University of Massachusetts Dartmouth Department of Computer and Information Science

SLTM: A Sentence Level Topic Model for Analysis of Online Reviews

A Thesis in Computer Science By

Yuhan Zhang

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science

Fall 2015

I grant the University of Massachusetts Dartmouth the non-exclusive right to use the work for the purpose of making single copies of the work available to the public on a not-for-profit basis if the University's circulating copy is lost or destroyed.

Yuhan Zhang

Date

We approve the thesis of Yuhan Zhang

Date of Signature

Haiping Xu

Associate Professor of Computer and Information Science Thesis Advisor

Xiaoqin Zhang

Associate Professor and Graduate Program Director Department of Computer and Information Science Thesis Committee

Firas Khatib

Assistant Professor of Computer and Information Science Thesis Committee

Jan Bergandy

Chairperson, Department of Computer and Information Science

Robert E. Peck

Dean, College of Engineering

Tesfay Meressi

Associate Provost for Graduate Studies

# ABSTRACT

SLTM: A Sentence Level Topic Model for Analysis of Online Reviews by Yuhan Zhang

Many eGcommerce websites use the average star rating mechanism to help customers with their buying decisions; however, such average ratings are not accurate and do not necessarily reflect the actual quality of the products. To deal with this issue, users are typically allowed to provide reviews for the products they bought. Although it is a very useful service, due to large amounts of product reviews for many similar products, a user usually has a hard time to determine which product has the most desirable features that they want. In this thesis, we propose a review summary system that uses a Sentence Level Topic Model (SLTM), which can classify sentences into different classes that correspond to different product features. Similar to the probabilistic Latent Semantic Analysis (pLSA) model and the Latent Dirichlet Allocation (LDA) model, our approach adopts the idea of introducing a hidden layer, called the topic layer, between corpus and words. The data points in the training dataset can be labeled by any user through a graphic user interface. Once a SLTM is trained, by applying the Bayes’ rule, it can output the most related topic for each sentence. We analyze the sentiment of each sentence in each product review, and count the number of reviews that like / dislike each product feature. Then we give a review summary for a list of similar products with highlighted strengths and weaknesses based on their features. By comparing the list of similar products, a user may have a much easier time to find the products that meet his/her actual needs. To demonstrate the feasibility of our proposed model, we use a case study of online products from Amazon, and show that our approach can greatly save customers’ time and help them to make better decisions on purchasing the right online products.

# ACKNOWLEDGEMENT

First and foremost I would like to express my deepest gratitude to my thesis advisor, Dr. Haiping Xu for his excellent guidance, patience and help through every step of this thesis. He consistently helped me to keep my work on track through weekly meetings and steered me in the right direction whenever he thought I needed it. He was always very quick to respond to my queries and emails. His organized approach allowed me to reach my goal in time. One simply could not wish for a better advisor and it was an absolute privilege to work with such a talented professor.

I would like to extend my sincere thanks to my committee members: Dr. Xiaoqin Zhang and Dr. Firas Khatib for sparing their valuable time and being part of the thesis committee. Their feedback and comments have been very helpful.

Finally I would like to thank my parents for being the source my strength. I undoubtedly could not have done this without their support and constant encouragement. I would also like to thank my friends for constantly inspiring me.

## TABLE OF CONTENTS

[LIST OF FIGURES v](#_TOC_250032)

1. [Introduction 1](#_TOC_250031)
2. [Related Work 5](#_TOC_250030)
   1. [Review Mining 5](#_TOC_250029)
   2. [Topic Model 5](#_TOC_250028)
   3. [Sentiment Analysis 7](#_TOC_250027)
3. [Background Knowledge 10](#_TOC_250026)
   1. [Distributions 10](#_TOC_250025)
   2. [Topic Model 17](#_TOC_250024)
      1. [Latent Semantic Analysis (LSA) 17](#_TOC_250023)
      2. [Probabilistic Latent Semantic Analysis (pLSA) 17](#_TOC_250022)
      3. [Latent Dirichlet Allocation (LDA) 19](#_TOC_250021)
   3. [Stanford CoreNLP 22](#_TOC_250020)
      1. PartGOfGSpeech Tagger 23
      2. [Sentiment Analysis Toolkit 23](#_TOC_250019)
4. [Sentence Level Topic Model 29](#_TOC_250018)
   1. [Basic Framework 29](#_TOC_250017)
   2. [Review Text Preprocessing 31](#_TOC_250016)
      1. [Review Crawler 32](#_TOC_250015)
      2. [Review Level Preprocess 32](#_TOC_250014)
      3. [Sentence Level Preprocess 33](#_TOC_250013)
   3. [Topic Model 34](#_TOC_250012)
      1. [Model Creation 36](#_TOC_250011)
      2. [Topic Identification 40](#_TOC_250010)
      3. [Incrementally Learning 49](#_TOC_250009)
   4. [Sentiment Analysis 53](#_TOC_250008)
   5. [Review Summary 54](#_TOC_250007)
5. [Case Study 57](#_TOC_250006)
   1. [Product Feature Predefine 57](#_TOC_250005)
   2. [Review Data Collection and Preprocessing 60](#_TOC_250004)
   3. [Experiment 1 63](#_TOC_250003)
   4. [Experiment 2 69](#_TOC_250002)
6. [Conclusion and Future Work 74](#_TOC_250001)

[Reference 75](#_TOC_250000)

# LIST OF FIGURES

Figure 1: example of star rating system 3

Figure 2: simplified LDA Topic Model plate notation 20

Figure 3: completed LDA topic model plate notation 21

Figure 4: example of recursive neural model 25

Figure 5: recursive neural tensor network 27

Figure 6: SLTM basic framework 30

Figure 7: SLTM detailed framework 35

Figure 8: CoreNLP Sentiment Analysis toolkit example 54

Figure 9: example product information on amazon 57

Figure 10: example of amazon customer review system 58

Figure 11: user interface for labeling data points 61

Figure 12: user interface for labeling ambiguous data points 65

Figure 13: topic model accuracy line chart 66

Figure 14: Experiment 2 comparison result 71

# Introduction

E-commerce, short for electronic commerce, is trading in products or services using computer networks, such as the Internet. Electronic commerce draws on technologies such as mobile commerce, electronic funds transfer, supply china management, Internet marketing, online transaction processing, electronic data interchange (EDI), inventory management systems, and auto mated data collection systems. Modern electronic commerce typically uses the World Wide Web for at least one part of the transaction’s life cycle, although it may also use other technologies such as e-mail.

Along with the e-commerce and its unique charm that has appeared gradually, virtual enterprise, virtual bank, network marketing, online shopping, payment and advertising, such this new vocabulary which is unheard-of and now has become as familiar to people. This reflects that the e-commerce has huge impact on the economy and society from the other side.

Amazon.com, Inc is an electronic commerce company with headquarter in Seattle Washington. It is the largest internet-based retailer in the United States. Amazon product lines include several media, apparel, baby products, consumer electronics, beauty products, gourmet food, groceries, health and personal-care items, industrial & scientific supplies, kitchen items, jewelry and watches, lawn and garden items, musical instruments, sporting goods, tools, automotive items and toys & games.

A review is an evaluation of a publication, service, or company such as a movie (a movie review), video game (video game review), musical composition (music review of

a composition or recording), book (book review); a piece of hardware like a car, home appliance, or computer; or an event or performance, such as a live music concert, play, musical theater show, dance show, or art exhibition. In addition to a critical evaluation, the review’s author may assign the work a rating to indicate its relative merit.

A consumer review refers to a review written by a consumer for a product or a service based on her experience as a user of the reviewed product. Popular sources for consumer reviews are e-commerce sites like Amazon. E-commerce sites often have consumer reviews for products and sellers separately. Usually, consumer review of a product usually comments on how well the product measures up to expectations based on the specifications provided by the manufacturer or seller. It talks about performance, reliability, quality defects, if any, and value for money.

Products on amazon.com usually have hundreds of reviews. People write reviews may be because they have some feelings to share with others. They had benefit or suffered in some way, and they want others to benefit or avoid in the same way. However, each product has hundreds or even thousands of reviews (depending on the popularity), which often takes a long time to finish reading them. Most likely, people just quickly browse some reviews and skip the others. In this case, some important information may get lost. What amazon does now is to output an average score of the item and list the number of reviews given each score. As shown in figure 1:

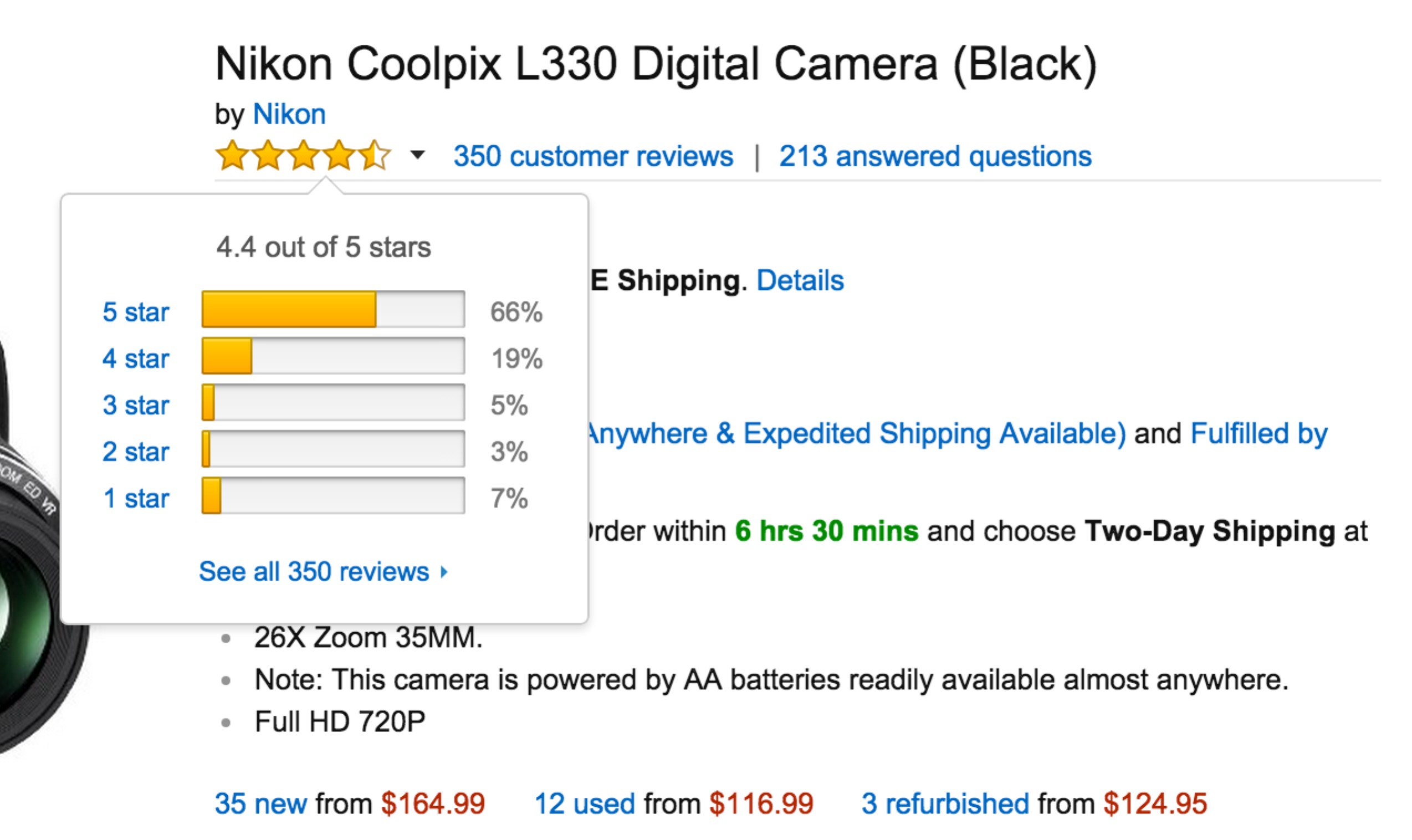


Figure 1: example of star rating system

In order to save customers’ time browsing the review and find out useful information at the same time. We want to create a system, which can analyze all the reviews automatically and output which feature of the product is good/bad. To reach this goal, we propose a way to summarize reviews. To be more specific, for each review, we analyze the sentences and draw a conclusion which product feature is this sentence describing about. Moreover, we study the sentiment of this sentence so that we can draw a conclusion: either the review author likes or dislikes a product feature. We repeat these two steps for all the reviews of a single product and count each feature is liked or disliked by how many customers.

In this paper, we proposed a Sentence Level Topic Model (SLTM), which can be used to classify sentences into different classes (each class refers to a product feature).

Same as in pLSA and LDA, this model uses the idea of introducing a hidden layer between corpus and words. By applying Bayes’ rule, SLTM outputs a topic of each sentence. Also users can help us to improve the performance of this model through a graphic user interface while using it. Furthermore, we analyze the sentiment of this sentence and count each product feature is liked/disliked by how many review authors. In this way, we are able to give a review summary to tell customers pros and cons of each product. In a case study, we show that our approach can greatly save customers’ time and help them to make better decisions while selecting products online.

With the help this system, when customer shopping online, they will quickly have a brief idea of which feature of this item is popular among other customers and which one is criticized. Then later on the summary result will be used as evidence to calculate cost-effectiveness value of this product.

The rest of the thesis is organized as follows: we discuss some related work in section 2. Section 3 discusses some background knowledge to help fully understanding our approach. Section 4 is the frame of our approach. In this section, we formally describe our approach. A case study is discussed in section 5, which will also analyzes the system accuracy. Finally, in section 7, summarize our conclusion and discuss future work.

# Related Work

## Review Mining

With the rapid growth of online reviews, review mining has attracted a great deal of attention. Pang et al. [14] employed three machine-learning approaches to label the polarity of IMDB movie reviews. Later on, they proposed to first extract the subjective portion of text with a graph min-cut algorithm, and then feed then into the sentiment classifier [15]. Rather than applying the straightforward frequency-based bag-of-words feature selection methods, Whitelaw et al. [16] defined the concept of “adjectival appraisal groups” headed by an appraising adjective and optionally modified by words like “not” or “very”. Each appraisal group was further assigned four type of features: attitude, orientation, graduation, and polarity. Turney [17] measures the strength of sentiment by the difference of the Mutual Information (PMI) between the given phrase and “excellent” and the PMI between the given phrase and “poor”. Pand et al [18], Zhang [19] attempt to determine the author’s opinion with different rating scales (i.e., the number of stars). Liu et al. build a framework to compare consumer opinions of competing products using multiple feature dimensions. After deducting supervised rules from product reviews, the strength and weakness of the product are visualized with an “Opinion Observer”.

## Topic Model

In Machine Learning and Natural Language Processing, a topic model is a type of

statistical model for discovering the abstract “topics” that occur in a collection of document. Topic model can be divided into two type of models: Discriminative model and Generative model.

Discriminative models, also called conditional models, are a class of models used in machine learning for modeling the dependence of an unobserved variable y on an observed variable x. Within a probabilistic framework, this is done by modeling the conditional probility distribution P(y|x), which can be used for predicting y from x.

Generative model is a model for randomly generating observable-data values, typically given some hidden parameters. It specifies a joint probability distribution over observation and label sequences. Generative models are used in machine learning for either modeling data directly (i.e., modeling observations drawn from a probability density function), or as an intermediate step to forming a conditional probability density function. A conditional distribution can be formed from a generative model through Bayes’ rule.

