *cluster classifier* employ maximum entropy classifiers, trained on the same features as our model, *i.e.*, word counts. As with the last two baselines, the former is high-precision/low-recall, because for any particular keyphrase, its synonymous keyphrases would be considered negative examples. The latter broadens the positive examples, improving recall while likely hurting precision. We used Zhang Le's Maxent toolkit[2] to build these classifiers.

Evaluation Methodology

Our first evaluation examines the accuracy of our models and the baselines by comparing their output against the keyphrases provided by the review authors. More specifically, we test whether the model supports each of the author's actual keyphrases, given the review.

As mentioned before, the author's keyphrases are incomplete. Therefore to perform a noise-free comparison, we based our second evaluation on the manually constructed gold standard for the restaurant category. We took the most commonly observed keyphrase from each of the eight annotated properties, and tested whether the model supports them.

[2]http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

In both types of evaluation, we measure the model's performance using precision, recall, and F-score. These are computed in the standard manner, based on the model's keyphrase predictions compared against the corresponding references. The sign test was used for statistical significance testing.

Our models do not attempt to predict whether a single keyphrase is supported by a given document. On the contrary, it predicts whether the underlying semantic property – represented by a clustering of the keyphrases – is supported by the document. In the evaluations above, a keyphrase is taken to be predicted if it occurs in any of the clusters predicted by the model for the document. Since a single keyphrase is used as a representative of a complete cluster, these evaluations are highly sensitive to the keyphrase clustering produced by the model. Therefore we perform two additional evaluations.

In the first of these, we compare a model output against each gold standard paraphrasing of the keyphrases provided by the original author of the review. Specifically, given one of the author's actual keyphrases, we identify all of its paraphrases according to the gold standard clustering, and evaluate the model's output on each paraphrase. This evaluation is more robust since the model neither benefits from a single correctly clustered keyphrase, nor is it heavily penalized for a single incorrect clustering.

In the second additional evaluation, we performed this paraphrasing based comparison of the model against the manually constructed gold standard for the restaurant reviews. Both of these additional evaluations were also performed against the *Cluster in Text* and *Cluster Classifier* baselines.

3.3.2 Results and Analysis

Tables 3.2 and 3.3 present the results of the evaluation scenarios described above. Our models outperforms every baseline by a wide margin in all evaluations. In particular, the *Document Topic Model* performs best is all cases except one - where the *Global Topic Model* does better. The results are consistent across the two types of evaluations - those using the most common keyphrase to represent a semantic property, and those using all paraphrasings.

The absolute performance of the automatic methods indicates the difficulty of our task. For instance, evaluation against gold annotations (see Table 3.2) shows that the random baseline outperforms all of the other baselines. We observe similarly disappointing results for the baselines on the restaurant category against the free-text annotations. The precision and recall characteristics of the baselines match our previously described expectations.

The poor performance of the discriminative models seems surprising at first. However, these results can be explained by the degree of noise in the training data, specifically, the aforementioned sparsity of free-text annotations. As previously described, our technique allows document text topics to stochastically derive from either the keyphrases or from a topic distribution [3] – this allows our models to learn effectively from incomplete annotations. In fact, when we force all text topics to derive from keyphrase clusters in our document topic model, its performance degrades to the level of the classifiers or below, with an F-score of 0.390 in the restaurant category and 0.171 in the cell phone category (compare to free-text results in Table 3.2).

As expected, paraphrasing information contributes significantly to baseline performance, generally improving recall with low impact on precision. In fact, in some instances adding paraphrasing information to the *phrase in text* baseline raises its performance to a level close to that of our models. As previously observed in entailment research [10], paraphrasing information contributes greatly to improved performance in inference tasks.

Clustering Performance

In light of this observation, it is important to quantify the quality of automatically computed paraphrases. One way to assess clustering quality is to compare it against a "gold standard" clustering, as constructed by humans. For this purpose, we use the *Rand Index* [27], a measure of cluster similarity. This measure varies from zero to one; higher scores are better. In the restaurant category, the Rand Index of our model's clusters is 0.9660; for cell phones, it is 0.8760.

Another way of assessing cluster quality is to consider the impact of using the gold

[3]Depending on the model, this is either the document topic distribution, or the global background topic distribution

clustering instead of our model's clustering in our model and the *cluster in text* and *cluster classifier* baselines. As Table 3.4 shows, using the model clustering yields results comparable to using the gold clustering. This indicates that for the purposes of our task, the model clustering is of sufficient quality.

Comparison of Models Variants

It is interesting to note the performance differences among the three variants of our model. As mentioned before, the *document topic model* is potentially more expressive than the *global topic model* and this is borne out by the results – the *document topic model* performs better in most cases, sometimes by a large margin.

The PCA variant of our model addresses one of the theoretical weaknesses of the *document topic model* by removing the strong independence assumptions made in modelling keyphrase similarity. As such, the relatively poor performance of the *PCA model* compared to the *document topic model* may be surprising. However, the PCA modification also introduces additional complexity to the model – increasing the number of model parameters by $O(\text{clusters} \times \text{phrases})$. With the datasets used in our experiments, the number of parameters that need to be estimated during training increases by approximately 1000. This additional complexity can potentially offset the benefits of better modelling, and explain the drop in performance.

	Restaurants gold annotation			Restaurants free-text annotation			Cell phones free-text annotation		
	Recall	Prec.	Fscore	Recall	Prec.	Fscore	Recall	Prec.	Fscore
Random	0.500	0.300	$*$ 0.375	0.500	0.500	$*$ 0.500	0.500	0.489	$*$ 0.494
Phrase in text	0.048	0.500	$*$ 0.087	0.078	0.909	$*$ 0.144	0.171	0.529	$*$ 0.259
Cluster in text	0.223	0.534	0.314	0.517	0.640	$*$ 0.572	0.829	0.547	0.659
Phrase clas.	0.028	0.636	$*$ 0.053	0.068	0.963	$*$ 0.126	0.029	0.600	$*$ 0.055
Cluster clas.	0.113	0.622	$\diamond$ 0.192	0.255	0.907	$*$ 0.398	0.210	0.759	0.328
DTM	0.625	0.416	**0.500**	0.901	0.652	0.757	0.886	0.585	**0.705**
PCA	0.602	0.374	0.461	0.766	0.589	0.666	0.876	0.558	0.681
GTM	0.741	0.368	0.491	0.883	0.668	**0.761**	0.867	0.520	0.650

Table 3.2: Comparison – using the most common phrase – of the property predictions made by our model and the baselines in the two categories as evaluated against the gold and free-text annotations. The methods against which our model has significantly better results on the sign test are indicated with a $*$ for $p \leq 0.05$, and $\diamond$ for $p \leq 0.1$

	Restaurants gold annotation			Restaurants free-text annotation			Cell phones free-text annotation		
	Recall	Prec.	Fscore	Recall	Prec.	Fscore	Recall	Prec.	Fscore
Cluster in text	0.260	0.421	$*$ 0.322	0.487	0.648	$*$ 0.556	0.744	0.479	$*$ 0.583
Cluster clas.	0.138	0.733	0.232	0.238	0.914	$*$ 0.378	0.198	0.754	0.314
DTM	0.617	0.549	**0.581**	0.925	0.682	**0.785**	0.853	0.571	**0.684**
PCA	0.629	0.494	0.554	0.905	0.607	0.727	0.883	0.548	0.677
GTM	0.565	0.360	0.440	0.874	0.635	0.735	0.865	0.530	0.657

Table 3.3: Comparison – using paraphrasing – of the property predictions made by our model and the baselines in the two categories as evaluated against the gold and free-text annotations. The methods against which our model has significantly better results on the sign test are indicated with a $*$ for $p \leq 0.05$.

	clustering	Restaurants			Cell phones		
		Recall	Precision	F-Score	Recall	Precision	F-Score
Cluster in text	automatic	0.517	0.640	0.572	0.829	0.547	0.659
	gold	0.542	0.608	0.573	0.914	0.497	0.644
Cluster classifier	automatic	0.255	0.907	0.398	0.210	0.759	0.328
	gold	0.221	0.895	0.354	0.162	0.739	0.266
Our model	automatic	0.901	0.652	**0.757**	0.886	0.585	**0.705**
	gold	0.795	0.627	0.701	0.886	0.520	0.655

Table 3.4: Our model and two of the baselines make use of paraphrasing information derived from our model's clustering. By providing these methods with the gold standard clustering instead, we can indirectly evaluate the quality of our model's clustering, and its impact on inference.