An early topic model was described by Papadimitriou, Raghavan, Tamaki and Vempala in 1998 [20]. Another one, called Probabilistic latent semantic indexing (PLSI), was created by Thomas Hofmann in 1999 [21]. Latent Dirichlet allocation (LDA), perhaps the most common topic model currently in use, is a generalization of PLSI developed by David Blei, Andrew Ng, and Michael I. Jordan in 2002, allowing documents to have a mixture of topics [22]. And there are a lot of variation algorithm published based on LDA.

## Sentiment Analysis

Sentiment analysis, also called opinion mining, is the field of study that analyzes people’s opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities. The term sentiment analysis first appeared in Nasukawa and Yi [1].

In general, sentiment analysis has been investigated mainly at three levels.

**Document level**: The task at this level is to classify whether a whole opinion document expresses a positive or negative sentiment [2] [3].

**Sentence level**: The task at this level goes to the sentences and determines whether each sentences expressed a positive, negative, or neutral opinion. This level of analysis is closely related to subjectivity classification [4].

**Entity and Aspect level**: Both the document-level and sentence-level analyses do not discover what exactly people liked and did not like. Aspect level performs finer-grained analysis [5].

From another perspective, sentiment analysis can be divided into two types: *regular opinions* and *comparative opinions* [6]. A regular opinion expresses a sentiment only on a single aspect or feature, while a comparative opinion compares multiple aspects or features.

Detecting sentiments from text usually contains two basic methodologies, which are symbolic techniques and Machine Learning techniques [7]. Symbolic Techniques

usually assume the corpus is a “bag of words” [8], which means the document is represented as a collection of words. In other words, the relationships between the individual words are lost. Kamps et al. [9] used the lexical database WordNet to determine the emotional content of a word along different dimensions. WordNet [10] is a

large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. SentiWordNet is lexical resource for opinion mining. SentiWordNet [11] is an extension of WordNet, which assigns to each synset of WordNet three sentiment scores: positivity, negativity, objectivity.

Machine Learning Techniques use a training set and a test set for classification. Training set contains some predefined input feature vectors and their corresponding class label. The training set helps to develop a discriminative model, which can be used to classify he input feature vectors into corresponding classes. Then a test data set is used to validate the model by predicting the class labels of unseen feature vectors. Some machine learning techniques like Naïve Bayes (NB), Maximum Entropy (ME), and Support Vector Machines (SVM) are used to classify reviews [12]. Term Presence, Term Frequency, negation, n-grams and Part-of-Speech are often used as input features. These features can be used to find out the semantic orientation of words, phrases, sentences and that of documents. Semantic orientation may be positive, neural, or negative. Naïve Bayes works well for certain problems [13], since one of the most important assumptions in NB is that the features are independent. However, it is proved that under a large enough training data set, the three classifiers’ preformation are almost

the same.

In this thesis, we only consider sentence level and regular opinions level sentiment analysis. The reason we do not use aspect and entity level sentiment analysis is we do aspect extraction and sentiment analysis sequentially. In the first step, we already know the sentence is describing which aspect. Therefore, the sentiment of this sentence indicates the author’s opinion of this aspect. Moreover, we used a supervised Machine Learning Naïve Bayes to create a discriminative model.

# Background Knowledge

## Distributions

* + 1. **Bernoulli Distribution**

In probability theory and statistics, the Bernoulli Distribution, is the probability distribution of a random variable which takes value 1 with success probability p and value with failure probability q = 1 – p. It can be used, for example, to represent the toss of a (not necessarily fair) coin, where “1” is defined to mean “heads” and “0” is defined to mean “tails” (or vice versa). The Bernoulli distribution is a special case of the two-point distribution, for which the two possible outcomes need not be 0 and 1.

## Binomial Distribution

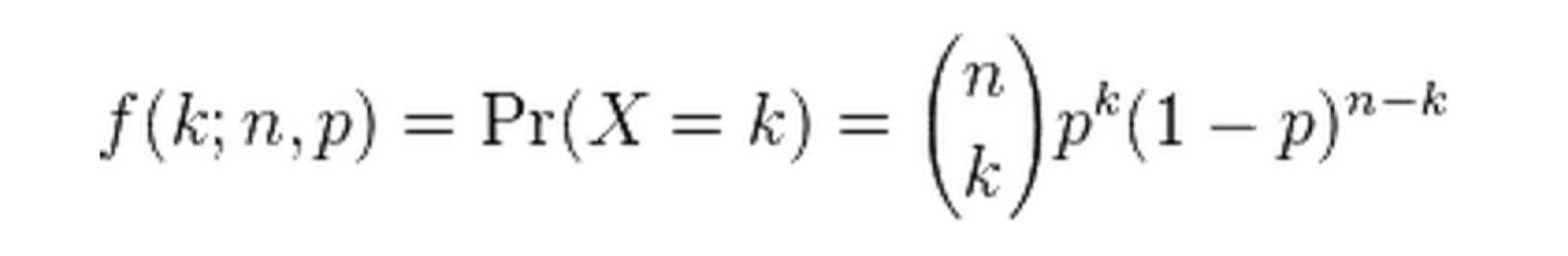
In probability theory and statistics, the Binomial Distribution with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent yes/no experiments, each of which yields success with probability p. A success/failure experiment is also called a Bernoulli experiment or Bernoulli trial; when n = 1, the binomial distribution is a Bernoulli distribution. The binomial distribution is the basis for the popular binomial test of statistical significance.

The binomial distribution is frequently used to model the number of successes in a sample of size n drawn with replacement from a population of size N. If the sampling is carried out without replacement, the draws are not independent and so the resulting

distribution is a hyper geometric distribution, not a binomial one. However, for N much larger than n, the binomial distribution is a good approximation, the widely used.

## Probability mass function

In general, if the random variable X follows the binomial distribution with parameters n and p, we write X ~ B (n, p). The probability of getting exactly k successes in n trials is given by the probability mass function:



## Multinomial Distribution

In probability theory, the multinomial distribution is a generalization of the binomial distribution. For n independent trials each of which leads to a success for exactly one of k categories, with each categories having a given fixed success probability, the multinomial distribution gives the probability of any particular combination of numbers of successes for the various categories.

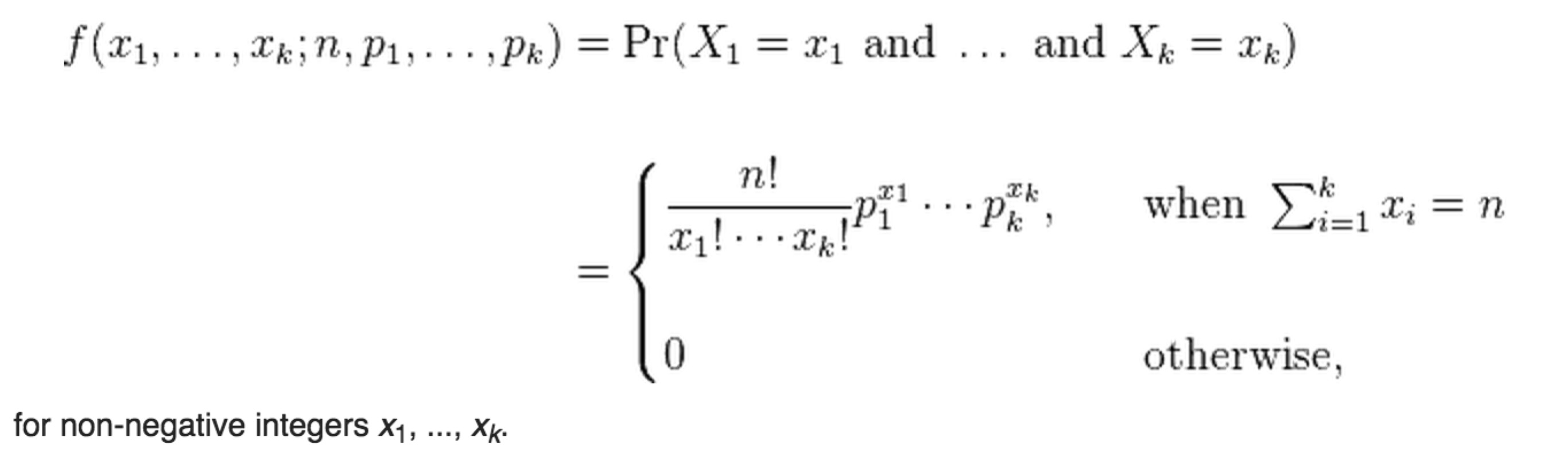
The binomial distribution is the probability distribution of the number of successes for one of just two categories in n independent Bernoulli trial, with the same probability of success on each trial. In a multinomial distribution, the analog of the Bernoulli distribution is the categorical distribution, where each trial results in exactly one of

some fixed finite number k possible outcomes, with probabilities p1, …, pk (so that pi ≥ 0 for I = 1, …, k and ∑pi = 1), and there are n independent trials. Then if the random variables Xi indicate the number of times outcome number I is observed over the n trials, the vector x = (x1, …, xk) follows a multinomial distribution with parameters n and p, where p = (p1, …, pk). Note that while the trials are independent, their outcomes X are dependent because they must be summed to n.

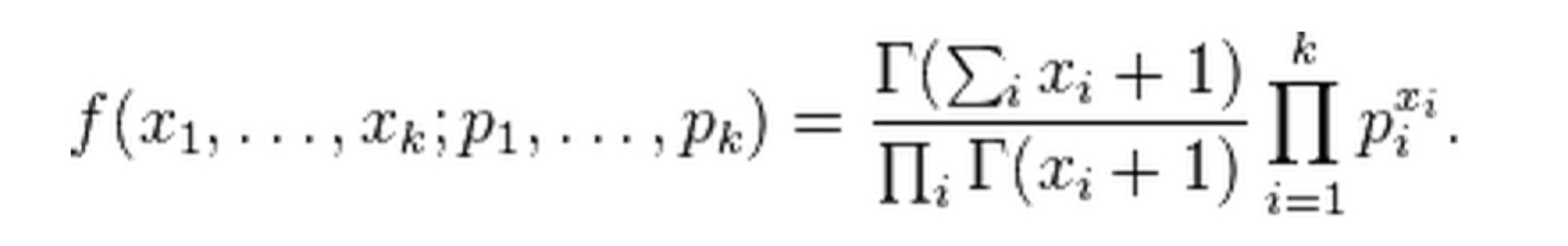
Note that, in some fields, such as natural language processing, the categorical and multinomial distributions are conflated, and it is common to speak of “multinomial distribution” when a categorical distribution a actually meant. This stems from the fae that it is sometimes convenient to express the outcome of a categorical distribution as a “1-of-K” vector (a vector with on element containing a 1 and all other elements containing a 0) rather as an integer in the range 1… K; in this form, a categorical distribution is equivalent to a multinomial distribution over a single trial.

## Probability mass function

Suppose one does an experiment of extracting n balls of k different colors from a bag, replacing the extracted ball after each draw. Balls from the same color are equivalent. Denote the variable which is the number of extracted balls of color i (i = 1, …, k) as Xi, and denote as pi the probability that a given extraction will be in color i. Let there be n balls extracted. The probability mass function of this multinomial distribution is:



The probability mass function can be expressed using the gamma function as:



This form shows its resemblance to the Dirichlet distribution, which is its conjugate prior.

## Beta Distribution

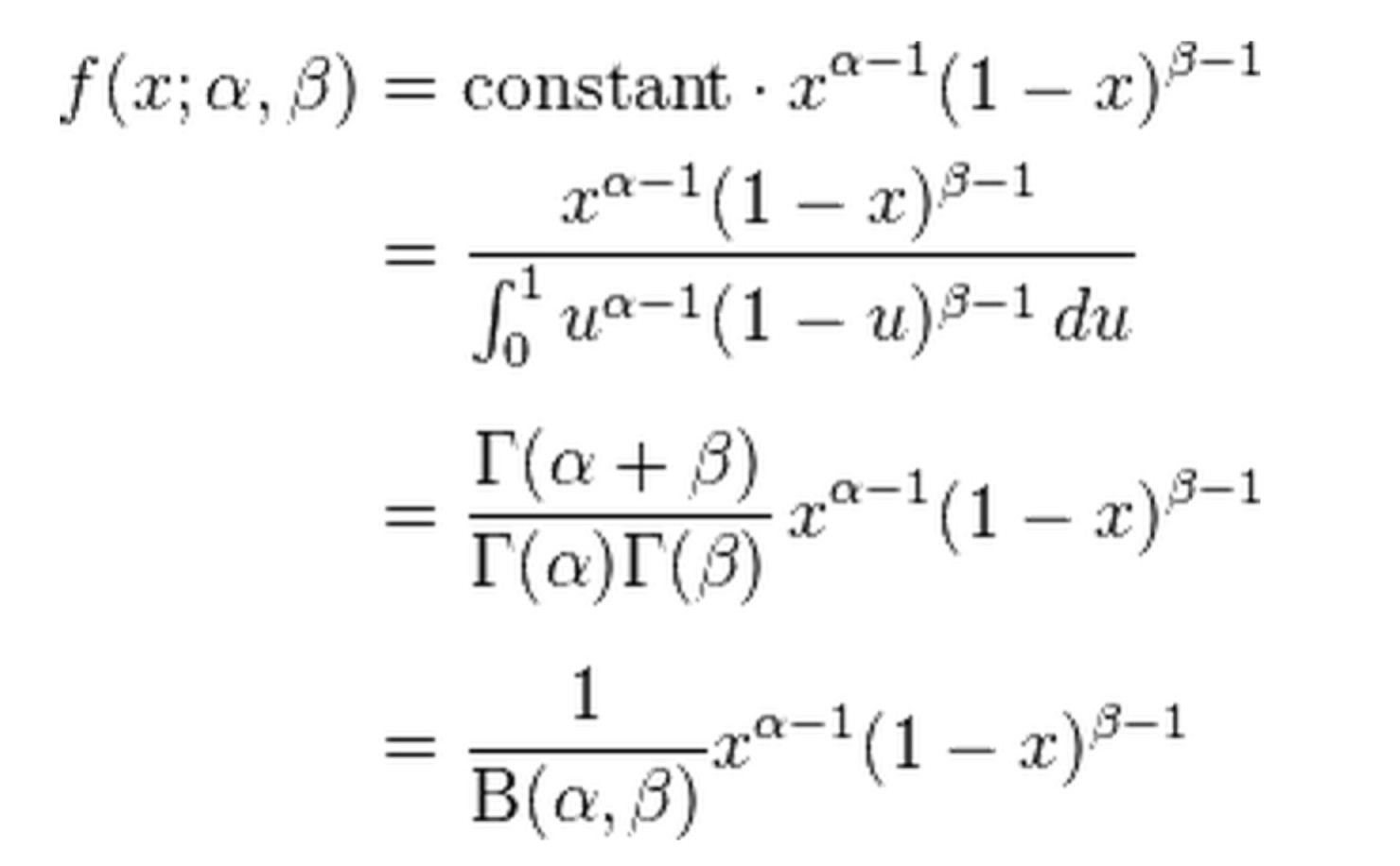
In probability theory and statistics, the beta distribution is a family of continuous probability distribution defined on the interval [0, 1] parameterized by two positive shape parameters, denoted by α and β, that appear as exponents of the random variable and control the shape of the distribution.

In Bayesian inference, the beta distribution is the conjugate prior probability distribution for the Bernoulli, binomial, negative binomial and geometric distribution. For example, the beta distribution can be used in Bayesian analysis to describe initial

knowledge concerning probability of success such as the probability that a space vehicle will successfully complete a specified mission. The beta distribution is a suitable model for the random behavior of percentages and proportions.