Chapter 4

Generating Tables of Contents

4.1 Introduction

Current research in summarization focuses on processing short articles, primarily in the news domain. While in practice the existing summarization methods are not limited to this material, they are not universal: texts in many domains and genres cannot be summarized using these techniques. A particularly significant challenge is the summarization of longer texts, such as books. The requirement for high compression rates and the increased need for the preservation of contextual dependencies between summary sentences places summarization of such texts beyond the scope of current methods.

In this chapter, we investigate the automatic generation of *tables of contents*, a type of indicative summary particularly suited for accessing information in long texts. A typical table of contents lists topics described in the source text and provides information about their location in the text. The hierarchical organization of information in the table further refines information access by specifying the relations between different topics and providing rich contextual information during browsing. Commonly found in books, tables of contents can also facilitate access to other types of texts. For instance, this type of summary could serve as an effective navigation tool for understanding a long, unstructured transcript for an academic lecture or a meeting.

Given a text, our goal is to generate a tree wherein a node represents a segment of text and a title that summarizes its content. This process involves two tasks: the hierarchical

- Scientific computing
  - Remarkable recursive algorithm for multiplying matrices
    - Divide and conquer algorithm design
    - Making a recursive algorithm
  - Solving systems of linear equations
    - Computing an LUP decomposition
    - Forward and back substitution
  - Symmetric positive definite matrices and least squares approximation

Figure 4-1: A fragment of a table of contents generated by our method.

segmentation of the text, and the generation of informative titles for each segment. The first task can be addressed by using the hierarchical structure readily available in the text (e.g., chapters, sections and subsections) or by employing existing topic segmentation algorithms [18]. In this paper, we take the former approach. As for the second task, a naive approach would be to employ existing methods of title generation to each segment, and combine the results into a tree structure.

However, the latter approach cannot guarantee that the generated table of contents forms a coherent representation of the entire text. Since titles of different segments are generated in isolation, some of the generated titles may be repetitive. Even non-repetitive titles may not provide sufficient information to discriminate between the content of one segment and another. Therefore, it is essential to generate an entire table of contents tree in a concerted fashion.

This paper presents a hierarchical discriminative approach for table of contents generation. Figure 4-1 shows a fragment of a table of contents automatically generated by this algorithm. Our method has two important points of departure from existing techniques. First, we introduce a structured discriminative model for table of contents generation that accounts for a wide range of phrase-based and collocational features. The flexibility of this model results in improved summary quality. Second, our model captures both global dependencies across different titles in the tree and local dependencies within sections. We decompose the model into local and global components that handle different classes of dependencies. We further reduce the search space through incremental construction of the model's output by considering only the promising parts of the decision space.

We apply our method to process a 1,180 page algorithms textbook. To assess the contribution of our hierarchical model, we compare our method with state of the-art methods that generate each segment title independently.[1] The results of automatic evaluation and manual assessment of title quality show that the output of our system is consistently ranked higher than that of non-hierarchical baselines.

4.2 Problem Formulation

We formalize the problem of table of contents generation as a supervised learning task where the goal is to map a tree of text segments S to a tree of titles T. A segment may correspond to a chapter, section or subsection.

Since the focus of our work is on the generation aspect of table of contents construction, we assume that the hierarchical segmentation of a text is provided in the input. This division can either be automatically computed using one of the many available text segmentation algorithms [18], or it can be based on demarcations already present in the input (e.g., paragraph markers).

During training, the algorithm is provided with a set of pairs (S^i, T^i) for $i = 1, \ldots, p$, where S^i is the i^{th} tree of text segments, and T^i is the table of contents for that tree. During testing, the algorithm generates tables of contents for unseen trees of text segments.

We also assume that during testing the desired title length is provided as a parameter to the algorithm.

4.3 Algorithm

To generate a coherent table of contents, we need to take into account multiple constraints: the titles should be grammatical, they should adequately represent the content of their segments, and the table of contents as a whole should clearly convey the relations between the segments. Taking a discriminative approach for modelling this task would allow us to

[1]The code and feature vector data for our model and the baselines are available at http://people.csail.mit.edu/branavan/code/toc.

achieve this goal: we can easily integrate a range of constraints in a flexible manner. Since the number of possible labels (i.e., tables of contents) is prohibitively large and the labels themselves exhibit a rich internal structure, we employ a structured discriminative model that can easily handle complex dependencies. Our solution relies on two orthogonal strategies to balance the tractability and the richness of the model. First, we factor the model into local and global components. Second, we incrementally construct the output of each component using a search-based discriminative algorithm. Both of these strategies have the effect of intelligently pruning the decision space.

Our model factorization is driven by the different types of dependencies which are captured by the two components. The first model is *local*: for each segment, it generates a list of candidate titles ranked by their individual likelihoods. This model focuses on grammaticality and word selection constraints, but it does not consider relations among different titles in the table of contents. These latter dependencies are captured in the *global* model that constructs a table of contents by selecting titles for each segment from the available candidates. Even after this factorization, the decision space for each model is large: for the local model, it is exponential in the length of the segment title, and for the global model it is exponential in the size of the tree.

Therefore, we construct the output for each of these models *incrementally* using beam search. The algorithm maintains the most promising partial output structures, which are extended at every iteration. The model incorporates this decoding procedure into the training process, thereby learning model parameters best suited for the specific decoding algorithm. Similar models have been successfully applied in the past to other tasks including parsing [8], chunking [11], and machine translation [9].

4.3.1 Model Structure

The model takes as input a tree of text segments S. Each segment $s \in S$ and its title z are represented as a *local feature vector* $\Phi_{\mathbf{loc}}(s, z)$. Each component of this vector stores a numerical value. This feature vector can track any feature of the segment s together with its title z. For instance, the i^{th} component of this vector may indicate whether the bigram

$(z[j]z[j+1])$ occurs in s, where $z[j]$ is the j^{th} word in z:

$$(\Phi_{\mathbf{loc}}(s,z))_i = \begin{cases} 1 & \text{if } (z[j]z[j+1]) \in s \\ 0 & \text{otherwise} \end{cases}$$

In addition, our model captures dependencies among *multiple titles* that appear in the same table of contents. We represent a tree of segments S paired with titles T with the *global feature vector* $\Phi_{\mathbf{glob}}(S,T)$. The components here are also numerical features. For example, the i^{th} component of the vector may indicate whether a title is repeated in the table of contents T:

$$(\Phi_{\mathbf{glob}}(S,T))_i = \begin{cases} 1 & \text{repeated title} \\ 0 & \text{otherwise} \end{cases}$$

Our model constructs a table of contents in two basic steps:

Step One The goal of this step is to generate a list of k candidate titles for each segment $s \in S$. To do so, for each possible title z, the model maps the feature vector $\Phi_{\mathbf{loc}}(s,z)$ to a real number. This mapping can take the form of a linear model,

$$\Phi_{\mathbf{loc}}(s,z) \cdot \alpha_{\text{loc}}$$

where α_{loc} is the local parameter vector.