## Probability density function

The probability density function of the beta distribution, for 0 ≤ x ≤ 1, and shape parameter α, β > 0, is a power function of the variable x and of its reflection (1 - x) as follows:



where Γ(*z*) is the gamma function. The beta function, B, is normalization constant to ensure that the total probability integrates to 1. In the above equations x is a realization

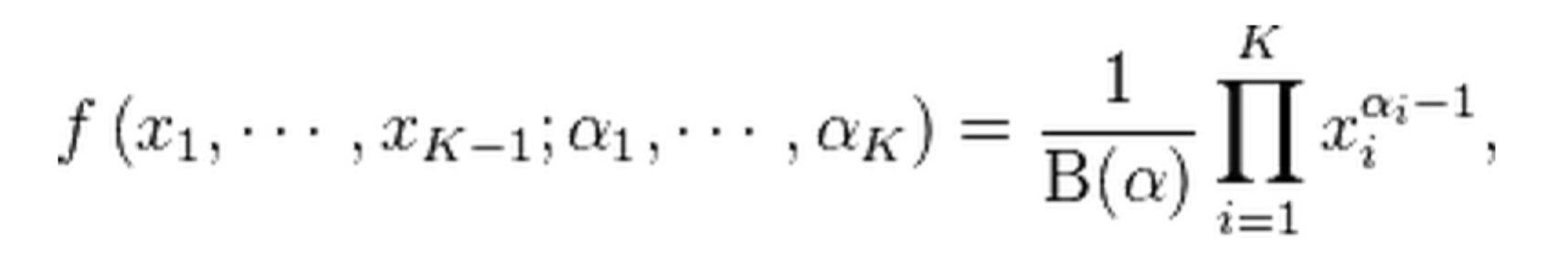
– an observed value that actually occurred – of random process X.

## Dirichlet Distribution

In probability and statistics, the Dirichlet distribution, often denoted Dir(α), is a family of continuous multivariate probability distributions parameterized by a vector α of positive reals. It is the multivariate generalization of the beta distribution. Dirichlet distributions are very often used as prior distributions in Bayesian statistics, and in face the Dirichlet distribution is the conjugate prior of the categorical distribution and multinomial distribution. That is, its probability density function returns the belief that the probabilities of K rival events are Xi given that each event has been observed αi – 1 times.

## Probability density function

The Dirichlet distribution of order K ≥ 2 with parameters α1, …,αk > 0 has a probability density function with respect to Lebesgue measure on the Euclidean space RK-1 given by:



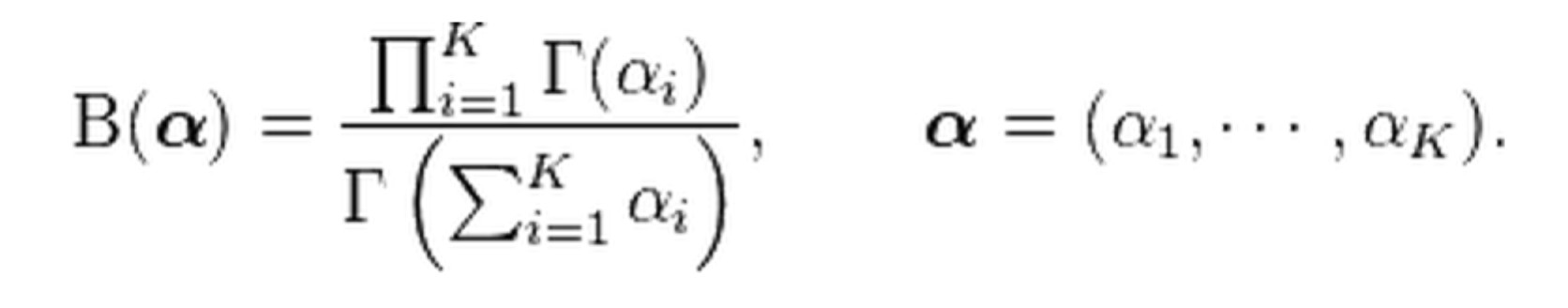
on the open (K - 1) – dimensional simplex defined by: x1, …, xk-1 > 0

x1 + … + xk-1 < 1

xk = 1 – x1 - … - xk-1 and zero else where.

The normalizing constant is the multinomial Beta function, which can be expressed in terms of the gamma function:

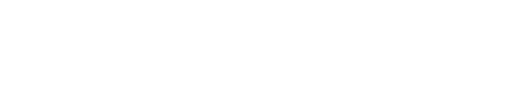
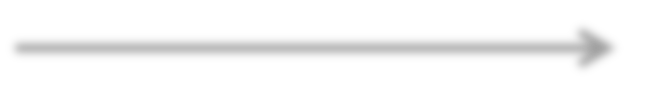
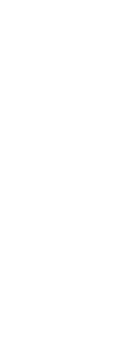
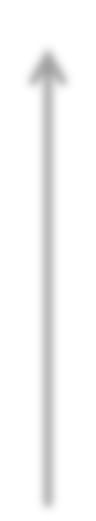
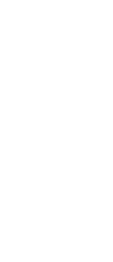
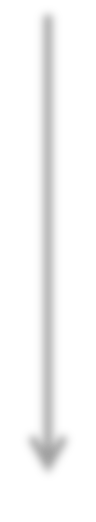
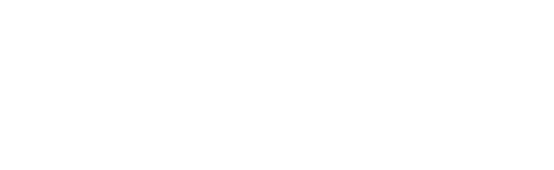
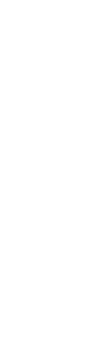
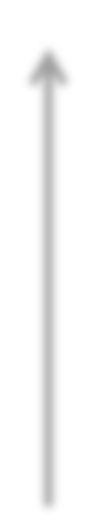
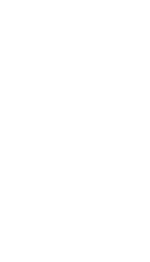
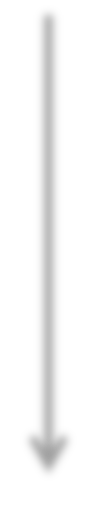
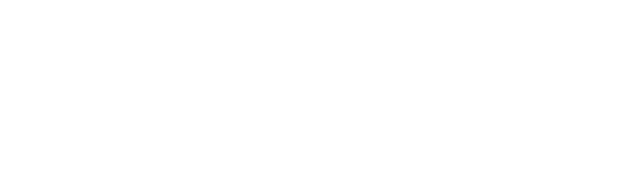
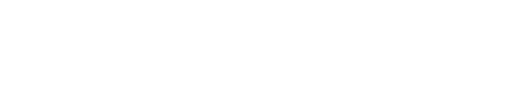
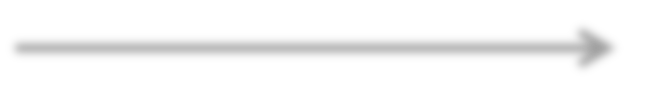
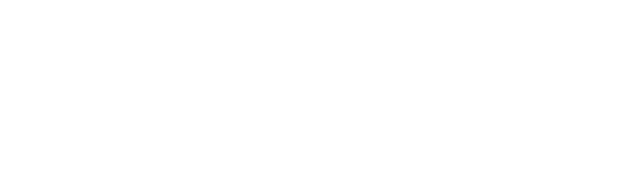
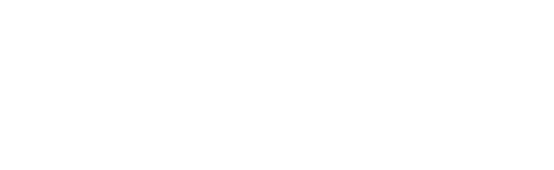
Continuous



Now we discuss the relationship between the four distributions: Binomial Distribution: B (n, p), Where n is an integer, P has two choices Multinomial Distribution: in B (n, p), if p has k (k > 2) choices

Beta Distribution: in B (n, p), if n is a real number

Dirichlet Distribution: in B (n, p), if n is a real number, p has k (k > 2) choices



Binomial Distribution

Multinomial Distribution

Beta Distribution

Dirichlet Distribution

k possible outcomes

k possible outcomes

Conjugate prior

Conjugate prior

Continuous

## Topic Model

In machine learning and natural language processing, a topic model is a type of statistical model for discovering the abstract “topics” that occur in a collection of documents.

## Latent Semantic Analysis (LSA)

LSA is a technique in natural language processing, in particular in vectorial semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text. A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. Words are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words.

## Probabilistic Latent Semantic Analysis (pLSA)

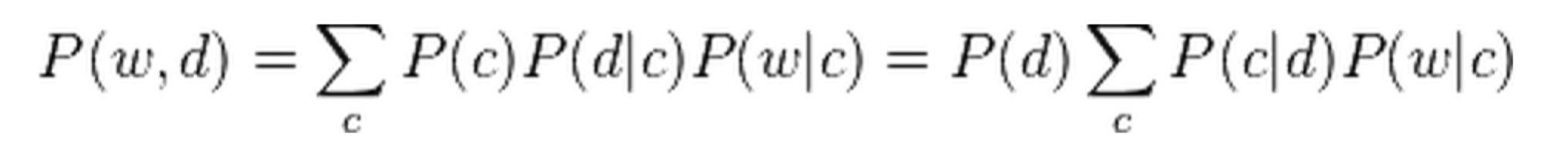
PLSA, also known as probabilistic latent semantic indexing (PLSI, especially in

information retrieval circles) is a statistical technique for the analysis of two-mode and co-occurrence data. In effect, one can derive a low-dimensional representation of the observed variables in terms of their affinity to certain hidden variable, just as in latent semantic analysis, from which PLSA evolved.

Compared to standard latent semantic analysis, which stems from linear algebra and downsizes the occurrence tables (usually via a singular value decomposition), probabilistic latent semantic analysis is based on a mixture decomposition derived from a latent class model.

## Model

Considering observations in the form of co-occurrences (w, d) of words and documents, PLSA models the probability of each co-occurrence as a mixture of conditionally independent multinomial distributions:



being c the word’s topic. The first formulation is the symmetric formulation, where w and d are both generated from the latent class c in similar ways (using the consitional probabilities P(d|c) and P(w|c)), whereas the second formulation is the asymmetric formulation, where, for each document d, a latent class is chosen consitionally to the document according to P(c|d), and a word if then generated from that class according to P(w|c). Although we have used words and documents in this example, the co-occurrence of any couple of discrete variables may be modeled in exactly the same

way.

So, the number of parameters is equal to cd + wc. The number of parameters grows linearly with the number of documents. In addition, although PLSA is a generative model of the documents in the collection it is estimated on, it is not a generative model of new documents.

## Latent Dirichlet Allocation (LDA)

In natural language processing, LDA is a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each documents is a mixture of a small number of topics and that each word’s creation is attributable to one of the document’s topics. LDA is an example of a topic model and was first presented as a graphical model for topic discovery by David Blei, Andrew Ng, and Michael Jordan in 2003.

In LDA, each document may be viewed as a mixture of various topics. This is similar to pLSA, except that in LDA the topic distribution is assumed to have a Dirichlet prior. In practice, this results in more reasonable mixtures of topics in a document. It has been noted, however, that the pLSA model is equivalent to the LDA model under a uniform Dirichlet prior distribution.

## Model

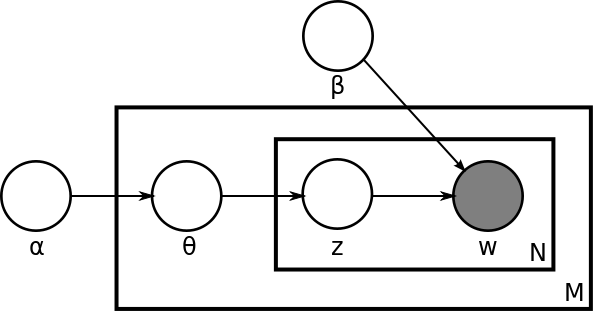


Figure 2: simplified LDA Topic Model plate notation

With plate notation, the dependencies among the many variables can be captured concisely. The boxes are “plates” representing replicates. The outer plate represents documents, while the inner represents the repeated choice of topics and words within a document. M denotes the number of documents, N the number of words in a document. Thus:

α is the parameter of the Dirichlet prior on the per-document topic distributions

β is the parameter of the Dirichlet prior on the per-topic word distribution

θi is the topic distribution for document i

φk is the word distribution for topic k

Zij is the topic for the jth word in document i Wij is the specific word

The Wij are the only observable variable, and the other variables are latent variables. Mostly, the basic LDA model will be extended to a smoothed version to gain better results.

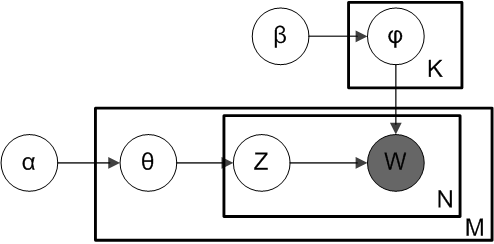


Figure 3: completed LDA topic model plate notation

The plate notation is shown above, where K denotes the number of topics considered in the model and

Φ is a K\*V (V is the dimension of the vocabulart) Markov matrix (transition matrix), and each row of which denotes the word distribution of a topic.

The generative process is as follows. Documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA assumes the following generative process for a corpus D consisting of M documents each of length Ni:

1. Choose θi ~ Dir(α), where i €{1, …, M} and Dir(α) is the Dirichlet distribution for

parameter α

1. Choose ~ Dir(β), where k €{1,…, K}
2. For each of the word position i, j, where j€{1, …, Ni}, and i € {1, …, M}
   1. Choose a topic Zi,j ~ Multinomial(θi)
   2. Choose a word Wi,j ~ Multinomial(φZi,j)

(Note that the Multinomial distribution here refers to the Multinomial with only one trial. It is formally equivalent to the categorical distribution.)

The lengths Ni are treated as independent of all the other data generating variables (q and z). The subscript is often dropped, as in the plate diagrams shown here.

## Stanford CoreNLP

Stanford CoreNLP [23] provides a set of natural language analysis tools which can take raw text input and give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, indicate sentiments, etc. Stanford CoreNLP is an integrated framework. Its goal is to make it very easy to apply a bunch of linguistic analysis tools to a piece of text. Starting from plain text, you can run all the tools on it with just two lines of code. It is designed to be highly flexible and extensible. With a single option you can change which tools should be enabled and which should be disable. Its analyses provide the foundational building blocks for higher-level and

domain-specific text understanding applications.

## Part-Of-Speech Tagger

A Part-Of-Speech Tagger (POS Tagger) [24][25] is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, adjective, etc. The English taggers use the Penn Treebank tag set.

## Sentiment Analysis Toolkit

Initially, Sentiment Analysis Toolkit was designed for predicting the sentiment of movie reviews. In previous research, most sentiment prediction systems look at isolated words, giving them positive points or negative points based on certain Sentiment Dictionary such as SentiWord Net. Then sum up these points with consideration of negative words. In that way, the order of words is ignored and some information is lost. In contrast, according to the authors, their model utilizes deep learning approach, which actually builds up a representation of whole sentences based on the sentence structure. Their model is built on phrase level rather than word level (though phrase is built on word level), so that the model computes the sentiment based on how words compose the meaning of longer phrases. This way, this model is not as easily fooled as previous models.

Let’s take an example from their toolkit main webpage:

*“This movie was actually neither that funny, nor super witty”*

Even though, “funny” and “witty” are two positive but the whole sentence is still negative.

The underlying technology of this model is a new type of Recursive Neural Network that builds on top of grammatical structures. The training data set of this model is from Stanford Sentiment Treebank. It is worth mentioning that users can help the authors to improve this model while using it. The model’s accuracy on a single sentence positive/negative classification is between 80% up to 85.4%. While on a

fine-grained sentiment, labels for all phrases reaches 80.7%, an improvement of 9.7% over bag of features.

## Stanford Sentiment Treebank

It is the first corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. The data set of it is introduced by Pang and Lee (2005) and consists of 11,855 single sentence extracted from movie reviews. The corpus was parsed by Stanford parser and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges. This dataset allows us to capture more complex linguistic phenomena.

## Recursive Neural Model

In this section, we will briefly introduce the algorithm of Stanford Sentiment Analysis Toolkit. For more details, please check the original paper Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank.

In Recursive Neural Model, each word is represented as a d-dimensional vector.

Then we can use the word vectors as input parameters to a *softmax* classifier. The toolkit is a five classes classifier, namely “very negative”, “negative”, “neutral”, “positive”, and “very positive”.

ya = softmax(Wsa)

where Ws€R5\*d is the sentiment classification matrix. For example:

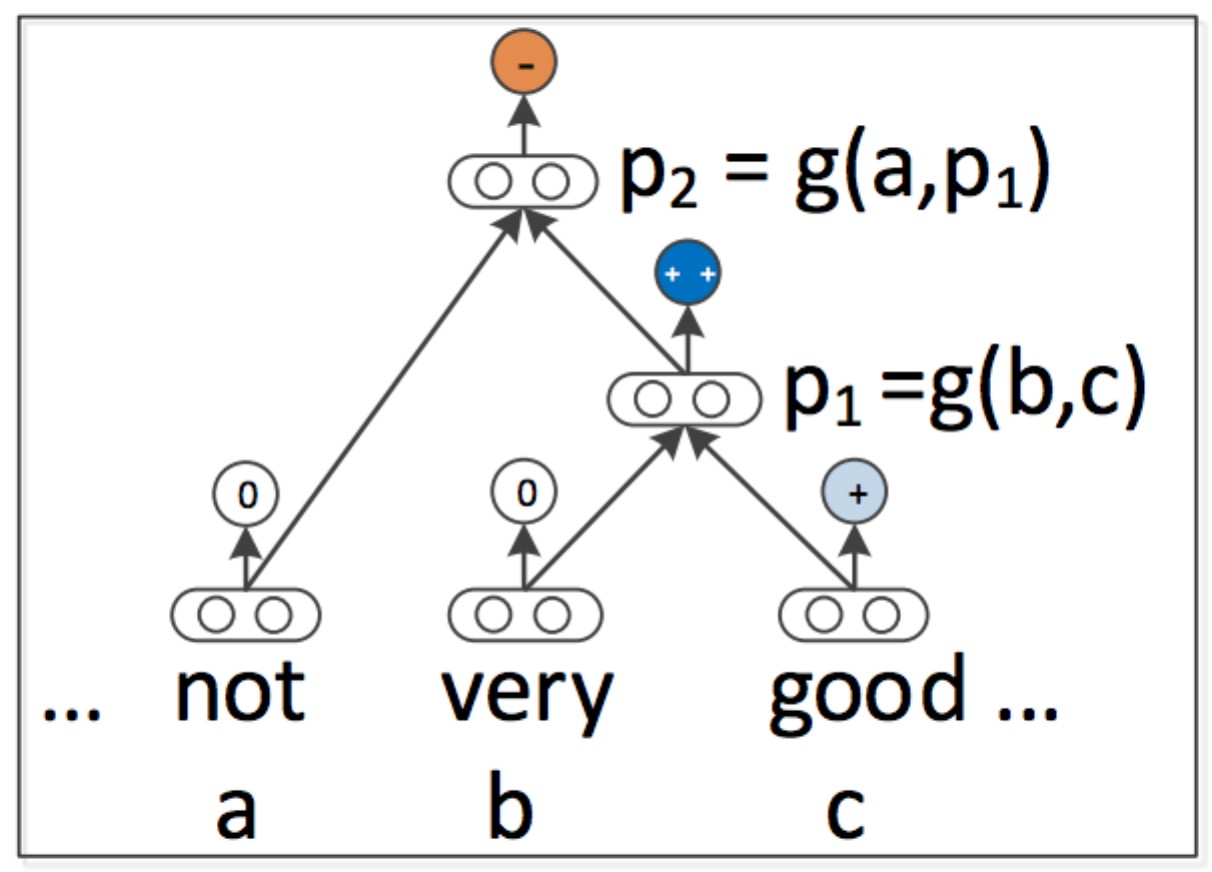
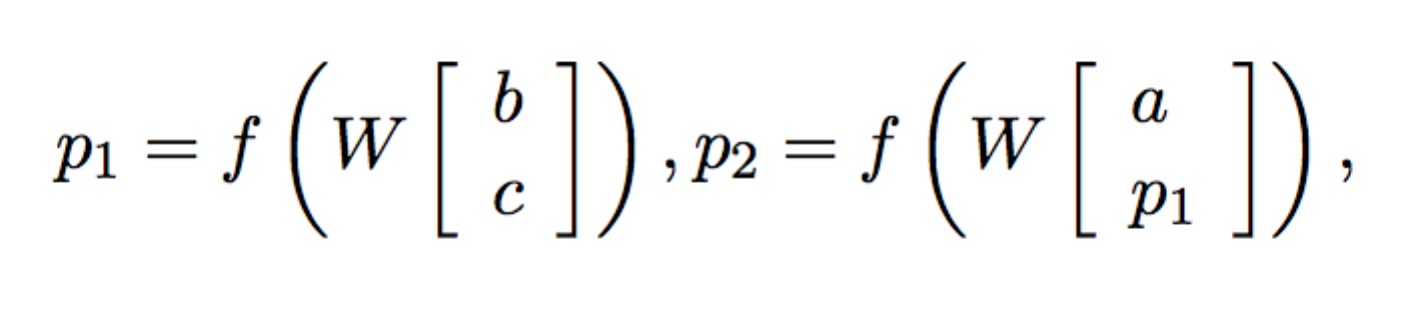


Figure 4: example of recursive neural model

in the given tri-gram, let ya be the sentiment class of word “not”, and repeat this for yb, yc. The main task and difference is how to compute pi in a bottom up fashion. Now we introduce a basic model and another one used in Stanford Sentiment Analysis.

## RNN: Recursive Neural Network

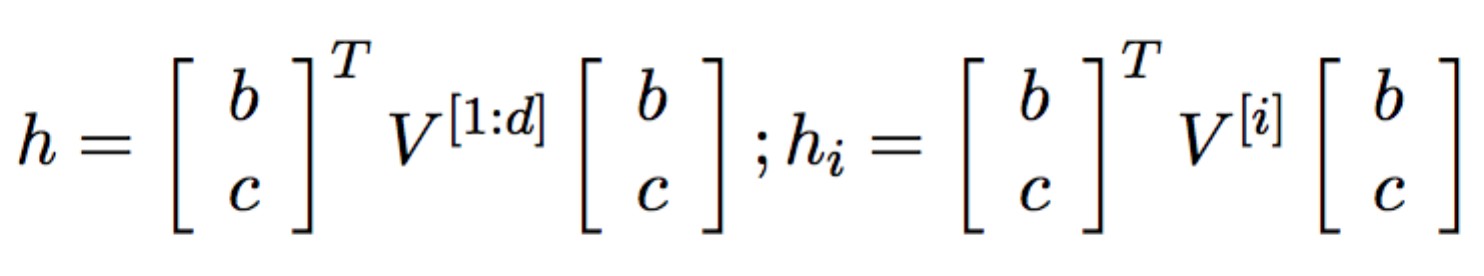
It is determined which parent already has all its children computed. In the above example, RNN uses the following equations to compute the parent vectors:



where f = tanh is a standard element-wise nonlinearity, W€Rd\*2d is the main parameter to learn and the bias are omitted for simplicity. In this way, the information between words are lost since W calculates b and c respectively.

## RNTN: Recursive Neural Tensor Network

RNTN introduces a single tensor layer h€Rd to solve this problem.



where V[i]€Rd\*d. The whole formula is shown below:

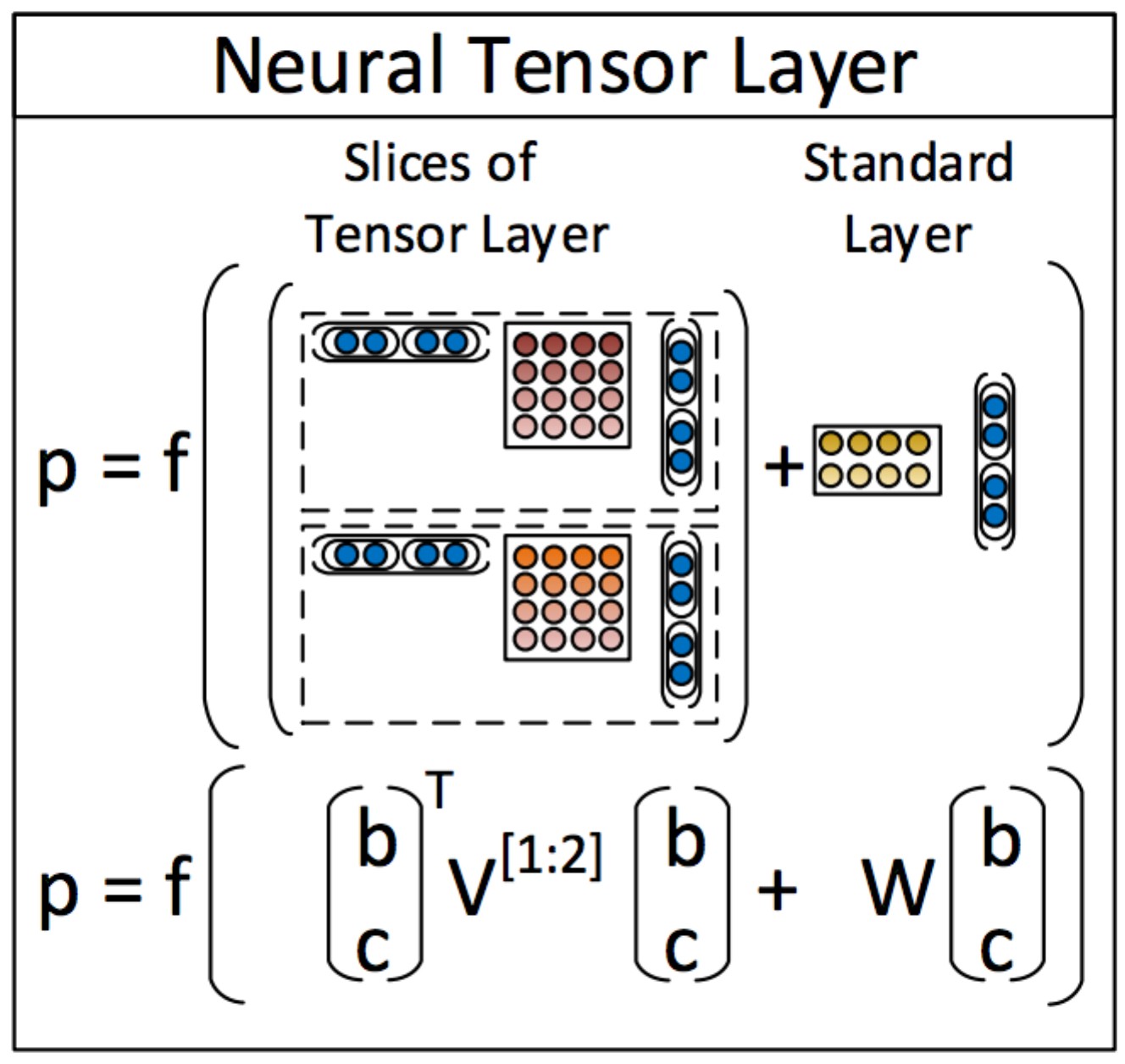
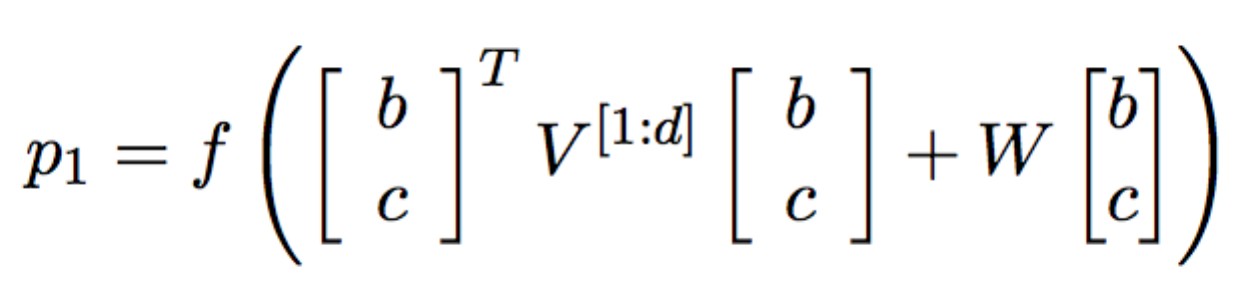
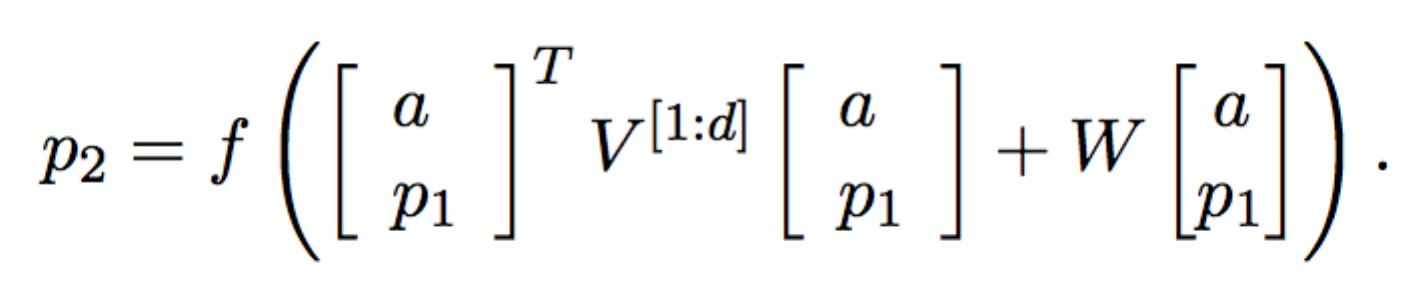


Figure 5: recursive neural tensor network

The RNTN uses this definition for computing p1:



where W is the same as defined in the previous models. The next parent vector p2 in the above example will be computed with the same weights:



Note that, when V is set to 0, RNTN is exactly the same as RNN. The main advantage

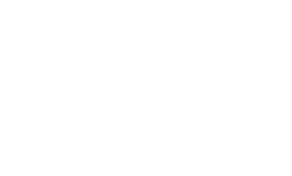
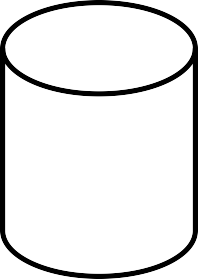
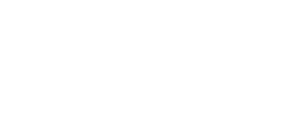
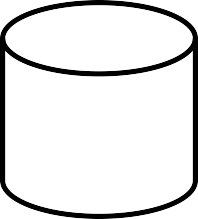
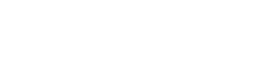
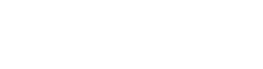
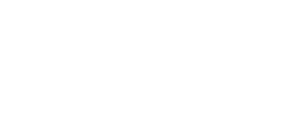
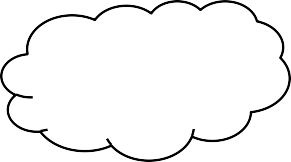
of RNTN is that the tensor V can directly relate input vectors. Such that we can interpret each slice of the tensor as capturing a specific type of composition.