Since the number of possible titles is exponential, we cannot consider all of them. Instead, we prune the decision space by incrementally constructing promising titles. At each iteration j, the algorithm maintains a beam Q of the top k partially generated titles of length j. During iteration $j+1$, a new set of candidates is grown by appending a word from s to the right of each member of the beam Q. We then sort the entries in Q: $z_1, z_2, \ldots$ such that $\Phi_{\mathbf{loc}}(s,z_i) \cdot \alpha_{\text{loc}} \geq \Phi_{\mathbf{loc}}(s,z_{i+1}) \cdot \alpha_{\text{loc}}, \forall i$. Only the top k candidates are retained, forming the beam for the next iteration. This process continues until a title of the desired length is generated. Finally, the list of k candidates is returned.

Step Two Given a set of candidate titles $z_1, z_2, \ldots, z_k$ for each segment $s \in S$, our goal is to construct a table of contents T by selecting the most appropriate title from each segment's candidate list. To do so, our model computes a score for the pair (S, T) based on the global feature vector $\Phi_{\mathbf{glob}}(S, T)$:

$$\Phi_{\mathbf{glob}}(S, T) \cdot \alpha_{\text{glob}}$$

where α_{glob} is the global parameter vector.

As with the local model (step one), the number of possible tables of contents is too large to be considered exhaustively. Therefore, we incrementally construct a table of contents by traversing the tree of segments in a pre-order walk (i.e., the order in which segments appear in the text). In this case, the beam contains partially generated tables of contents, which are expanded by one segment title at a time. To further reduce the search space, during decoding only the top five candidate titles for a segment are given to the global model.

4.3.2 Training the Model

Training for Step One We now describe how the local parameter vector α_{loc} is estimated from training data. We are given a set of training examples (s^i, y^i) for $i = 1, \ldots, l$, where s^i is the i^{th} text segment, and y^i is the title of this segment.

This linear model is learned using a variant of the incremental perceptron algorithm [8, 11]. This on-line algorithm traverses the training set multiple times, updating the parameter vector α_{loc} after each training example in case of mis-predictions. The algorithm encourages a setting of the parameter vector α_{loc} that assigns the highest score to the feature vector associated with the correct title.

The pseudo-code of the algorithm is shown in Figure 4-2. Given a text segment s and the corresponding title y, the training algorithm maintains a beam Q containing the top k partial titles of length j. The beam is updated on each iteration using the functions GROW and PRUNE. For every word in segment s and for every partial title in Q, GROW creates a new title by appending this word to the title. PRUNE retains only the top ranked candidates based on the scoring function $\Phi_{\mathbf{loc}}(s, z) \cdot \alpha_{\text{loc}}$. If $y[1 \ldots j]$ (i.e., the prefix of y of length j) is not in the modified beam Q, then α_{loc} is updated[2] as shown in line 4 of the pseudo-code

[2]If the word in the j^{th} position of y does not occur in s, then the parameter update is not performed.

in Figure 4-2. In addition, Q is replaced with a beam containing only $y[1 \dots j]$ (line 5). This process is performed $|y|$ times. We repeat this process for all training examples over 50 training iterations. [3]

s	– segment text.
y	– segment title.
$y[1 \dots j]$	– prefix of y of length j.
Q	– beam containing partial titles.

```
1  for $j = 1 \dots |y|$
2      $Q = \text{PRUNE}\ (\text{GROW}\ (s, Q))$
3      if $y[1 \dots j] \notin Q$
4          $\alpha_{\text{loc}} = \alpha_{\text{loc}} + \Phi_{\mathbf{loc}}(s, y[1 \dots j]) - \sum_{z \in Q} \frac{\Phi_{\mathbf{loc}}(s, z)}{|Q|}$
5          $Q = \{y[1 \dots j]\}$
```

Figure 4-2: The training algorithm for the local model.

Training for Step Two To train the global parameter vector α_{glob}, we are given training examples (S^i, T^i) for $i = 1, \dots, p$, where S^i is the i^{th} tree of text segments, and T^i is the table of contents for that tree. However, we cannot directly use these tables of contents for training our global model: since this model selects one of the candidate titles $z_1^i, \dots, z_k^i$ returned by the local model, the true title of the segment may not be among these candidates. Therefore, to determine a new target title for the segment, we need to identify the title in the set of candidates that is closest to the true title.

We employ the L_1 distance measure to compare the content word overlap between two titles.[4] For each input (S, T), and each segment $s \in S$, we identify the segment title closest

[3] For decoding, α_{loc} is averaged over the training iterations as in Collins and Roark [8].
[4] This measure is close to ROUGE-1 which in addition considers the overlap in auxiliary words.

in the L_1 measure to the true title y[5]:

$$z^* = \arg\min_i L_1(z_i, y)$$

Once all the training targets in the corpus have been identified through this procedure, the global linear model $\Phi_{\mathbf{glob}}(S, T) \cdot \alpha_{\text{glob}}$ is learned using the same perceptron algorithm as in step one. Rather than maintaining the beam of partially generated titles, the beam Q holds partially generated tables of contents. Also, the loop in line 1 of Figure 4-2 iterates over segment titles rather than words. The global model is trained over 200 iterations.

4.4 Features

Local Features Our local model aims to generate titles which adequately represent the meaning of the segment and are grammatical. Selection and contextual preferences are encoded in the local features. The features that capture selection constraints are specified at the word level, and contextual features are expressed at the word sequence level.

The selection features capture the position of the word, its TF*IDF, and part of speech information. In addition, they also record whether the word occurs in the body of neighbouring segments. We also generate conjunctive features by combining features of different types.

The contextual features record the bigram and trigram language model scores, both for words and for part of speech tags. The trigram scores are averaged over the title. The language models are trained using the SRILM toolkit. Another type of contextual feature models the collocational properties of noun phrases in the title. This feature aims to eliminate generic phrases, such as *"the following section"* from the generated titles.[6] To achieve this effect, for each noun phrase in the title, we measure the ratio of their frequency in the segment to their frequency in the corpus.

[5]In the case of ties, one of the titles is picked arbitrarily.

[6]Unfortunately, we could not use more sophisticated syntactic features due to the low accuracy of statistical parsers on our corpus.

Segment has the same title as its sibling
Segment has the same title as its parent
Two adjacent sibling titles have the same head
Two adjacent sibling titles start with the same word
Rank given to the title by the local model

Table 4.1: Examples of global features.

Number of Titles	540
Number of Trees	39
Tree Depth	4
Number of Words	269,650
Avg. Title Length	3.64
Avg. Branching	3.29
Avg. Title Duplicates	21

Table 4.2: Statistics on the corpus used in the experiments.

Global Features Our global model describes the interaction between different titles in the tree (See Table 4.1). These interactions are encoded in three types of global features. The first type of global feature indicates whether titles in the tree are redundant at various levels of the tree structure. The second type of feature encourages parallel constructions within the same tree. For instance, titles of adjoining segments may be verbalized as noun phrases with the same head (e.g., *"Bubble sort algorithm"*, *"Merge sort algorithm"*). We capture this property by comparing words that appear in certain positions in adjacent sibling titles. Finally, our global model also uses the rank of the title provided by the local model. This feature enables the global model to account for the preferences of the local model in the title selection process.

4.5 Evaluation Set-Up

Data We apply our method to an undergraduate algorithms textbook. For detailed statistics on the data see Table 4.2. We split its table of contents into a set of independent subtrees. Given a table of contents of depth n with a root branching factor of r, we generate r

subtrees, with a depth of at most $n-1$. We randomly select 80% of these trees for training, and the rest are used for testing. In our experiments, we use ten different randomizations to compensate for the small number of available trees.