# 4. Sentence Level Topic Model

In this section, we will introduce how our proposed system could verify the quality of a product, which is reflected by its features based on the information collected from e-commerce websites such as Amazon.

## Basic Framework

The basic framework of this system consists of four major parts, which are Text Preprocess, Topic Extraction, Sentiment Analysis, and Summary. The system is flexible and highly modularized. Both Topic Extraction part and Sentiment Analysis part can be used independently. In other words, Topic Extraction part can be used to detect topics in a conversion or any kind of documents. However, these two parts must be used in a sentence level corpus. The framework is shown below:



Item

Reviews

Text Preprocess

Topic

Feature Extraction

Sentiment Analysis

Sentence 1

…

Sentiment

Analysis

Feature Preference Combination

Review r

feature 1 score

feature 2 score

…

feature k score

Feature Extraction Sentiment Analysis

Sentence n

Figure 6: SLTM basic framework

As the graph shows, the input of this system is an item’s Amazon Standard Identification Number (ASIN). While the output is for each predefined item features how many customers like it and how many customers dislike it. We will discuss each part in details later. Now we focus on the transmission between them.

Text Preprocessing part split the raw reviews into few sentences, and output each

sentence to Feature Extraction sequentially. In Feature Extraction part, we use a LDA based topic model to calculate the probability of each topic. It will determine the features mentioned in this sentence and output the result. If the sentence does contain a topic, then output this sentence to Sentiment Analysis toolkit, else Topic Extraction ignore this sentence. Regarding Sentiment Analysis part, it outputs an integer between 0

~ 4 with 0 represents very negative and 4 represents very positive.

To identify product features and analyze sentiment, we first collect the reviews for a certain product of certain category P (e.g. Camera). Then we define some customer most interested features (in this thesis, we use the term “topic” and “features” interchangeably) of this category. Note that customer may use various words or phrase to describe a feature. For example, words or phrases: “small”, “fit in my pocket”, “handy”, etc are all point to feature “size”. In Topic Extraction section, we will talk about how to detect the product features in depth. Likewise, customers also express their feelings in different ways.

## Review Text Preprocessing

Text Preprocessing is an import part in NLP. It normalizes input text to a certain form, which will be used as input to subsequent modus. In our system, preprocessing part contains three sub-parts namely Review Crawler, Review Level Preprocessing, and Sentence Level Preprocessing.

## Review Crawler

Review Crawler is a simple algorithm downloading all the reviews of a product on amazon.com and saving to local disk for future use. Amazon does provide some API to retrieve customer reviews for advertising product use. However, since amazon API may not be stable or change over time, we use JAVA HttpURLConnection to gather reviews directly.

Amazon Standard Identification Number (ASIN) is a 10-character alphanumeric unique identifier assigned by Amazon.com and its partners for product identification within the Amazon.com organization. We use this number to identify items on amazon.com.

## Review Level Preprocess

Review module keeps track of a single review. With the help of Standford CoreNLP sentence split toolkit, raw reviews are split into sentences in this module. Note that CoreNLP provides a set of natural language process tool and we will utilize it a few times in our thesis. We ignore the sentences with less than three words since a meaningful sentence should at least has a topic word, sentiment word and a sentence end mark. A Review usually contains a list of sentences.

## Sentence Level Preprocess

Again, we use Standford CoreNLP Part-Of-Speech tagger to indicate each word in this sentence as noun, verb, adjective, adverb or other tags. Here we use an example to explain how POS tagger parses a sentence. Let the sentence be

*“The display is nice and big.”*

By using the CoreNLP toolkit, the parser outputs the sentence with the following POS tags:



In the above tagged sentence, the tags DT, NN, VBZ, JJ, and CC stand for “Determiner”, “Noun”, “Verb, third person singular present”, “Adjective”, “Coordinating Conjunction”, respectively. After detecting all sentences in a review and parsing each sentence with POS tags, we can save each sentence along with the POS tag information into local disk. This information will be used to identify product features in the following sections.

The reason we do POS tag is, based on our observation, a topic in a sentence are usually determined by nouns or adjectives. Therefore, we use POS tag to get rid of those noise data (words such as “You”, “I”, “a”, “the” etc.). Same as in LDA, we assume the sentences are “bag of words”. Finally we output the word with tag NN, JJ to Topic Extraction Section.

Moreover, for each word, we consider the “lemma” (original form) only. For

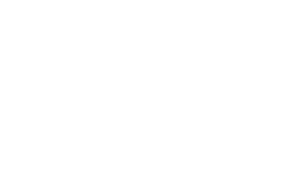
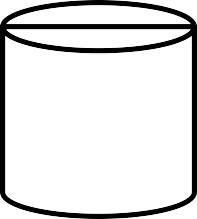
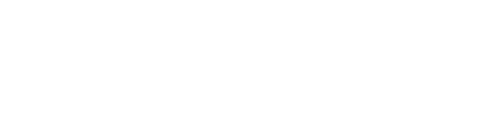
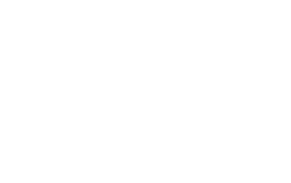
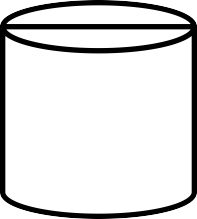
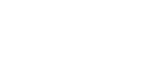
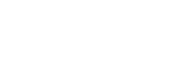
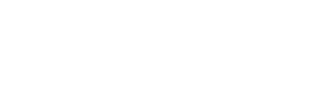
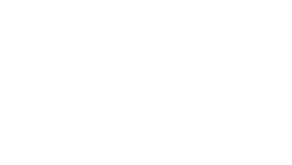
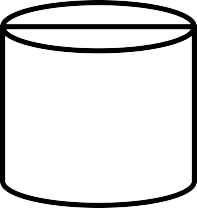
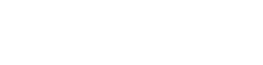
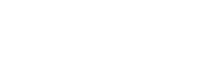
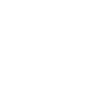
example: “likes”, “like”, “liked” are all considered as “like”. CoreNLP Lemma toolkit will help us do this work.

## Topic Model

In this section, we will discuss how to create and use this model in details. As mentioned in previous section, we first define some customer most interested features in this section. These predefined features are considered as different classes (assume we have K classes). Then we create a topic\_word matrix with K (0 ~ K-1) row and V size column, where V is a dynamic number and will be increased by 1 every time a new word appears. Then we manually label some sentences belonging to which class. Finally we count topic word co-occurrence and update the matrix. The figure below shows the framework of this model:

…

Figure 7: SLTM detailed framework



P(f1)

P(f2)

P(fk)

Determine Feature

Sentence

Review

Predefined Features

Database

CoreNLP POS Toolkit

F1 count

*adjectives*

*nouns*

F2 count

Topic Word Count Matrix

Fk count

Topic Model

Unlabeled

Labeled

Determined?

No

Sentence

Sentence

Yes

Output feature

Manually label feature

From this figure we can see that topic model accepts only *nouns* and *adjectives* as input. Then first, retrieve topic word count matrix for these words’ frequency for each

feature. Second, calculate the probability of this sentence being each class. Third, determine the sentence feature based on the probabilities: if one of the probabilities is much higher than others, then we claim this sentence is describing the corresponding feature and output this feature; else (in this case, several probabilities are approximately evenly high) we save this sentence to out unlabeled sentence database. Finally, we let human manually determine the feature of sentences in *unlabeled sentence* dataset, then output the feature and save the sentences to *labeled sentence* dataset, which will be used as training dataset to create the matrix.

## Model Creation

1. **Predefine features**

For any category product P, we need to define a fixed number of features as classes. These K features are kept in a List named **topic** (we use topic and feature interchangeable in this thesis). We also have another List named **vocabulary,** which is used to keep track of all words. Note that there is no duplicated word in vocabulary.

## Matrix Creation

Now we create a topic\_word matrix named Φ (same as in LDA Model) with K rows and 0 column. Note that this is an empty matrix at beginning because the vocabulary is empty. But we will expand Φ to a two-dimensional matrix soon. The reason we use the word matrix here is to explain the meaning of rows and columns in this matrix clearly. Each row of this matrix represents the distribution of words in this

topic. While, each column represents the word co-occurred with different topics.

Φk : distribution of words in topic k (k ranges from 0 ~ K-1)

Φk, w: counts of word w occurring in topic k (w ranges from 0 ~ V-1)

The matrix is used to save topic word co-occurrence information. And each cell in the matrix is initially set to 0. We save Φ to a local file for future use.

## Updating the Matrix

Now we need to expand the column (vocabulary) in matrix Φ and increase the number in each cell of Φ. First we manually browse some reviews and determine each sentence in the review belongs to which class (describing which feature).

For example: *“The display is nice and big.”* is describing features “view\_screen”, so this sentence belongs to class “display”. Then we put a “#view\_screen” at the end of this sentence. Same idea:

*“The display is nice and big.”* #view\_screen

*“They had the best price on the camera, and got it here on time!”* #price

*“It is also much easier to see camera settings on the big LCD.”* #view\_screen

We save these labeled sentences to our training data, which will be used to update the matrix. However, as we mentioned in Text Preprocessing section, only *nouns* and *adjectives* will be considered as input for Topic Model section. For each preprocessed sentence, we call it a **Data Point**. So in the three samples above, the Data Points are:

Data Point 1: *“display, nice, big”* #view\_screen Data Point 2: *“best, price, camera, time”* #price

Data Point 3: *“be, easy, camera, setting, big, LCD”* #view\_screen The algorithm for updating the matrix is shown as follows:

## Algorithm: Update Matrix Φ

**Input**: Matrix Φ, topic List, vocabulary List, and a set of data points

**Output**: updated matrix Φ

* 1. Load matrix Φ from a local file
  2. **for each** data point *s*
  3. let *k* be the feature of *s*, and *topicIndex* be the index of *k* in *topic list*
  4. **for each** word *w* in *s*
  5. **if** *vocabulary* contains *w*
  6. **then** let *wordIndex* be the index of *word* in *vocabulary list*

## else

* 1. add *w* to vocabulary, add 1 column to Φ, and let *wordIndex* equals to (|*vocabulary|*-1)
  2. increase Φ[*topicIndex*][*wordIndex*] by 1
  3. store matrix Φ

In this algorithm, we first load Φ from local file. Then for each data point s, we let k be s’s feature. For each word w in s, retrieve vocabulary list for w. If w is in

vocabulary, let wordIndex be the index of w. Else, w is an out of vocabulary word. In this case, we add w to vocabulary and one column to Φ. The word w is at the end of vocabulary list now, and we let wordIndex equals to vocabulary size minus one. Finally we increase Φ[*topicIndex*][*wordIndex*] by one. At the end of this algorithm, we save matrix again to that local file. In this way, we expand vocabulary size and update the matrix gradually.

Now we use an example to explain the algorithm. For the 3 data poings above, initially, the topic\_word matrix is:

|  |
| --- |
|  |
| Feature 0 |
| Feature 1 |
| View\_screen |
| Price |
| … |
| Feature K-1 |

Retrieve vocabulary list for word “display”. The vocabulary does not contain this word, then add this word to vocabulary. Increase the topic\_word matrix column by 1. Then increase cell topic\_word[view\_screen][display] by 1. Now the matrix is like:

|  |  |
| --- | --- |
|  | display |
| Feature 0 | 0 |
| Feature 1 | 0 |

|  |  |
| --- | --- |
| View\_screen | 1 |
| Price | 0 |
| … | 0 |
| Feature K-1 | 0 |

We repeat this step for all the words in the 3 data points. Finally, we will have the matrix like this:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | display | nice | big | best | price | camera | time | be | easy | setting | LCD |
| Feature 0 |  |  |  |  |  |  |  |  |  |  |  |
| Feature 1 |  |  |  |  |  |  |  |  |  |  |  |
| View\_screen | 1 | 1 | 2 |  |  | 1 |  | 1 | 1 | 1 | 1 |
| Price |  |  |  | 1 | 1 | 1 | 1 |  |  |  |  |
| … |  |  |  |  |  |  |  |  |  |  |  |
| Feature K-1 |  |  |  |  |  |  |  |  |  |  |  |

Where all the blank cells are 0.

## Topic Identification

Once we have updated the matrix from all the data points in training dataset, we can use the matrix as a discriminative model to classify sentences into different class.

The foundation of this model is Bayes’ theorem.

Let T0, …, TK-1 be the corresponding topic;

Then P(Tk|S) represents the probability of topic Tk given sentence S, where k € 0

to K-1, note that

!!!

!!! P(Tk|S)

= 1.

Now we only consider the calculation of probability of sentence S being topic k.

The calculation procedure is same for other topics. From Bayes’ rule, we know:

P(Tk|S) =

!(!|!") ∗ !(!")

!(!)

(1)

Moreover, we assume S is “bag of words” and use unigram meaning words are independent from each other. Then S can be represented as W1, …, Wn. Therefore formula (1) can be represented as:

P(Tk|S) =

!(!|!") ∗ !(!") =

!(!)

!(!!,…,!")

! !!,…,!" !" ∗!(!")

(2)

Now we convert the problem of calculating P(Tk|S) to three pieces of small tasks, which are calculation of P(W1,…, Wn|Tk), P(Tk), and P(W1, …, Wn).

## 1) Calculation of P(W1,…, Wn|Tk)

1. **Apply Bayes’ rule**

Since we assume W1, …, Wn are independent from each other, so P(W1,…, Wn|Tk) = P(W1|Tk) \* P(W2|Tk) \* … \* P(Wn|Tk)

Where P(Wn|Tk) indicates the probability of Wn given Tk. And P(Wn|Tk) can be easily calculated from matrix Φ.

As we mentioned in above section: Φk is the distribution of words in topic k. So P(Wn|Tk) = Φ[k][n] / Sum(Φ[k])

Where Sum(Φ[k]) is the sum of word counts in topic k.

Sum(Φ[k]) = Φ[k][0] + Φ[k][1] + … + Φ[k][V-1]

Therefore

P(W1,…, Wn|Tk) =

![!][!]

!"#![!]

\* … \*

![!][!]

!"#![!]

However, in real data,

![!][!]

!"#![!]

is usually a small decimal and n could be a large

number (if the sentence is complicated and pretty long). In this case, during the

procedure of calculating

![!][!]

!"#![!]

\* … \*

![!][!]

!"#![!]

, an accuracy overflow error

may happen. To solve this problem, we calculate !" ![!][!]

!"#![!]

for each word wn

instead of calculating

![!][!]

!"#![!]

directly. Let

P’(W1,…, Wn|Tk) = ln(P)

Therefore:

P’(W1,…, Wn|Tk) = ln(

![!][!]

![!][!]

\* … \*

![!][!]

![!][!]

) = !"

Φ[k][1]

Φ[k][0]

+ … + !"