Admittedly, this method of generating training and testing data omits some dependencies at the level of the table of contents as a whole. However, the subtrees used in our experiments still exhibit a sufficiently deep hierarchical structure, rich with contextual dependencies.

Baselines As an alternative to our hierarchical discriminative method, we consider three baselines that build a table of contents by generating a title for each segment individually, without taking into account the tree structure, and one hierarchical generative baseline. The first method generates a title for a segment by selecting the noun phrase from that segment with the highest TF*IDF. This simple method is commonly used to generate keywords for browsing applications in information retrieval, and has been shown to be effective for summarizing technical content [33].

The second baseline is based on the noisy-channel generative (flat generative, FG) model proposed by Banko et al., [2]. Similar to our local model, this method captures both selection and grammatical constraints. However, these constraints are modeled separately, and then combined in a generative framework.

We use our local model (Flat Discriminative model, FD) as the third baseline. Like the second baseline, this model omits global dependencies, and only focuses on features that capture relations within individual segments.

In the hierarchical generative (HG) baseline we run our global model on the ranked list of titles produced for each section by the noisy-channel generative model.

The last three baselines and our algorithm are provided with the title length as a parameter. In our experiments, the algorithms use the reference title length.

Experimental Design: Comparison with reference tables of contents Reference based evaluation is commonly used to assess the quality of machine-generated headlines [34]. We compare our system's output with the table of contents from the textbook using ROUGE

	Rouge-1	Rouge-L	Rouge-W	Full Match
HD	**0.256**	**0.249**	**0.216**	**13.5**
FD	0.241	0.234	0.203	13.1
HG	0.139	0.133	0.117	5.8
FG	0.094	0.090	0.079	4.1
Keyword	0.168	0.168	0.157	6.3

Table 4.3: Title quality as compared to the reference for the hierarchical discriminative (HD), flat discriminative (FD), hierarchical generative (HG), flat generative (FG) and Keyword models. The improvement given by HD over FD in all three Rouge measures is significant at $p \leq 0.03$ based on the Sign test.

metrics. We employ a publicly available software package,[7] with all the parameters set to default values.

Experimental Design: Human assessment The judges were each given 30 segments randomly selected from a set of 359 test segments. For each test segment, the judges were presented with its text, and 3 alternative titles consisting of the reference and the titles produced by the hierarchical discriminative model, and the best performing baseline. In addition, the judges had access to all of the segments in the book. A total of 498 titles for 166 unique segments were ranked. The system identities were hidden from the judges, and the titles were presented in random order. The judges ranked the titles based on how well they represent the content of the segment. Titles were ranked equal if they were judged to be equally representative of the segment.

Six people participated in this experiment. All the participants were graduate students in computer science who had taken the algorithms class in the past and were reasonably familiar with the material.

4.6 Results

Figure 4-3 shows fragments of the tables of contents generated by our method and the four baselines along with the reference counterpart. These extracts illustrate three general phenomena that we observed in the test corpus. First, the titles produced by keyword

[7]http://www.isi.edu/licensed-sw/see/rouge/

Reference:

- hash tables
  - direct address tables
  - hash tables
    - collision resolution by chaining
    - analysis of hashing with chaining
  - open addressing
    - linear probing
    - quadratic probing
    - double hashing

Keyword Extraction:

- hash table
  - dynamic set
  - hash function
    - worst case
    - expected number
  - hash table
    - hash function
    - hash table
    - double hashing

Flat Generative:

- linked list
  - worst case time
  - wasted space
    - worst case running time
    - to show that there are
  - dynamic set
    - occupied slot
    - quadratic function
    - double hashing

Flat Discriminative:

- dictionary operations
  - universe of keys
  - computer memory
    - element in the list
    - hash table with load factor
  - hash table
    - hash function
    - hash function
    - double hashing

Hierarchical Generative:

- dictionary operations
  - worst case time
  - wasted space
    - worst case running time
    - to show that there are
  - collision resolution
    - linear time
    - quadratic function
    - double hashing

Hierarchical Discriminative:

- dictionary operations
  - direct address table
  - computer memory
    - worst case running time
    - hash table with load factor
  - address table
    - hash function
    - quadratic probing
    - double hashing

Figure 4-3: Fragments of tables of contents generated by our method and the four baselines along with the corresponding reference.

	better	worse	equal
HD vs. FD	68	32	49
Reference vs. HD	115	13	22
Reference vs. FD	123	7	20

Table 4.4: Overall pairwise comparisons of the rankings given by the judges. The improvement in title quality given by HD over FD is significant at $p \leq 0.0002$ based on the Sign test.

extraction exhibit a high degree of redundancy. In fact, 40% of the titles produced by this method are repeated more than once in the table of contents. In contrast, our method yields 5.5% of the titles as duplicates, as compared to 9% in the reference table of contents.[8]

Second, the fragments show that the two discriminative models — Flat and Hierarchical — have a number of common titles. However, adding global dependencies to rerank titles generated by the local model changes 30% of the titles in the test set.

Comparison with reference tables of contents Table 4.3 shows the average ROUGE scores over the ten randomizations for the five automatic methods. The hierarchical discriminative method consistently outperforms the four baselines according to all ROUGE metrics.

At the same time, these results also show that only a small ratio of the automatically generated titles are identical to the reference ones. In some cases, the machine-generated titles are very close in meaning to the reference, but are verbalized differently. Examples include pairs such as *("Minimum Spanning Trees", "Spanning Tree Problem")* and *("Wallace Tree", "Multiplication Circuit")*.[9] While measures like ROUGE can capture the similarity in the first pair, they cannot identify semantic proximity between the titles in the second pair. Therefore, we supplement the results of this experiment with a manual assessment of title quality as described below.

Human assessment We analyze the human ratings by considering pairwise comparisons between the models. Given two models, A and B, three outcomes are possible: A is better than B, B is better than A, or they are of equal quality. The results of the comparison are summarized in Table 4.4. These results indicate that using hierarchical information yields statistically significant improvement (at $p \leq 0.0002$ based on the Sign test) over a flat counterpart.

[8]Titles such as *"Analysis"* and *"Chapter Outline"* are repeated multiple times in the text.
[9]A Wallace Tree is a circuit that multiplies two integers.

Chapter 5

Conclusions and Future Work

In this work, we have presented two summarization methods which are able to achieve high compression rates on par with human editors. By attaining compression rates previously beyond the reach of widely applicable automatic methods, our algorithms make possible novel means of searching, navigating and accessing large collections of textual information.

Our first algorithm is able to condense a collection of document into a summary less than 0.1% the size of the collection. This summary is composed of a list of semantic properties supported by the given documents. The method uses consistent wording in generating this list, thus enabling novel applications such as the automatic comparison of documents, and searching or browsing documents based on their semantic content. In this work, we have evaluated the algorithm on a collection of product reviews. However, the method is not dependent on any feature specific to this dataset, and is directly applicable to any collection of documents with associated free-text annotations.

The second model summarizes long documents such as books or lecture transcripts into tables of contents with a compression rate of better than 1%. This summary is a succinct representation of the document in both content and structure, and brings with it the benefits of manually created tables of contents - allowing readers to efficiently navigate and search through long documents. But in addition, it opens up the possibility of automatically adding such a summary to the vast amounts of information presently available in electronic form for which tables of contents do not exist.

The potency of these methods is due in part to the extremely high compression rates they are able to achieve while producing summaries that are representative of the original documents. This ability is because of their in-depth modelling of document content. As the empirical results show, both of the models significantly out-perform prior methods on real-world tasks and data, confirming the benefits of such refined modelling of document structure.

Many approaches in natural language processing are based on the use of professionally annotated training data. Often, the lack of such training data becomes a significant limiting factor. The method presented in this work for summarizing text collections learns successfully from documents partially annotated with free-text by lay authors. Like the corpus used in our experiments on review summarization, a variety of free-text annotated data is freely available on the internet. In addition to confirming the feasibility of using such datasets, our method also suggests one potential approach to learning from these noisy annotations. By reducing the dependence on expensive professionally created corpuses, this potentially opens up new applications and avenues of research.

In the future, we would like to explore further refinements in the modelling of document structure. For example, in summarizing documents into semantic properties, we have assumed that the properties themselves are unstructured. In reality, these properties are related in interesting ways. As a trivial example, a single document is unlikely to support conflicting properties. Thus it would be desirable to model the correlations between the properties. More complex structures such as hierarchical relationships can also be considered. Finally, it would be interesting to investigate other applications to which free-text annotations can be applied.

Appendix A

Examples of Product Reviews

Given below are a few examples of product reviews and associated pros/cons phrases downloaded from the Epinions.com website.

Pros:	**Cons**:
Very convenient and tasty food	Not a substitute for a good meal - - and not healthy to have all the time.

Review Text:

If there is ever an occasional night when you’re really hungry but you don’t feel like cooking or paying a fortunate to go out, then Burger King might be an okay alternative. However in general, I would not recommend Burger King as a regular source of nutrition. Although healthier than Mc. Donalds (because Burger King broils their burgers), there is still an awful lot of grease that goes into the food. That is not to say that you should never go to Burger King. Also, Burger King does have halfway nutritious things that you can eat. I always look forward to the occasional burger from the restaurant as well, but at the same time, I don’t depend on it for one or even two meals a day. Burger King is the kind of place to go when you really don’t have anything to eat in the house and when you want something quick and satisfying. It’s not a substitute for healthy foods though. Burger King is a very convenient place to go when you want to pick something up on the run. The ”drive-thru” makes it pretty easy to order your food and then take off, even though the service at the drive thru doesn’t always live up to the ”fast food” term, with an emphasis on ”fast.” I certainly won’t lie to you. There’s nothing more enjoyable than one or two BK Broilers or Whoppers when you’re hungry. The food from Burger King is very tasty. The first bite into a Whopper is just as good as the last. It does become difficult to avoid the temptation to eat there a number of times a month or even a week. The main reason Burger King has a three star rating in my opinion is because in general, the food isn’t very healthy and it certainly isn’t a substitute for a balanced meal or even a balanced diet for that matter. Believe me. I know people that eat there every other night, and you don’t want to see what can happen to people that eat there almost all the time. Again, this is not say that Burger King is a horrible restaurant. Their food is very tasty and is okay to have a couple times a month, but it’s important not to base your diet around the place and to make more healthy meals for lunch or dinner. Try having chicken, rice, and a nice salad when you want a good meal. On the other hand, Burger King is a convenient place to pick something up when you’re on the go all the time. Even if you’re on the go every night, you can still find other places to eat a more nutritious meal though. I would recommend Burger King for families who want to have a treat once in a while. The food is tasty and enjoyable, but far from nutritious. Just be careful where you eat and don’t make a habit of going to Burger King all the time. I must say that having a Double Whopper once in a while is almost better than having a chocolate sundae, however to have that kind of food as a normal diet is far from nutritious. All of the food tastes great, but you might change your mind if you saw a nutrition fact table. Service is pretty efficient, but that’s all.

Pros:
- Excellent wait staff
- Great sweet tea

Cons:
- The food lacked spices
- The restaurant lacks distinction

Review Text:

Bob Evans is a chain of restaurants that originated in the Midwest, and now claims over 400 locations throughout the country. I have seen the restaurant many times off the interstate when I travel, but rarely had I dared to set foot in the restaurant. That is, until today . Upon entering the restaurant, the first thing that struck me was the highly unoriginal atmosphere. The "country store"-style cashier area reminded me of Cracker Barrel. The bar, complete with metal swivel stools, harkened of Steak and Shake or Waffle House. The booth seating was reminiscent of Shoney's. Almost every bit of the atmosphere seemed ripped off from other restaurant chains, but I resolved to reserve my judgement for the food. One definite plus to the Bob Evans dining experience is the helpful wait staff. Upon arrival, drink orders were taken almost immediately and menus were presented. Both my dining partner and I chose sweet tea, and I would be remiss if I didn't mention that the tea was fabulous! (It was just the right sweetness: not quite syrupy, but nowhere near watery, with the tart lemon wedge bringing the flavors together in harmony.) Food orders were taken a few minutes later (I suspect that myself and my dining partner were given extra time to digest the menu, as we were new to the restaurant). The food at Bob Evans, described often as "homestyle", struck me more as being "safe". The dinner entrees are very simple dishes: chicken (grilled, fried, or BBQ'ed), fried steak, pot roast and chicken salad sandwiches. The breakfast menu features the usual suspects: pancakes, various omelets, and many biscuit platters. There is nothing on the Bob Evans menu that can be considered exotic, sensual, or the least bit daring, so those who seek food that is at all arousing will be sorely disappointed with Bob Evans. Those who seek comfort food at its most basic will probably be pleased. I decided on the Wildfire Chicken Breast entree, which came with two sides (I chose the grilled veggies and green beans), and a biscuit or a roll (I chose the roll). My dining companion chose the Grilled Chicken Breast entree, with mashed potatoes and corn and a biscuit. I thought that we would have to wait a while for our food, but no more than 10 minutes passed before we were served. While my chicken breast was thoroughly cooked and was at the perfect medium between dry and juicy, the bliss of the perfectly cooked meat was completely thrown off by the unrelenting sweetness of the barbeque sauce. A molasses base in the wrong hands is a terrible thing, but what was also confusing was how a barbeque sauce with a name like "Wildfire" could have absolutely no spices. It was as if I had entered a parallel universe where paprika, cumin, and hot peppers no longer existed. Thus began the downward spiral. My vegetables were equally disappointing. The grilled veggies were a very soggy mish-mash of squash, zucchini, and julienned carrots. My green beans were served with a heavy dose of ham hock; while I like ham in my green beans, the flavor of the green beans was overwhelmed by the amount of ham in the dish. (Vegetarians, take note: it is not stated on the menu that the green beans contained meat, so be aware and ask questions about anything on the menu.) I began to deeply regret my dinner selection. I ended up nibbling off my dining partner's plate. His grilled chicken was well-cooked, with a seasoning both smoky and peppery at the same time. The mashed potatoes and corn were eerily plain in taste, but not awful. The biscuit was an adequately fluffy accompaniment to the meal. By way of not being topped by a sickly-sweet, syrupy disaster of a sauce, his dish was much better than mine. Bob Evans has excellent wait staff, who made sure that my glass was never empty and attended to all of my needs. The price of the food was very reasonable; two people ate dinner for less than $20. However, I simply cannot recommend Bob Evans based on the strength of its dinner menu. The dearth of spice in the food was not the worst part; it was as if the restaurant was not making nearly as much of an effort with the food as with the wonderful service. Because the reputation of a restaurant ultimately stands on its food, the restaurant experience could only be described as lackluster at best, and disastrous at worst. After reading some of the opinions of my fellow Epinionators, I realize in hindsight that I probably would have had more enjoyment of the breakfast food . it comes highly recommended and is served all day. There are many restaurants that attempt the breakfast-all-the-time/dinner-by-night concept, but few succeed (one that succeeds in a grand way is Aunt Sarah's Pancakes, a chain of restaurants based in Virginia off I-95). Perhaps in the future I will try Bob Evans' breakfast menu, but it will have to be after I shake off the memory of plain-Jane food, the absence of spices, and disgustingly sweet BBQ sauce.