Φ[k][n]

Φ[k][0]

P(W1,…, Wn|Tk) = !!’(!!,…,!!|!!) (3)

## Smoothing

However, word Wn may have not appeared in topic k in training set, that means Φ[k][n] is 0. Since one of the factors in the formula is 0, we know P(W1,…, Wn|Tk) eventually will be 0 too. To solve this problem, now we briefly introduce three smoothing algorithms, which are:

## Additive Smoothing

Additive Smoothing is also called Laplace smoothing, which is adding a constant number (usually 1) to all the numerators and adding n to the denominator. Such that all the numerators are greater than 0 and the denominator is still the sum of all the numerators. The formula is shown below:

P(W1,…, Wn|Tk) =

! ! ! !!

! ! ! !!

\* … \*

! ! ! !!

! ! ! !!

## Good-Turing Smoothing

This is a statistical technique for estimating the probability of encountering an object of a hitherto unseen species, given a set of past observations of objects from different specifies []. In our problem, the species refers to words.

First we define some notation and required data structures:

1. Assuming in w1 … wn, X distinct species have been observed, numbered 1… X.
2. Then the frequency vector, R has elements Rx that give the number of individuals that have been observed for word x.
3. The frequency of frequencies vector, (Nr) r=0,1, …, shows how many times the frequency r occurs in the vector R; i.e. among the elements Rx.

For example N1 is the number of words for which only one individual was observed. Note that the total number of objects observed, N, can be found from

∝

!!!

!"!

The first step in the calculation is to find an estimate of the total probability of unseen words. This estimate is

!0

!0 = !

The next step is to find an estimate of probability for species, which were seen r times. For a single species this estimate is:

! + 1 ∗ !(!" + 1)

!"(!")

To estimate a probability of encountering any species from this group (i.e., the group of species seen r times) one can use the following formula:

! + 1 ∗ !(!" + 1)

!

Here, the notation S() means the smoothed or adjusted value of the frequency shown in parenthesis.

## 3) Kneser-Ney Smoothing

Kneser-Ney smoothing is a method primarily used to calculate the probability distribution of n-grams in a document based on their histories. This algorithm gives a smallest number in Φ[k] to Φ[k][p] if Φ[k][p] is 0. And deduct this number by subtracting a fixed number from other Φ[k][p] if Φ[k][p] is not 0. For example, initially in matrix Φ, we assume

Φ[k][min] is the smallest number in Φ[k] Φ[k][z1] = Φ[k][z2] = … = Φ[k][zm] = 0

Φ[k][y1] ≠ 0, Φ[k][y2] ≠ 0 … Φ[k][yn] ≠ 0 By using Kneser-Ney algorithm, we let

Φ[k][z1] = Φ[k][z2] = … =Φ[k][zm] = Φ[k][min]

In the step above, we add totally (m \* Φ[k][min]) to Φ[k]. To keep the sum of Φ[k] unchanged, we need to deduct this number from Φ[k][y1], Φ[k][y2] … and Φ[k][yn].

More specifically, we deduct a same number from Φ[k][y1], Φ[k][y2] … and Φ[k][yn]

which is

! ∗ ![!][!"#].

!

Φ[k][yn] = Φ[k][yn] -

! ∗ ![!][!"#]

!

Note that Φ[k][yn] ≥ Φ[k][min], and in real data: n >> m. So Φ[k][yn] > 0.

Although Kneser-Ney smoothing algorithm is widely considered the most effective method for n-grams, our system is based on unigram. We use a combined algorithm based on both add one and Kneser-Ney smoothing. Our detailed algorithm is shown below:

## Algorithm: Smoothing Algorithm

**Input**: vector *Φ[k]*, sentence *S* (a list of words *W1, …, Wn*), *vocabulary* list

**Output**: smoothed probability vector *P* from *P(W1)* to *P(Wn)*

1. Let *n/m* represent the number of how many cells in *Φ[k]* equal to 0/not equal to 0

respectively. Initially set n and m to 0.

1. For each integer *Φ[k][x]* in *Φ[k]*
2. If *Φ[k][x]* equals to 0
3. then increase *n* by 1
4. else
5. increase *m* by 1
6. let *offset* = *n/m \** 1
7. for each *wn* in *S*
8. let *x* be the index of *wn* in *vocabulary*
9. if *Φ[k][x]* equals to 0 then let *p(wn)* equals to 1*/SumΦ[k]*
10. else let *p(wn)* equals to *(Φ[k][x] – offset)/SumΦ[k]*
11. output *P*

In this algorithm, we add 1 to all 0 elements just like add-one smoothing algorithm and deduct a constant number from other none 0 elements just like Kenser-Ney smoothing algorithm. In this way, we get rid of 0 factors in calculation of P(W1,…, Wn|Tk). So that the result will be between 0 and 1.

## Calculation of P(Tk)

P(Tk) represents the probability of topic k appearing. We let N(tk) indicate the number of topic k appeared times. Then

P(Tk) =

!(!")

! !! !⋯!!(!")

where N(T1) + … + N(Tn) is the total number of data points, also sentences.

## Calculation of P(W1W2 … Wn)

Again from Bayes’ rule, we know

P(W1W2 … Wn) = P(W1) \* P(W2) \* .. \* P(Wn)

P(Wn) is the probability of word n appearing. Retrieve vocabulary list, let xn be the index of Wn in *vocabulary*. Note that SumΦ is the total number of words in data points. Therefore

p(Wn) =

![!][!"]

!"#!

Moreover

P(W1W2 … Wn) =

![!][!"]

!"#!

![!][!"]

\*

!"#!

\* … \*

![!][!"]

!"#!

Note that in the finally expression P(W1W2…Wn) is not dependent on any topic. In other words, P(W1W2 … Wn) is a constant number for every topic once the sentence is given.

Since we are able to calculate P(W1,…, Wn|Tk), P(Wn) and P(W1W2 … Wn) respectively, we use formula (2) to combine them together.

P(Tk|S) =

!(!!,…,!")

! !!… !" !" ∗!(!")

We know P(Tk|S) indicates the probability of topic k given sentence S. We repeat these three steps for every topic from 0 to K-1. By definition of conditional probability

P(T0|S) + … + P(TK-1|S) = 1

And as we explained in step 3, P(W1W2 … Wn) is a constant number for every topic. So to be efficient, we do not calculate P(W1W2 … Wn) at all. Instead, we let

P’(Tk|S) = P(W1…Wn|Tk) \* P(Tk) and normalize the sum of probability P(Tk|S) to 1

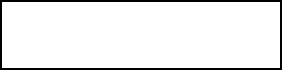
P(Tk|S) =

!’(!"|!)

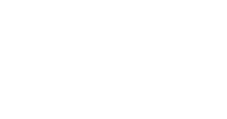
!’ !" ! !⋯! !’(!"|!)

## Incrementally Learning

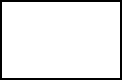
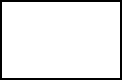
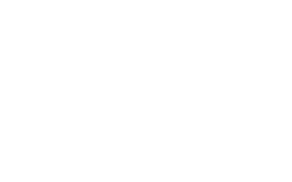
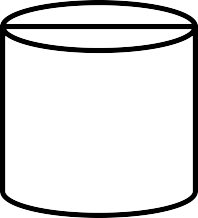
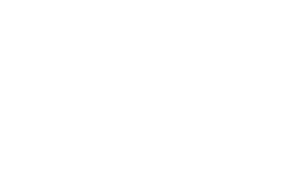
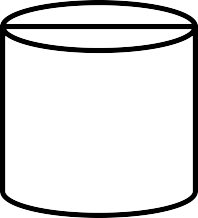
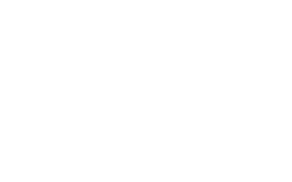
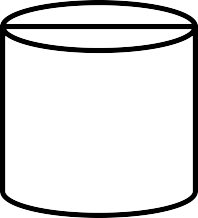
**Basic Framework**



Manually Label



Raw Sentence



Ambiguous

Dataset

Training

Dataset

SLTM

Train

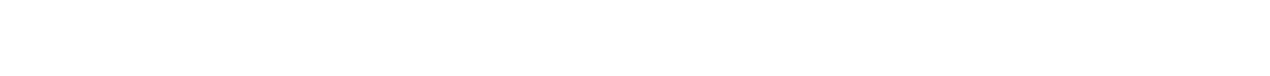
Testing

Dataset

Train

**Dataset Composition**

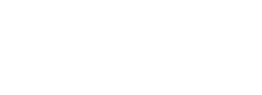
Our dataset consists of 3 parts namely Testing Dataset, Training Dataset, and Ambiguous Dataset.



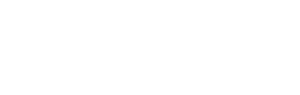
Training

Ambiguous

Testing



Labeled



Unlabeled

**Testing dataset** is a stand along dataset, which contains a fixed number of data points.

**Training dataset** is used as our training dataset to train the topic model. All the data points in it have been labeled manually.

**Ambiguous dataset** is used to save sentences, whose topic is ambiguous to SLTM and need to be determined by human. The sentences in this dataset will be shift to Training dataset, once human determined the topic of them.

## Incremental Leaning Algorithm

Now we have a list of probabilities P(T1|S) … P(Tk|S), where P(Tk|S) is the probability of sentence S describing topic k. The determine topic algorithm is shown below:

## Algorithm: Incremental Learning

**Input**: probability list P of sentence S

**Output**: topic of S or “ambiguous”

* + - 1. let max = 0
      2. for each P(Tk|S) in P
      3. if P(Tk|S) > P(Tmax|S), then let min = k
      4. if P(Tmax|S) > THRESHOLD
      5. then return Topic(max)
      6. else
      7. let a, b, c be the 3 most likely topics
      8. save sentence S and a, b, c to unlabeled sentence dataset
      9. return “ambiguous”

In this algorithm, we find the max number in list P. If this number greater than a certain threshold, then we return the corresponding Topic; else we save sentence S with the top 3 most likely topics into a local file and claim the topic of this sentence is “ambiguous” and out put string “ambiguous”.

For example, if we input sentence “*It's a small complaint though and a small price to pay for a 10 megapixel sensor.*” to our topic model, then the output probability list may be:

P(feature 1) = 1%

P(feature 2) = 2%

P(size\_weight) = 22%

P(price) = 36%

P(resolution) = 31% P(other features) = 8%

From this list, we can see that the sentence has 22% possibility to be classified into “size” feature, 36% into “price” feature, and 31% into “resolution” feature. Obviously, topic “price” has the higher possibility. However, if we set THRESHOLD equals to 0.5, then based on “Determine Topic” algorithm, the output is “ambiguous” rather than “price”, since 36% < THRESHOLD. In this case, “Determine Topic” module will append a JSON string to unlabeled sentences dataset (namely “ambiguous\_sentences”)

{“sentence”:” *It's a small complaint though and a small price to pay for a 10 megapixel sensor.*”,

“topic1”:”price”, “topic2”:”resolution” “topic3”:”size\_weight”

}

We also developed a simple human friendly interface to display these ambiguous sentences and let human determine which topic does this sentence belong to. Once human determined the topic of this sentence, then we save this sentence to *labeled sentence dataset* and delete it from *unlabeled sentence dataset*. In this way, with the help of human being, the system will gradually update the matrix towards a more accurate topic word distribution.

## Sentiment Analysis

Since we are able to extract product features from review sentences, in this section, we will introduce how to analyze the customer’s preference of the extracted feature. Here we utilize Stanford Sentiment Analysis toolkit rather than develop our own algorithm. The algorithm of this toolkit is introduced in the background section. In this section, we will give some demo samples of this toolkit.

Contrastive conjunction is a common way of expression in natural language. People express good feeling at the beginning of a sentence, then express some bad feeling at the end of this sentence connected by certain negation words such as but, however, etc. One example of this case would be:

“*There are slow and repetitive parts, but is has just enough spice to keep in interesting.*”

By applying the toolkit, we can see the structure of this sentence and sentiment of each word, phrase.

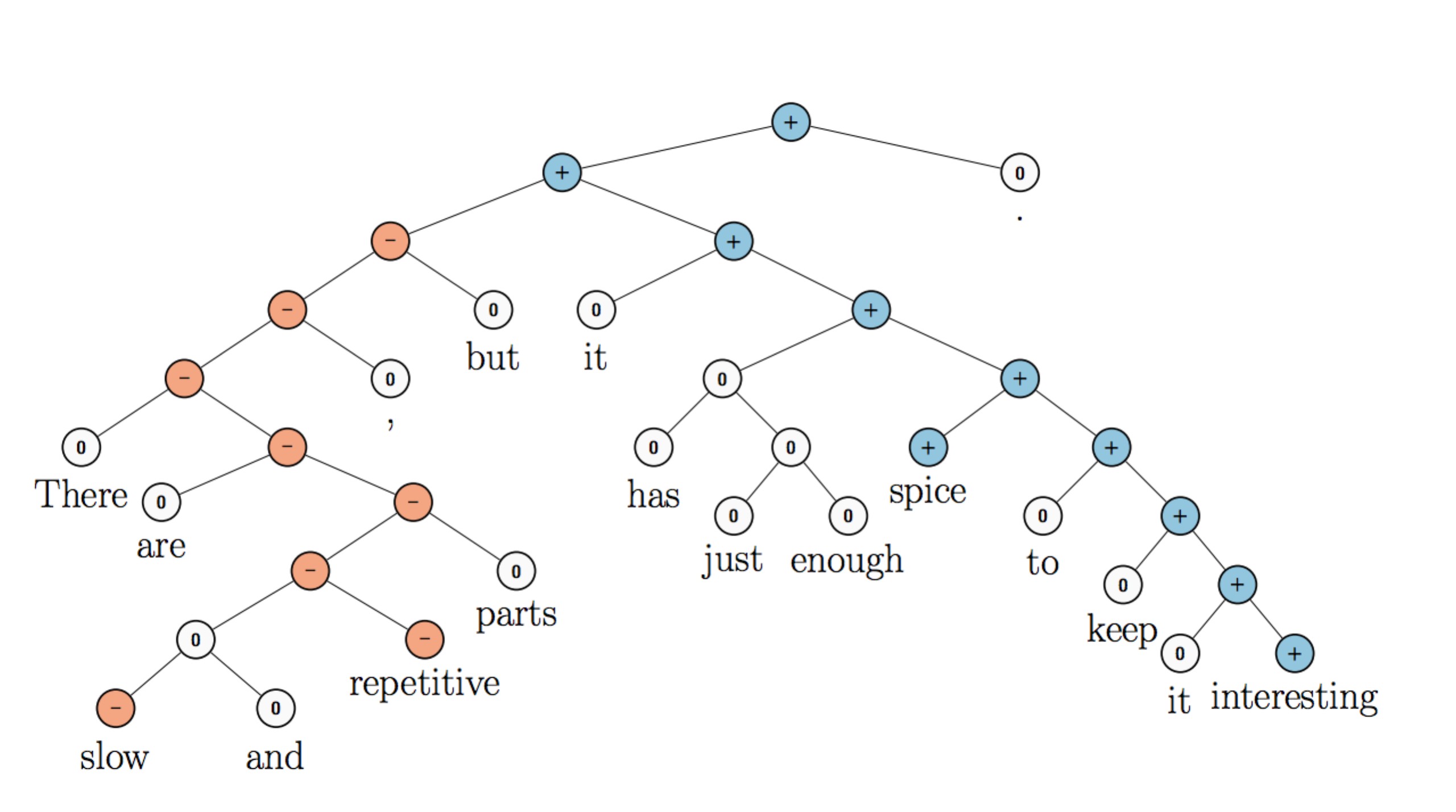


Figure 8: CoreNLP Sentiment Analysis toolkit example

From the parsing tree we can see that this toolkit first split the sentence into phases, then phrases into words. For the sentiment calculation, the toolkit uses bottom up algorithm. It calculates the sentiment for each word first, then for each phrase, and eventually for the whole sentence.