Pros:
- Thick shakes
- Flavor choices

Cons:
- Service
- Waffle cones
- Not that great
- Expensive

Review Text:

Thankfully, there are plenty of choices, when it comes to finding an ice cream parlor. You can drop by Carvel, which is known for its former owner's advertisements, that extolled the values of eating Cookie Pus. There is Dairy Queen, which is known in the Northeast for its Pick Up Windows. This is a good place to "pick up" a date and a higher cholesterol reading, in addition to an ice cream cone. Baskin Robbins is a perennial favorite, with its 1 billion flavors, and Ben and Jerry's always seems to have odd named products. This would include Lime Flavored Antifreeze and I Can't Believe this is 7-11 coffee flavored ice cream. After eating at many of these establishments, and some local competitors that haven't made the big time, I am something of an ice cream expert. My belly hovers farther away from the belt after each tasting, and the weight scale cries when I step on it. In search of the perfect ice cream, I decided to try a Cold Stone Creamery. Their advertisements caught my eye, as they claim to have the freshest ice cream. They have been highly sucessful, with over 1300 stores being built in under 20 years. In Northern Virginia, they occupy every street corner, much like a "cocktail waitress with a Dolly Parton wig" as described by the country band Confederate Railroad. I decided to go for something exciting on my first visit. The chocolate waffle cone that I ordered was a bit disappointing, and at a price tag of $4.61 for a little over a single scoop, it was not favorable on my wallet. While it was not as bad as a fast food ice cream cone, it lacked the necessary flavor to be worthy of its price tag. It was fresh, which to their credit is what they advertised, but the quality was a bit below premium. It melted quickly in the reasonably cool store, and the waffle was downright disgusting. It tasted like an old Eggo that had been sitting around, as it was the very definition of stale. It broke apart easily, and led to a rather messy dessert meal. After this visit of low satisfaction, I decided to give them a try on one of their original flavors, with their Coffee Ice Cream. In addition to the coffee, it has almonds, caramel, and Heath Bars. It looks like everything was mixed in together equally, and it has a distinctive flavor. This is due to the delicious candy bars and caramel, but the coffee taste was disappointing. At $4 for a cup of this, I would expect it to taste like one of Starbucks or Caribou's exotic beverages from Africa. Instead, it tasted like the local gas station blend, that is sold on the street for "donations only." This led to my decision that is not worth checking out any other of their flavors. I did try a milkshake, so that I could at least say I gave them another chance. Their large vanilla shake cost $4, and it is of decent size. It is also thick, as in the way a milkshake should be. However, the flavor was again nothing special, and I believe it possible to get a much better product elsewhere. The cup that it was placed in did hold up, and this is a shake that you almost need a spoon for. They offer a variety of sizes on this, and even smaller ones for little ones who haven't acquired full use of their taste buds. The service has been below average. The first time I went in there, I had to practically yell to get assistance, as the employees were busy discussing "who's dating who," and "Do you think she likes me, even though she dropped an anvil on my head?" Each time has been slow, even though they have not been busy. During summer periods, they do have lines out the door for some reason, but the lines were only two or three at most. The employees didn't seem to knowledgeable about their products either, and were not able to recommend any product that I would like according to my horned rim glasses conservative taste. The comfort level was decent. They could use benches with padding, considering their contributions towards healthy dining, but they have wooden chairs that are attached to tables. These are a lot like the local prison. The chairs offer good support, although they are a bit uncomfortable if you sit there for an extended period of time. They also are not going to make taller people happy, with the arrangement of being attached together. As for cleanliness, they were above average, as the tables, service area, and bathrooms were clean. I cannot recommend Cold Stone Creamery. They are below their competitors when it comes to selections and quality of products. If you are going to spend $4 for an ice cream cone, I would suggest going to Baskin Robbins or your local Uncle Ernie's Its a shame as their advertisements are appealing, but they just cannot compete with their subpar products. I am however interested in seeing if my wife, would like to drop an anvil on me, if I eat there again.

Appendix B

Gold standard annotations

B.1 Annotation Instructions

Given below are the instructions given to annotators when creating the gold standard annotations for the product review corpus. The instructions were created as follows: first, two people were asked to independently annotate the same ten reviews. The annotations were then compared, and the annotators asked to rationalize their decisions. Any potential causes of confusion were identified through this process. The following instructions were then created to ensure that all annotation decisions were made on the same basis.

ANNOTATOR INSTRUCTIONS

You will be presented with a series of restaurant reviews, taken from a popular online review website. For each review, we ask that you make a judgement as to whether the review expresses one or more of the following opinions, from the perspective of the reviewer:

1. That the food was good (+ *food*)
2. That the atmosphere was good (+ *atm*)
3. That the service was good (+ *service*)

4. That the staff was good (+ *staff*)
5. That the pricing was good (+ *price*)
6. That the food was bad (- *food*)
7. That the service was bad (- *service*)
8. That the pricing was bad (- *price*)

Make as many annotations for each review as necessary.

Notes:

1. All judgements should be from the perspective of the reviewer. For example, if the reviewer states "someone who likes Mexican food would have liked this restaurant, but I didn't like it," this should be annotated as - *food*.
2. Only make attributions for explicit (or strongly implied) judgements. For example, if the reviewer describes the atmosphere in neutral terms, this should not be annotated.
3. The dimensions are not symmetric, because in our study of major opinion groups we did not find consistent negative reviews in the atmosphere and staff aspects. If the review criticizes the atmosphere, this does not need to be annotated.
4. The staff aspect is about direct interactions with the staff, whereas the service aspect is about everything else service-related, such as wait time, order correctness, etc. For example, + *staff* may include knowledgeable or friendly staff, whereas + *service* may include promptness of seating.
5. The restaurant being clean, by itself, does not mean + *atm* unless the reviewer states that he or she enjoyed the atmosphere as well.
6. + *price* should be made if the reviewer states the food was cheap for what he or she got (a good value), and the opposite for - *price*. Price annotations should not be made on the basis of absolute values of price (e.g., a $100 meal is not automatically - *price*), or based on the reviewer's comparison against other restaurants.

B.2 Examples of annotations from multiple judges

Table B.2 below shows the gold standard annotations produced by the human annotators for a few reviews. Also shown are the pros and cons phrases that had been written by the original author along with the review. The difference between the gold standard annotations and the pros/cons is because reviews authors often do not list all of their opinions as pros or cons.

Original Author's pros/cons	**Annotator 1**	**Annotator 2**
pros: Excellent wait staff Great sweet tea *cons:* The food lacked spices The restaurant lacks distinction	- atmosphere + staff + service + price - food	- food + service + price
pros: Thick shakes Flavor choices *cons*: Service Waffle cones Not that great Expensive	- food - service - staff - price	- food - price - service
pros: Quality ice cream and dairy products *cons*: A little expensive but worth it	+ food + price + service	+ food + service + staff + price
pros: Great pizza Great service Fresh ingredients *cons*: You do have to bake it yourself Have to go pick it up yourself	+ food + price + staff + service	+ food + service + staff + price

Appendix C

Examples of Automatically Computed Keyphrase Clusters

Figures C-1 and C-2 show examples of keyphrase clusters that were automatically computed by the *Document Property Model* described in section 3.2.2.