In our system, we assume one sentence only contains one topic. Therefore the sentiment score of whole sentence is the score for the topic.

## Review Summary

In previous section, we focus on sentence level to extract features and sentiment scores. In this section, we focus on review level to draw a conclusion of product pros and corns according to the feature either liked or disliked by review writers.

The calculation algorithm is pretty straightforward as shown below:

## Algorithm: Feature Preference Combination

**Input**: Product P with all of its reviews

**Output**: two vectors: LIKE representing how many customers like this product, and DISLIKE representing how many customers dislike this product.

1. Initial two vectors LIKE and DISLIKE, each with K-dimensions and set to 0.
2. Initial vector *PREFER* with K-dimensions and set to 0.
3. for each review R in product P
4. for each sentence S in review R
5. let F be the feature in S, Senti be the sentiment score of S
6. if Senti is positive, then increase *PREFER* [F] by 1
7. else decrease *PREFER* [F] by 1
8. for each number N in PREFER
9. if N greater than 0, then increase LIKE by 1
10. else increase DISLIKE by 1.
11. Output LIKE and DISLIKE

In this algorithm, for each review, we scan each sentence and keep track of either the review writer likes or dislike each feature. And for each product, we count how many review writers like each feature or dislike each feature. For example:

Product P has 3 reviews namely A, B, and C.

A’s author likes feature 1, 2, and 3; dislike feature 4, 5, and 6

B’s author likes feature 1, 3, and 4; dislike feature 6

C’s author likes feature 1, 2, and 6; dislike feature 3, 4, and 5 The LIKE and DISLIKE vector is counted as below:

|  |  |  |
| --- | --- | --- |
|  | LIKE | DISLIKE |
| Feature 1 | 1 | 0 |
| Feature 2 | 2 | 0 |
| Feature 3 | 2 | 1 |
| Feature 4 | 1 | 0 |
| Feature 5 | 0 | 2 |
| Feature 6 | 1 | 2 |

Finally we will use a bar graph to show customer this table for easier comparison.

# 5. Case Study

In this section, we will present a series of experiments designed to demonstrate the performance of our system.

## Product Feature Predefine

We take “Nikon Coolpix L300 Digital Camera (Black)” (ASIN: B00HQDBLDO) on amazon.com as our experiment product. The product’s basic information is shown below:

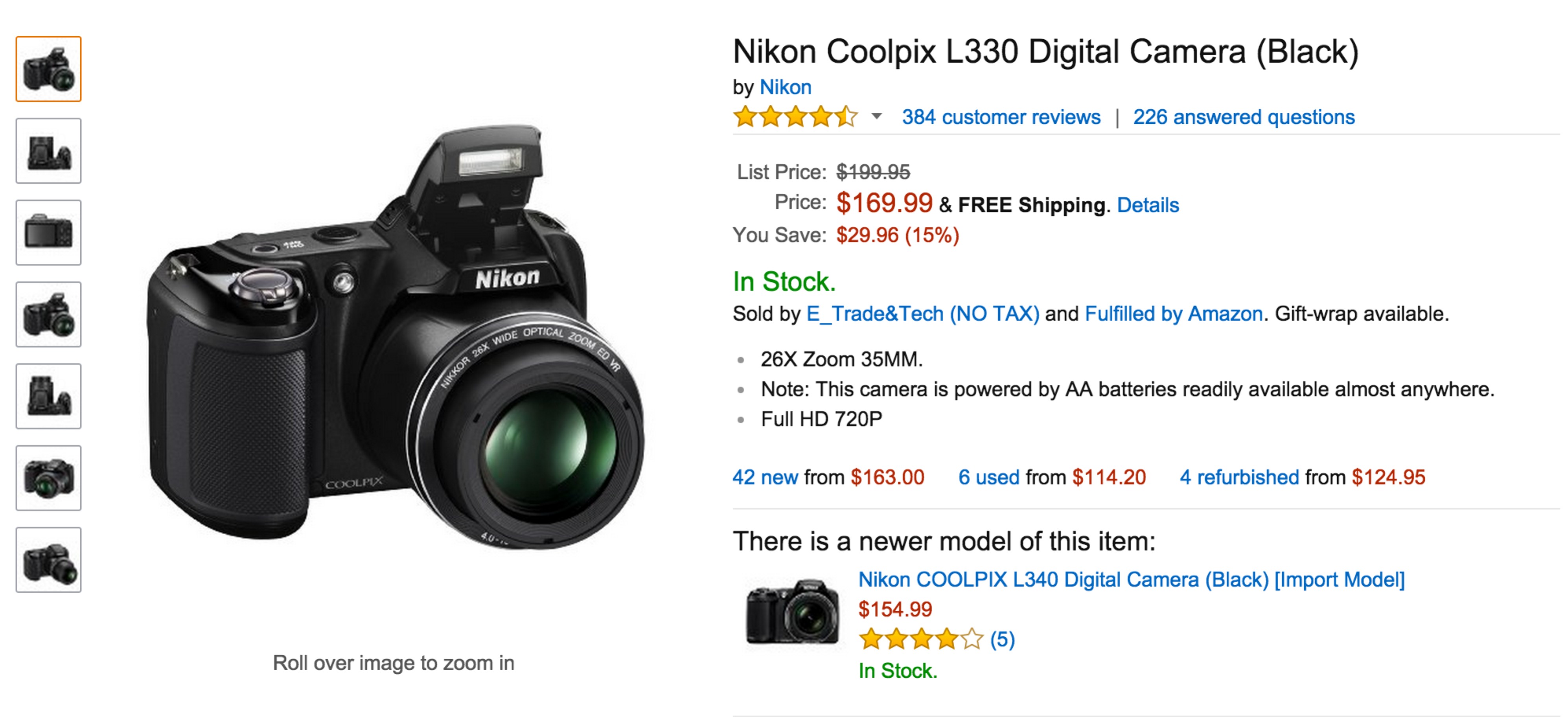


Figure 9: example product information on amazon

In the figure, we can see this product has 384 customer reviews with a 4.4 out of 5 stars rating as shown below:



Figure 10: example of amazon customer review system

Although, from this information, we know this product is a good one since the rating star is high, we still do not know which feature of this product is good. People care about different features such as: Does the battery last long enough or is it small and light enough for a long trip? Our system helps customers to answer these questions.

For digital camera, we define 10 customer most interested features, which are:

1. **Null**: refers to an empty feature. It indicates the sentence does not have a specific topic. Sentences like “*I like this camera*” or “*I bought this camera last week*” belongs to this features
2. **Lens:** is a transmissive optical device that affects the focus of a light beam through refraction.
3. **Size\_weight**: refers to the appearance of a camera. Such as a camera is big or small, heavy or light.
4. **Price**: such like a camera is cheap or expensive
5. **Resolution**: Camera resolution is measured in megapixels (meaning millions of pixels); both image file resolution and monitor resolution are measured in wither pixels per inch (pps) or pixel dimensions (such as 1024 by 768 pixels)
6. **Stabilization**: this feature prevents or compensates for unwanted camera movement, such as “camera shake”
7. **Accessory**: such as battery, SD card etc.
8. **Shutter**: is a device that allows light to pass for a determined period of time, exposing photographic file or a light-sensitive slectronic sensor to light in order to capture a permanent image of a scene.
9. **View\_screen**: also known as LCD or viewfinder. It is a device used to display images.
10. **Mode**: digital camera usually supports a number of modes for use in carious situations. Professional DSLR cameras modes focus more on manual modes, consumer point-and-shoot cameras focus on automatic modes. This feature also refers to the topic such as: if the camera is easy to setup or if the manual is clear.

Note that “null” feature means the sentence does not describe any features of the camera. For example, “I love this camera so much”, “My last camera was the Canon Powershot A495”, or “So happy I got this”. Such sentences either state a fact or express a general feeling of the camera.

## Review Data Collection and Preprocessing

We download and save all 384 reviews of this product by using our review clawer. Since we know the ASIN of this product is “*B00HQDBLDO*”, we can find this product’s main webpage at

[www.amazon.com/dp/B00HQDBLDO](http://www.amazon.com/dp/B00HQDBLDO)

Similar idea, we can find all this product’s reviews in:

[http://www.amazon.com/product-reviews/B00HQDBLDO/ref=cm\_cr\_pr\_btm\_link](http://www.amazon.com/product-reviews/B00HQDBLDO/ref%3Dcm_cr_pr_btm_link)

\_next\_2?ie=UTF8&showViewpoints=1&sortBy=recent&reviewerType=all\_reviews&fi

lterByStar=all\_stars&pageNumber=PAGE\_NUMBER

Where the PAGE\_NUMBER is an integer number indicating the review indexes, with each page contains at most 10 reviews, for example:

When PAGE\_NUMBER = 1, the url above display review indexed form 1 – 10; When PAGE\_NUMBER = 2, the url above display review indexed form 11 – 20; And so on, and so forth

The product *B00HQDBLDO* contains 384 reviews (by the time July 28, 2015), PAGE\_NUMBER ranges from 1 to 39. We save each review as an independent file in our local disk for further use.

Then, in review level data preprocessing, we use CoreNLP “ssplit” toolkit to split each review into sentences. In sentence level data preprocessing, we add annotators “POS” and “Lemma” to each word.

## Training Dataset Creation

To create initial topic word matrix, we need to manually label (classify) few sentences as training dataset. And to make this work easier, we design a simple Graphic User Interface (GUI) as shown below:

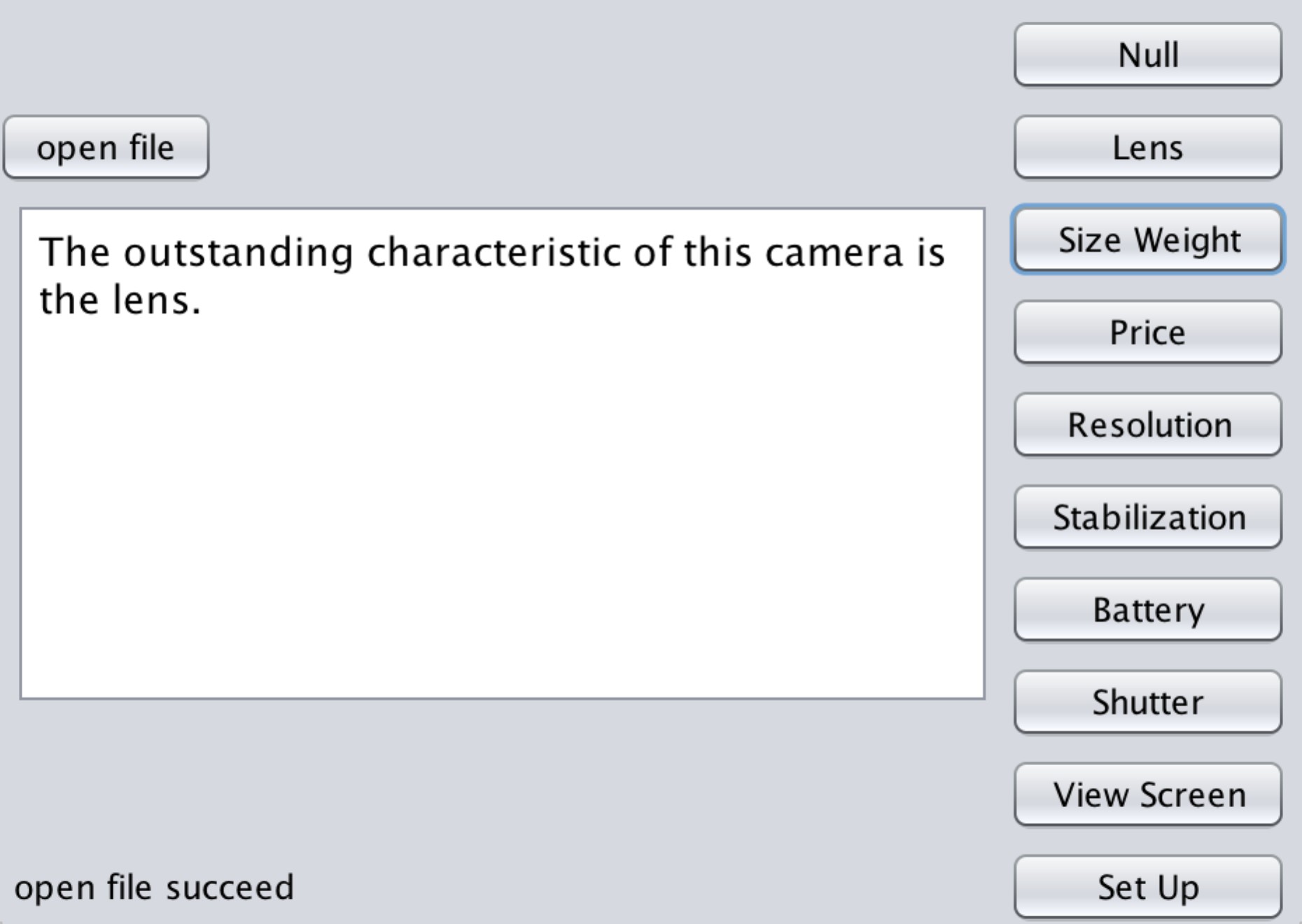


Figure 11: user interface for labeling data points

The left side of the GUI is where to display a sentence text to users and the right side is the feature buttons. The system reads one sentence from local file and displays it on left side and let users to choose a feature for this sentence. In the above example, users should choose the second button “Lens”, since the sentence is describing the lens. Once users click on one of the buttons, the system will save this sentence on training

dataset file with a “#feature” at end of it. In the above example, system will append “The outstanding characteristic of this camera is the lens. #lens” at the end of training dataset. Then the GUI displays next sentence.

Initially, we have labeled 441 sentences as our training dataset and the feature’s distribution is shown in table below:

|  |  |
| --- | --- |
| Features | Count |
| Null | 236 |
| Lens | 33 |
| Size\_weight | 21 |
| Price | 38 |
| Resolution | 9 |
| Stabilization | 5 |
| Accessories | 20 |
| Shutter | 16 |
| View Screen | 8 |
| Mode | 55 |

From the table we can see that, half of the sentences do not contain any camera features. Feature “Mode”, “Price”, and “Lens” are top 3 mentioned features in reviews. Note that we train the model gradually, such that we will add more data points into training dataset while using the model.

## Testing Dataset Creation

Likewise creating training dataset, we created testing date using the same way. We labeled 400 sentences as our testing dataset.

## Ambiguous Dataset Creation

We do not create this dataset initially. However while we using the topic model, some sentences cannot be classified by our model. These sentences will be put into this dataset and wait until human determine their topics.

## Experiment 1

In this experiment, we will evaluate topic model and sentiment analysis’s performance. We will propose an approach to calculate the system’s accuracy and introduce how to improve the performance incrementally.

## Data Set

We update the topic word matrix using the algorithm defined in previous section. Initially, Vocabulary list is empty. Then we scan the training dataset, if the word POS tag is “NN” or “JJ” and the word is not in the Vocabulary list, then add this word’s lemma to Vocabulary list. Finally, the Vocabulary size is 595. Therefore our topic words matrix has 10 rows and 595 columns.

## Test Accuracy

Since “null” topic is kind of special topic and we do not care about this features in real life, we test the performance in two ways:

* + 1. Consider “null” topic
    2. Do not consider “null” topic

We use such approach to test the model’s accuracy. Each time, we randomly pick 100 data points from it and let AC1 be the model’s accuracy on these 100 data points. Then let AC2 be the model’s accuracy on 100 data points from the left 300 data points. Same idea, we get AC3 and AC4. The model’s overall accuracy is the average number of AC1, AC2, AC3 and AC4.

AC = (AC1 + AC2 + AC3 + AC4) /4

When training dataset size = 441, the accuracy with “null” topic is 72%, without “null” is 78%.

## Incremental-Learning

Then we use the model to classify the reserved dataset. If the model returns “ambiguous”, then output the sentence and its top 3 most possible topics to our GUI to let human being determine the topic of this sentence. But this time, the GUI is a little bit different. On the right side, the most possible 3 topics will be highlighted in different color to help the user quickly find out the answer.



Figure 12: user interface for labeling ambiguous data points

Once human determined the topic, this sentence will be added to training dataset. In this way, the training dataset is gradually expanded. It is worth mentioning that the later added sentences are more helpful to improve the model’s accuracy since these sentences are all ambiguous.

We evaluate the performance when dataset size equals to 441, 700, 900, 1100, 1300 and 1500 respectively. The accuracy table is shown below:

|  |  |  |
| --- | --- | --- |
| Training Dataset Size | Accuracy with “null” | Accuracy without “null” |
| 441 | 72% | 76% |
| 700 | 78% | 84% |

|  |  |  |
| --- | --- | --- |
| 900 | 81% | 88% |
| 1100 | 83% | 92% |
| 1300 | 85% | 94% |
| 1500 | 86% | 95% |

In a line chart, we can see the accuracy improves smoothly:

1

0.95

0.9

0.85

with null feature

without null feature

0.8

0.75

0.7

441

700

900

1100

1300

1500

Figure 13: topic model accuracy line chart

Finally our topic model’s accuracy is about 95% (without null topic). Now we list some misclassified sentences

1. **Sentence:** “*I upgraded from my Canon S1IS to the Rebel and I am very pleased with everything the Rebel can do and how simple it is for me to grasp the*

*features and use them.*” **Real Topic:** Mode **Predicted Topic:** Null

In this sentence, the author is trying to say the camera’s feature is simple and easy to use. So it should refer to Mode feature. However, since this sentence’s structure is completed and the author implicitly describes the Mode feature, our topic model misclassified this one to Null feature.

1. **Sentence:** “*It was time to move to digital however and I just couldn't justify the cost of the 10mega-pixel EOS, so I opted to "step-down" to the Digital Rebel XTi.*”

**Real Feature:** Resolution

**Predict Feature:** Price

In this sentences, the author is trying to say Digital Rebel XTi’s resolution is not as good as EOS, but EOS is too expensive. Finally he bought XTi. Therefore the real feature described in this sentence is “Resolution”. The word “*10mega-pixel*” can indicate this feature. However, people type “*10mega-pixel*” in many different ways, such like “10 mega-pixel”, “10 mega pixels”. Our model cannot recognize such words are actually referring to meaning. So this sentence is misclassified.

1. **Sentence:** “After a little more than a month reading reviews I finally decided to go for this camera (Canon Rebel XTi).”

**Real Topic:** Null

**Predict Topic:** Mode

Since in training dataset, we classify sentences such as “reading manuals” into Mode feature. In this sentence, the word “reading” contributes lots of weight to feature Mode. Eventually, the sentence is misclassified into Mode feature.

## Evaluate Sentiment Analysis Performance

Since we are using the CoreNLP Sentiment Analysis Toolkit in this part and the authors have already evaluated the accuracy of this toolkit. Our testing dataset only contains 100 data points.

Based on our experiment, the accuracy of this toolkit is around 78% on a fine-grained classification. However, likewise we do not consider “null” topic, if we do not consider neural sentiment cases, the accuracy of this toolkit reaches 84%, which matches the number claimed by the authors.

## Overall Accuracy

The overall accuracy of our system is around 80.5%. Since we do topic extraction

and sentiment analysis sequentially, the overall accuracy is influenced by both of these two parts and can be calculated using the formula below:

Let AC1 be the accuracy of topic model

Let AC2 be the accuracy of sentiment analysis Let AC be the overall accuracy.

AC = AC1 – ((1 – AC2) \* AC1)

In our system, AC1 = 95%, AC2 = 84%. Therefore

AC = 0.95 \* 0.84 = 79.64%

Which is similar to our tested accuracy (80.5%).

## Experiment 2

In this experiment, we will compare 6 digital cameras on amazon and give out a result by analyzing their reviews using our SLTM system.

## Data Set

We choose 6 digital cameras, whose prices are all around $200, as our experiment product. These 6 products are in different brands but all belong to top-100 camera seller

on amazon. Moreover, they all have a 4.4 star rating, such that we cannot tell which one is the best through the classic star rating system. Therefore, they are the perfect candidates in this experiment. The tested cameras ASINs are:

* + 1. Nikon Coolpix (ASIN: B00HQDBLDO)
    2. Canon SX520 (ASIN: B00M0QVTOS)
    3. Canon Rebel XT (ASIN: B0007QKN22)
    4. Canon PowerShot SX400 (ASIN: B00M0QVG3W)
    5. Canon EOS Rebel (ASIN: B00V73JZY6)
    6. Canon PowerShot ELPH (ASIN: B00HLDFNKQ)

## Result

These 6 cameras have 415, 315, and 648 reviews respectively. Through analyzing the reviews, SLTM is able to draw such a conclusion:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Item 1 | | Item 2 | | Item 3 | | Item 4 | | Item 5 | | Item 6 | |
|  | L | D | L | D | L | D | L | D | L | D | L | D |
| Lens | 16 | 26 | 25 | 24 | 81 | 181 | 8 | 9 | 12 | 6 | 5 | 5 |
| Size Weight | 30 | 10 | 25 | 13 | 92 | 102 | 21 | 7 | 7 | 4 | 16 | 10 |
| Price | 62 | 24 | 98 | 28 | 221 | 135 | 29 | 18 | 47 | 9 | 31 | 15 |
| Resolution | 17 | 8 | 34 | 12 | 35 | 46 | 7 | 6 | 2 | 1 | 6 | 3 |
| Stabilization | 4 | 9 | 12 | 2 | 15 | 22 | 2 | 1 | 4 | 0 | 2 | 5 |
| Battery | 10 | 29 | 20 | 4 | 35 | 90 | 2 | 7 | 1 | 4 | 2 | 13 |
| Shutter | 20 | 26 | 21 | 28 | 92 | 115 | 10 | 11 | 12 | 8 | 8 | 9 |
| View Screen | 1 | 11 | 6 | 28 | 24 | 103 | 3 | 5 | 2 | 6 | 3 | 12 |
| Set Up | 37 | 29 | 28 | 29 | 127 | 173 | 6 | 7 | 14 | 11 | 10 | 22 |

Table 1: Experiment 2 Cameras’ Feature Score



Moreover we use a weighted score to compare them. That means “*very positive*” sentences get 1 score, “*positive*” sentences get 0.7 scores for “*Like*”, “*negative*” sentences get 0.7 scores and “*very negative*” sentences get 1 score for “*Dislike*”. Then we use a bar chart to demonstrate the weighted score in Figure 14:



|  |  |  |  |
| --- | --- | --- | --- |
|  | **B00HQDBLDO** |  | **B00M0QVTOS** |
| Set Up | 19 25 | 15 22 |
| View Screen | 8 1 | 19 5 |
| Shutter | 18 13 | 19 13 |
| Battery | 19 6 | 4 14 |
| Stabilization | 4 3 | 1 3 |
| Resolutin | 6 10 | 8 24 |
| Price | 23 | 19 68 |
| Size Weight | 6 18 | 11 17 |
| Lens | 17 10 | 16 16 |
|  | **B0007QKN22** |  | **B00M0QVG3W** |
| Set Up | 125 92 |  | 6 4 |
| View Screen | 77 20 |  | 4 2 |
| Shutter | 84 66 |  | 10 8 |
| Battery | 62 25 |  | 5 1 |
| Stabilization | 18 12 |  | 1 2 |
| Resolutin | 37 28 |  | 4 5 |
| Price | 97 | 160 | 13 24 |
| Size Weight | 75 63 |  | 5 15 |
| Lens | 131 59 |  | 8 7 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **B00V73JZY6** |  | **B00HLDFNKQ** |
| Set Up | 8 10 |  | 19 8 |
| View Screen | 5 1 |  | 11 2 |
| Shutter | 7 9 |  | 7 6 |
| Battery | 3 1 |  | 11 1 |
| Stabilization | 1 4 |  | 4 2 |
| Resolutin | 1 1 |  | 3 5 |
| Price | 7 | 38 | 12 28 |
| Size Weight | 5 6 |  | 7 15 |
| Lens | 5 9 |  | 3 4 |

Figure 14: Experiment 2 comparison result



The left-side bars indicate *dislike* scores of the corresponding feature, while the right-side ones are *like scores*. From these figures, we can see that Coolpix has a really bargain price, perfect size & weight, however, its view screen and shutter is not good; SX520’s price is also great, it has a good resolution and its size & weight is acceptable, however, its shutter, battery and view screen are all not satisfied; Rebel XT’s price and resolution gets more complains, but its lens, stabilization and shutter are all better than the other two. Here are some evidences sentences:

## Nikon Coolpix (ASIN: B00HQDBLDO)

**Example Feature: View Screen.** Most people do not like this feature

1. “*The one thing that drives me nuts is if you turn it on to see the pictures it will not display them until you take off the lens cap so the camera can extend the lens*”
2. “*Only one thing that would make it better would be a view finder.*”
3. “*When I press the button to wake it up it was just a black screen with the message of*

*please turn camera off and take lens cover off.*”

## Canon SX520 (ASIN: B00M0QVTOS)

**Example Feature**: **Shutter**. Half people like it, while half people do not like it.

1. “*On the other side this camera is wonderful in a low light situations and zoom is fantastic.*”
2. “*Low light photography is not satisfactory.*”
3. “*I love the Wind detection shutter release.*”
4. “Let me tell you, that shutter lag is a really big deal to me.”

## Canon Rebel XT (ASIN: B0007QKN22)

**Example Feature: Lens.** Most people do not like this feature.

1. “*Order this body without the Canon lens.*”
2. “*Ship it with an EF lens, not an EF-S*”
3. “*The kit lens isn’t a good indoors or low-light lens.*”
4. “*It is a very inexpensive lens(about $60-$70) and takes amazing shots.*”

This experiment shows that, with the help of SLTM, customer can make a better decision based on certain specific demands without reading reviews.

# 6. Conclusion and Future Work

In this thesis we introduce a Sentence Level Topic Model System to evaluating the comparing online products based on their reviews. By counting the word frequency, we developed a topic model based on Bayes’ rule. In our approach, we use the products’ reviews as piece of evidence to identify whether a list of features of this product is a favorable one or not. Product data from e-commerce websites such as Amazon is quantified and evaluated using our approach. Our case study show that by using our system, customers can greatly save time reading reviews and easily compare several similar product based on their features.

The difference between our topic model and pLSA & LDA is that in pLSA & LDA, a document contains multi topics and a topic contains multi words, pLSA has two matrixes and is a two-layer Bayes classifier (LDA is a three-layer Bayes classifier because of the prior). However, in our model, we split a document into sentences and we assume one sentence contains only one topic. Therefore, our model has only one matrix (topic word matrix) and is a one layer classifier. Moreover, since we study on a specific domain, we use supervised learning approach rather than a unsupervised learning approach such as EM algorithm.

In future research, we plan to let the system dynamically identify new features in the reviews rather than predefine some features like now. Moreover, we would like use parameter estimation technics to create the model matrix rather than create it from manually labeled data points. Finally, we may train the Sentiment Analysis toolkit using different domain datasets.

# Reference

[1] Nasukawa, Tetsuya and Jeonghee Yi. Sentiment analysis: Capturing favorability using natural lan- guage processing. In Proceedings of the K-CAP-03, 2nd International Conference on Knowledge Capture. 2003.

[2] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using ma- chine learning techniques. In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2002). 2002.

[3] Turney, Peter D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classifica- tion of reviews. In Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2002). 2002.

[4] Wiebe, Janyce, Rebecca F. Bruce, and Thomas P. O’Hara. Development and use of a gold-standard data set for subjectivity classifications. In Proceedings of the Association for Computational Linguis- tics (ACL-1999). 1999.

[5] Hu, Minqing and Bing Liu. Mining and summarizing customer reviews. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004). 2004. doi:10.1145/1014052.1014073

[6] Jindal, Nitin and Bing Liu. Mining comparative sentences and relations. In Proceedings of National Conf. on Artificial Intelligence (AAAI-2006). 2006b.

[7] E.Boiy,P.Hens,K.Deschacht,andM.-F.Moens,“Automaticsentiment analysis in on-line text,” in Proceedings of the 11th International Conference on Electronic Publishing, pp. 349–360, 2007.

[8] P. D. Turney, “Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews,” in Proceedings of the 40th annual meeting on association for computational linguistics, pp. 417–424, Association for Computational Linguistics, 2002.

[9] J. Kamps, M. Marx, R. J. Mokken, and M. De Rijke, “Using wordnet to measure semantic orientations of adjectives,” 2004.

[10] Princeton University "About WordNet." WordNet. Princeton University. 2010.

[<ht](http://wordnet.princeton.edu/)t[p://wordnet.princeton.edu>](http://wordnet.princeton.edu/)

[11] Andrea Esuli, Fabrizio Sebastiani. A Publicly Available Lexical Resource for Opinion Mining

[12] G. Vinodhini and R. Chandrasekaran, “Sentiment analysis and opinion mining: A survey,” International Journal, vol. 2, no. 6, 2012.

[13] P. Domingos and M. Pazzani, “On the optimality of the simple bayesian classifier under zero-one loss,” Machine Learning, vol. 29, no. 2-3, pp. 103–130, 1997.

[14] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs Up? Sentiment Classification Using Machine Learning Techniques,” Proc. ACL-02 Conf. Empirical Methods in Natural Language Processing (EMNLP), 2002.

[15] B. Pang and L. Lee, “A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts,” Proc. 42nd Ann. Meeting on Assoc. for Computational Linguistics (ACL), pp. 271-278, 2004.

[16] C. Whitelaw, N. Garg, and S. Argamon, “Using Appraisal Groups for Sentiment Analysis,” Proc. 14th ACM Int’l Conf. Information and Knowledge Management (CIKM), pp. 625-631, 2005.

[17] P.D. Turney, “Thumbs Up or Thumbs Down?: Semantic Orienta- tion Applied to Unsupervised Classification of Reviews,” Proc. 40th Ann. Meeting on Assoc. for Computational Linguistics (ACL), pp. 417-424, 2001.

[18] B. Pang and L. Lee, “Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales,” Proc. 43rd Ann. Meeting on Assoc. for Computational Linguistics (ACL), pp. 115-124, 2005.

[19] Z. Zhang and B. Varadarajan, “Utility Scoring of Product Reviews,” Proc. 15th ACM Int’l Conf. Information and Knowledge Management (CIKM), pp. 51-57, 2006.

[20] Papadimitriou, Christos; Raghavan, Prabhakar; Tamaki, Hisao; Vempala, Santosh (1998). "Latent Semantic Indexing: A probabilistic

analysis" (POSTSCRIPT). Proceedings of ACM PODS.

[21] Hofmann, Thomas (1999). "Probabilistic Latent Semantic

Indexing" (PDF). Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval.

[22] Blei, David M.; Ng, Andrew Y.; Jordan, Michael I; Lafferty, John (January

2003). "Latent Dirichlet allocation". Journal of Machine Learning Research 3: 993– 1022.doi:10.1162/jmlr.2003.3.4-5.993.

[23] Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.

[24] Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.

[25] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer.

2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Proceedings of HLT-NAACL 2003, pp. 252-2