- size
- small size
- compact size
- great size
- good size
- tiny size
- nice size
- sleek

- battery life
- short battery life
- poor battery life
- low battery life
- bad battery life
- so so battery life
- battery life could be better
- terrible battery life

- looks
- style
- cute
- nice design
- nice looking
- appearance
- looks cool
- looks great
- looks good
- good looks
- styling
- functionality
- clear calls

Figure C-1: Examples of automatically computed keyphrase clusters for the Cellphone domain

bad customer service	fairly priced	overpriced food	high quality food
slow service	well priced	poor quality food	well prepared food
mediocre service	average prices	poor food quality	pretty good food
inconsistent service	competitive prices	awful food	good quality food
lousy service	cheap prices	nasty food	very good food
unfriendly service	affordable prices	terrible food	great tasting food
service can be slow	decent prices	poor food	food is good
rude service	moderate prices	food quality	superb food
poor service	excellent prices	average food	incredible food
horrible service	very reasonable prices	bad food	food is great
terrible service	reasonable prices	horrible food	amazing food
bad service	fair prices	quality of food	yummy food
very poor service	reasonable cost	lousy food	awesome food
customer service	great price	mediocre food	terrific food
spotty service	the price	cold food	quality food
the service		bland food	tasty food
		greasy food	delicious food
			excellent food
			food is excellent

Figure C-2: Examples of automatically computed keyphrase clusters for the Restaurant domain

Appendix D

Pseudo Code

Listed below is the pseudo code for estimating the parameters of the *Document Property Model* as described in section 3.2.2. The complete implementations of both of the summarization methods discussed in this thesis are available from *http://people.csail.mit.edu/branavan.*

```
EstimateModelParameters ()

  for $N_p$ iterations:
      SamplePhraseClustersIndependently

  for $N_l$ iterations:
      SampleDocumentParameters

  for $N_j$ iterations:
      SamplePhraseClustersJointly
      SampleDocumentParameters
```

```
SamplePhraseClustersIndependently ()

  for each phrase $\ell \in L$:
      compute $p(x_\ell \mid \ldots) = \text{Multinomial}(x_\ell; \psi) \left[ \prod_{\ell' \neq \ell} \text{Beta}\left(s_{\ell,\ell'}; \alpha_{x_\ell, x_{\ell'}}\right) \right]$

      sample a cluster assignment $x_\ell$ for phrase $\ell$ from
         the multinomial distribution defined by $p(x_\ell \mid \ldots)$

  compute $\psi'$ where $\psi'_i = \psi_0 + \text{count}(x_\ell = i)$
```

```
sample $\psi$ from $p(\psi \mid \ldots) = \mathrm{Dirichlet}(\psi; \psi')$
```

```
**SamplePhraseClustersJointly** ()

for each phrase $\ell \in L$:
    compute $p(x_\ell \mid \ldots) = \mathrm{Multinomial}(x_\ell; \psi) \left[ \prod_{\ell' \neq \ell} \mathrm{Beta}\left(s_{\ell,\ell'}; \alpha_{x_\ell, x_{\ell'}}\right) \right] \left[ \prod_{d}^{D} \prod_{c_{d,n}=1} \mathrm{Multinomial}(z_{d,n}; \eta_d) \right]$

    sample a cluster assignment $x_\ell$ for phrase $\ell$ from
        the multinomial distribution defined by $p(x_\ell \mid \ldots)$

compute $\psi'$ where $\psi'_i = \psi_0 + \mathrm{count}(x_\ell = i)$
sample $\psi$ from $p(\psi \mid \ldots) = \mathrm{Dirichlet}(\psi; \psi')$
```

```
**SampleDocumentParameters** ()

for each document:
    ▷ deterministically compute cluster annotations $\eta_d$ from
      document phrase annotations $h_d$ and phrase clusterings $x_\ell$

    $\eta_{d,i} \propto \begin{cases} 1 & \text{if } x_\ell = i \text{ for any } \ell \in h_d \\ \epsilon & \text{otherwise} \end{cases}$

    normalize $\eta_d$

    for each token $w_{d,n}$ in document:
        sample token topic $z_{d,n}$ from $p(z_{d,n} \mid \ldots)$
        $p(z_{d,n} \mid \ldots) \propto \begin{cases} \mathrm{Multinomial}(z_{d,n}; \eta_{\mathbf{d}})\mathrm{Multinomial}(w_{d,n}; \theta_{z_{d,n}}) & \text{if } c_{d,n} = 1 \\ \mathrm{Multinomial}(z_{d,n}; \phi_{\mathbf{d}})\mathrm{Multinomial}(w_{d,n}; \theta_{z_{d,n}}) & \text{otherwise.} \end{cases}$

    ▷ sample language models
    sample $\theta_k$ from $p(\theta_k \mid \ldots) \propto \mathrm{Dirichlet}(\theta_k; \theta'_k)$
    where $\theta'_{k,i} = \theta_0 + \sum_d \mathrm{count}(w_{n,d} = i \wedge z_{n,d} = k)$

    ▷ sample document topic model
    sample $\phi_d$ from $p(\phi_d \mid \ldots) \propto \mathrm{Dirichlet}(\phi_d; \phi')$
    where $\phi'_i = \phi_0 + \mathrm{count}(z_{n,d} = i \wedge c_{n,d} = 0)$

    ▷ sample word topic source
    sample word topic source $c_{d,n}$ from $p(c_{d,n} \mid \ldots)$
```

where $p(c_{d,n} \mid \ldots) \propto \begin{cases} \text{Bernoulli}(c_{d,n}; \lambda)\text{Multinomial}(z_{d,n}; \eta_d) & \text{if } c_{d,n} = 1 \\ \text{Bernoulli}(c_{d,n}; \lambda)\text{Multinomial}(z_{d,n}; \phi_d) & \text{otherwise.} \end{cases}$

$\triangleright$ *sample* λ
sample λ from $p(\lambda \mid \ldots) \propto \text{Beta}(\lambda \mid \lambda')$
where $\lambda' = \lambda_0 + \begin{bmatrix} \text{count}(c_{d,n} = 1) \\ \text{count}(c_{d,n} = 0) \end{bmatrix}$

Appendix E

Generated Tables of Contents

Given below are the actual and generated table of contents for each of eight chapters of the undergraduate textbook *Introduction to Algorithms*. The titles in the left column are from the book itself, and the ones on the right are those generated by our system.

Original Table of Contents

- Arithmetic Circuits
  - Combinational circuits
    - Combinational elements
    - Combinational circuits
    - Full adders
    - Circuit depth
    - Circuit size
  - Addition circuits
    - Computing carry statuses with a parallel prefix circuit
    - Completing the carry lookahead adder
  - Multiplication circuits
    - Analysis
    - Analysis
  - Clocked circuits
    - Analysis
    - Ripple carry addition versus bit serial addition
    - A slow linear array implementation
    - A fast linear array implementation

- Minimum Spanning Trees
  - Growing a minimum spanning tree
  - The algorithms of Kruskal and Prim
    - Kruskal 's algorithm
    - Prim 's algorithm

Automatically Generated Table of Contents

- Grade school
  - Combinational circuits
    - Combinational circuits
    - Combinational circuits
    - Truth table
    - Comparison networks
    - Combinational circuits
  - Ripple carry
    - Using an element with a parallel prefix circuit
    - Computing an input and output
  - Wallace tree
    - Lower
    - Inputs
  - Clock period
    - Periods
    - Replicated bit of a combinational circuit elements
    - Bit numbers takes constant time
    - Using a linear time algorithm

- Spanning tree problem
  - Finding a minimum spanning tree
  - Elaborations of a minimum spanning tree
    - Minimum spanning tree
    - Minimum spanning tree

Original Table of Contents

- Probabilistic Analysis and Randomized Algorithms
  - The hiring problem
    - Worst case analysis
    - Probabilistic analysis
    - Randomized algorithms
  - Indicator random variables
    - Analysis of the hiring problem using indicator random variables
  - Randomized algorithms
    - Randomly permuting arrays
    - An analysis using indicator random variables

- The Role of Algorithms in Computing
  - Algorithms
    - What kinds of problems are solved by algorithms ?
    - Data structures
    - Technique
    - Hard problems
  - Algorithms as a technology
    - Efficiency
    - Algorithms and other technologies

- All Pairs Shortest Paths
    - Chapter outline
  - Shortest paths and matrix multiplication
    - The structure of a shortest path
    - A recursive solution to the all pairs shortest paths problem
    - Computing the shortest path weights bottom up
    - Improving the running time
  - The Floyd Warshall algorithm
    - The structure of a shortest path
    - A recursive solution to the all pairs shortest paths problem
    - Computing the shortest path weights bottom up
    - Constructing a shortest path
    - Transitive closure of a directed graph
  - Johnson 's algorithm for sparse graphs
    - Preserving shortest paths by reweighting
    - Producing nonnegative weights by reweighting
    - Computing all pairs shortest paths

- Elementary Graph Algorithms
  - Representations of graphs
  - Breadth first search
    - Analysis
    - Shortest paths
    - Breadth first trees
  - Depth first search
    - Properties of depth first search
    - Classification of edges
  - Topological sort
  - Strongly connected components

Automatically Generated Table of Contents

- Probabilistic analysis and randomized algorithms
  - Counting the number
    - Situation in which
    - Running time
    - Probability distribution
  - Indicator random variables
    - Computing the value of the expected number of times
  - Probabilistic analysis
    - Permuting the input
    - Using the definition of the variables

- Study of the algorithms of chapter
  - Values
    - Solving the problem of sorting algorithms and solutions
    - Data structures
    - Problems
    - Efficient algorithms
  - Infinitely fast algorithms for
    - Differences
    - Computing an efficient algorithm

- Finding a shortest path
    - Warshall algorithm
  - Developing a shortest paths problem
    - Characterizing the shortest paths and vertices
    - Consisting of a shortest path from the minimum weight edges
    - Taking a shortest path weights and matrices
    - Computing the matrix product
  - Floyd warshall algorithm to
    - Vertices on shortest paths and algorithms
    - Formulation of a shortest path estimates and recursive and -
      - observations
    - Recurrence for the following procedure and values
    - Constructing shortest path weights
    - Transitive closure of a directed graph
  - Sparse graphs and shortest path weights
    - Showing the shortest path weights
    - Paths from a source vertex
    - Using the bellman ford algorithm

- Chapter 9999 presents
  - Adjacency list representation
  - Breadth first search
    - Easier
    - Source vertex
    - Breadth first search
  - Depth first search
    - Property of depth first search
    - Depth first search
  - Topological sort
  - Depth first search

Original Table of Contents

- Hash Tables
  - Direct address tables
  - Hash tables
    - Collision resolution by chaining
    - Analysis of hashing with chaining
  - Hash functions
    - What makes a good hash function ?
    - Interpreting keys as natural numbers
    - Designing a universal class of hash functions
  - Open addressing
    - Linear probing
    - Quadratic probing
    - Double hashing
    - Analysis of open address hashing

- NP Completeness
    - NP completeness and the classes <P> and <NP>
    - Overview of showing problems to be NP complete
    - Chapter outline
  - Polynomial time
    - Abstract problems
    - Encodings
    - A formal language framework
  - Polynomial time verification
    - Hamiltonian cycles
    - Verification algorithms
    - The complexity class NP
  - NP completeness and reducibility
    - Reducibility
    - NP completeness
    - Circuit satisfiability
  - NP completeness proofs
    - Formula satisfiability
    - 3 CNF satisfiability
  - NP complete problems

Automatically Generated Table of Contents

- Dictionary operations
  - Direct address table
  - Computer memory
    - Worst case running time
    - Searching for a given set
  - Hash functions
    - Real numbers and a hash function
    - Creating a set of keys
    - Hash functions and the number of keys
  - Hash table
    - Hash function
    - Hash function
    - Double hashing
    - Hash table with load factor

- Polynomial time
    - Classes of a directed graph of np completeness
    - Using the technique of a directed graph algorithms
    - NP completeness
  - NP completeness
    - Abstract problems
    - Programs
    - Using a decision problem
  - Algorithms for which
    - Hamiltonian cycle
    - Verification algorithm
    - Complexity class of languages
  - Why theoretical computer scientists
    - Problems
    - NP completeness
    - NP completeness
  - Formula satisfiability problems
    - Satisfiability problem
    - CNF satisfiability problem
  - NP complete problems

Bibliography

[1] R. Angheluta, R. De Busser, and M. Moens. The use of topic segmentation for automatic summarization. In *Proceedings of the ACL Post-Conference Workshop on Automatic Summarization.*, 2002.

[2] Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. Headline generation based on statistical translation. In *Proceedings of the ACL*, pages 318–325, 2000.

[3] David M. Blei and John D. Lafferty. Correlated topic models. In *NIPS*, 2005.

[4] David M. Blei and J. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems*, 2007.

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

[6] Branimir Boguraev and Mary S. Neff. Discourse segmentation in aid of document summarization. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pages 3004–3014, 2000.

[7] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[8] Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the ACL*, pages 111–118, 2004.

[9] Brooke Cowan, Ivona Kucerova, and Michael Collins. A discriminative model for tree-to-tree translation. In *Proceedings of the EMNLP*, pages 232–241, 2006.

[10] Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. *Lecture Notes in Computer Science*, 3944:177–190, 2006.

[11] Hal Daumé and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the ICML*, pages 169–176, 2005.

[12] Bonnie Dorr, David Zajic, and Richard Schwartz. Hedge trimmer: a parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop*, pages 1–8, 2003.

[13] Noemie Elhadad and Kathleen R. McKeown. Towards generating patient specific summaries of medical articles. In *Proceedings of NAACL Workshop on Automatic Summarization*, pages 31–39, 2001.

[14] Jenny R. Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the ACL*, pages 363–370, 2005.

[15] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Texts in Statistical Science. Chapman & Hall/CRC, 2nd edition, 2004.

[16] Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. Contextual dependencies in unsupervised word segmentation. In *Proceedings of ACL*, 2006.

[17] Donna Harman. Overview of the first text retrieval conference (trec-1). In *TREC*, pages 1–20, 1992.

[18] Marti Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the ACL*, pages 9–16, 1994.

[19] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM.

[20] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of SIGKDD*, pages 168–177, 2004.

[21] Soo-Min Kim and Eduard Hovy. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL*, pages 483–490, 2006.

[22] Kevin Knight and Daniel Marcu. Statistics-based summarization - step one: Sentence compression. In *AAAI/IAAI*, pages 703–710, 2000.

[23] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th ICML*, pages 296–304, 1998.

[24] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*.

[25] P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman & Hall, 1989.

[26] Ana-Maria Popescu, Bao Nguyen, and Oren Etzioni. OPINE: Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*, 2005.

[27] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, December 1971.

[28] Simone Teufel and Marc Moens. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445, 2002.

[29] Ivan Titov and Ryan McDonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the ACL*, 2008.

[30] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. *CoRR*, abs/0801.1063, 2008.

[31] Kristina Toutanova and Mark Johnson. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Advances in Neural Information Processing Systems 20*, 2007.

[32] Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[33] Nina Wacholder, David K. Evans, and Judith Klavans. Automatic identification and organization of index terms for interactive browsing. In *JCDL*, pages 126–134, 2001.

[34] R. Wang, J. Dunnion, and J. Carthy. Machine learning approach to augmenting news headline generation. In *Proceedings of the IJCNLP*, 2005.