Conversion Rate Prediction in Search Engine Marketing

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Razieh Nabi-Abdolyousefi

in partial fulfillment for the
degree of Master of Science

in
Electrical and Computer Engineering

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Computer Science and Engineering.

APPROVED BY:

Asst. Prof. Dr. Ahmet Bulut .....
(Thesis Advisor)

Asst. Prof. Dr. Ali Cakmak .....

Prof. Dr. Aslihan Nasir .....

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL: 24 Dec 2014

SEAL/SIGNATURE:

Declaration of Authorship

I, Razieh Nabi-Abdolyousefi, declare that this thesis titled, 'Conversion Rate Prediction in Search Engine Marketing' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Razieh Nabi

Date: 27/12/2014

Conversion Rate Prediction in Search Engine Marketing

Razieh Nabi-Abdolyousefi

Abstract

Search engines hold online auctions among search advertisers who are bidding for the advertisement slots in the search engine results pages. Search engines employ a pay-per-click model in which advertisers are charged whenever their ads are clicked by users. If a user clicks on an ad and then takes a particular action, which the corresponding advertiser has defined as valuable to her business, such as an online purchase, or signing up for a newsletter, or a phone call, then the user's action is counted as a conversion. A naive estimate of the conversion rate (CR) of an ad is the average number of conversions per click. The average number of clicks and the average position of the ad also affect its conversion rate. However, all such ad statistics are heuristics at best. The challenge here is that there is no performance statistics accrued for the newly created ads. In order to get any kind of performance data, new ads have to be advertised first and precious marketing dollars have to be spent. If CR estimates are precise, then advertisers can manage their campaigns more effectively and can have a better return on their investments. Alternatively, one can use the available data for the existing ads and engineer a set of features that best characterize conversions for an advertisement campaign in general. We took the second approach and used probabilistic inference for extracting text features. Using these text features, we built a prediction model to estimate the true CRs of unknown ads. Our experiment results demonstrated that such text features improved the accuracy of our predictions. Furthermore, hybrid models that combine text and numeric features achieved a superior predictive power compared to using only text features or only numeric features.

Keywords: Conversion (Rate), Sponsored Search Marketing, Online Advertising, Search Engine Marketing, Textual Ads, CPC Online Model

Arama Motoru Pazarlama Dönüşüm Oranı Tahmini

Razieh Nabi-Abdolyousefi

Öz

Arama motorları, sonuç sayfalarındaki reklam alanları için teklif veren arama reklamcıları arasında çevrimiçi açık artırmalar yapmaktadırlar. Bu arama motorları reklam verenlerin reklamlarına tıklandığında para ödedikleri bir tıklama başına ödeme modeli kullanırlar. Eğer bir kullanıcı bir reklama tıklar ve satın alma, bir haber bültenine kayıt olma veya telefonla arama gibi reklamcı tarafından önceden değerli olarak belirlenmiş bir eylemi gerçekleştirse kullanıcının eylemi bir dönüştürme olarak sayılır. Bir reklamın dönüştürme oranının sığ bir tahmini tıklama başına ortalama dönüştürme sayısıdır. Tıklamaların ortalama sayısı ve reklamın pozisyon bağlamındaki ortalama sırası da dönüştürme oranını ayrıca etkiler. Öte yandan, bu tür istatistikler en iyi ihtimalle sezgiseldir. Buradaki problem yeni oluşturulan reklamlar için bir performans istatistiği mevcut olmamasıdır. Halihazırda performansa ilişkin bir veri alınabilmesi için reklamlar öncelikle yayınlanmalı ve milyonlarca dolar harcanmalıdır. Öte yandan, eğer dönüştürme oranı tahminleri kesinse reklam verenler reklam kampanyalarınıdaha iyi yönetebilir ve yatırımlarının karşılığını daha iyi alabilir. Varolan metotlara alternatif olarak önceki reklamlar için varolan veriler kullanılabilir ve herhangi bir reklam kampanyası için dönüştürmeleri en iyi karakterize eden öznitelikler belirlenebilir. Bu çalışmada ikinci yaklaşım ele alınmakta ve olasılıksal çıkarım metin bazlı özniteliklerin çıkarılması için kullanılmaktadır. Bu metin bazlı öznitelikler kullanılarak doğru dönüştürme oranının tahmin edilebilmesi için bir tahmin modeli geliştirilmiştir. Deneysel sonuçlar bu metin bazlı özniteliklerin tahmin doğruluğunu artırdığını göstermiştir. Dahası metinsel ve sayısal öznitelikleri birleştiren hibrit modeller yalnızca metin bazlı veya sayısal bazlı modellere nazaran daha iyi bir tahmin performansı göstermiştir.

Anahtar Sözcükler: dönüştürme (oran), sponsorlu arama pazarlama, online reklam, arama motoru pazarlaması, metinsel reklamları, CPC online modeli

I would like to dedicate this body of work to my family members, who have supported me throughout my life.

Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor Prof. Ahmet Bulut. Words cannot express my gratitude and respect toward him. His patience, knowledge, philosophical point of view, and in one word his personality have influenced me in different ways. Working with a supervisor who considers both personal and professional growth of his students was a big fortune for me. He taught me having passions is one side of the story, but how we can have meaningful approaches toward them is another. He taught me that enough is never enough and there is no end to perfecting the quality of work we deliver. He taught me how to be a good teacher, a good researcher, and a better person. His lifetime lessons will stay with me forever.

I would also like to deeply thank my jury members, Prof. Aslihan Nasir and Prof. Ali Cakmak, for their time, insightful comments, and their support. I would like to greatly acknowledge and thank the Electrical and Computer Engineering department and, in particular, Secil Ocal for creating a friendly environment to conduct research. I would also like to thank my friends for all the great times we had together during my stay in this university, all laughs and cries, and ups and downs.

The last and not the least is my family. I would like to thank my family for all their love and encouragements from thousands of miles away. Their pure love and their memories have enabled me to go on. I have been extremely fortunate to have a thoughtful sister or I should say an angle like Marzieh for her enormous support in different stages of my study and my life, an angel like Sareh, who can give wise advice whenever I need it the most, a responsible brother like Aboozar, who I can trust the most to finish whatever I ask him to do, a hard working angel like Soheila, who can play our supportive roles at home when we are away, and my little angel Negin whose kindness is endless. Whatever I do in my life is an appreciation to our parents who taught us to love, be wise, responsible, supportive, kind, and have good morals.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Online Advertising Overview

Most advertisers use Internet to deliver their marketing messages to customers around the world. This type of advertising is called Online Advertising or Internet Advertising. Social media marketing, search advertising, contextual advertising, display advertising, and video advertising are some examples of digital advertising on Internet. The online advertising framework consists of a publisher who places the ads into its online content, an advertiser who provides the publisher with the advertisements to be displayed on the publisher's content, agencies who place and control the ad copy, and an ad server which delivers the ad and tracks the ad's statistics.

Search Engine Marketing (SEM) is one of the most common forms of online advertising. Search engines provide ad slots in search results pages (SERPs) for advertisers to display their ads next to organic search results for a user query. In other words, search engines sell their ad slots by running an auction among advertisers who are bidding and competing for these slots. There are three main advertising models used by search engines.

- Pay-per-Click (PPC): advertisers pay when their ads are clicked,
- Cost-per-Impression (CPM): advertisers pay when their ads are shown in SERPs, and
- Cost-per-Action (CPA): advertisers pay when users carry out a designated action, such as purchase of an item or signup to an online service.

PPC is widely used among search engines. In some contexts, Cost-Per-Click (CPC) is also used to refer to the PPC model. There are two main concepts in a PPC model: (1) advertiser is charged each time someone clicks on her ad, and (2) advertiser decides the maximum amount she is willing to pay per each received click on her ad.

Currently, there are three major PPC networks: Google (through: Adwords), Microsoft (through: Adcenter), and Facebook (through: Ads). In order to define general

terms used in online advertising, we mostly use Google Ad Network as our reference[1] as well as [1], [2], [3], and [4].

Google Ad Network mainly consists of a search network, a display network, and a video network (Youtube). In this study, we deal only with the search networks. Google provides a platform for advertisers called **Adwords**; to launch an advertisement in Google search engine, advertiser must have an Adwords account.

After creating an account, the next step is to create search campaigns. A **Campaign** is a set of ad groups that share a budget, that target a set of locations, and that have a time schedule for ad delivery, which determines what days and times are suitable for airing those ads. Campaigns are used to organize products or services that an advertiser offers. Advertiser manages a collection of ads, keywords, and bids together in one **Adgroup**. For a given product, advertiser picks a set of keywords that are most likely to appear in user queries along with an auction bid per each keyword. In a PPC model, a bid is an indicator of how much the advertiser is willing to pay for each user click.

When the campaign is launched, the search engine considers the ads in the campaign for future user queries that match the advertiser's specified keywords. In other words, when a user searches for a query, the search engine looks for eligible ads that their keywords match what the user is looking for. If the search engine determines that the ad matches the user query, then the corresponding will be considered to be shown on the SERP for the query.

When an ad is shown to a user, it is up to the user to decide whether to click on it. The number of clicks received by an ad normalized by the times the ad is shown is called **Clickthrough-Rate (CTR)**. CTR for an ad represents users likelihood to click on the ad. As an example, if an ad receives $1,000$ impressions, where **Impression** refers to how many times an ad has appeared on SERPs, and 20 clicks, its CTR value is 2%.

$$CTR_{ad} = \frac{\#\text{ received clicks}}{\#\text{ ad views (impression)}}.$$

In PPC, the search engine gets paid on user click. On the other hand, advertisers revenue depend on user's action post click. When someone clicks on an ad and then takes an action that is defined as valuable to advertiser's business, such as online purchase, or signing up in the website, or a call to the business from a mobile phone, a **conversion** has occurred. **Conversion Rate (CR)** is the average number of conversions per each click, Figure 1.1.

Managing bids for campaign keywords is challenging. Search auctions are competitive and dynamic; new advertisers can come in to the market and can increase or decrease their bids for changing the nature of competition constantly. In order to ease the bid management burden, Google uses conversion optimizer for adjusting bids more

[1]Google AdWords Help Center,https://support.google.com/adwords/

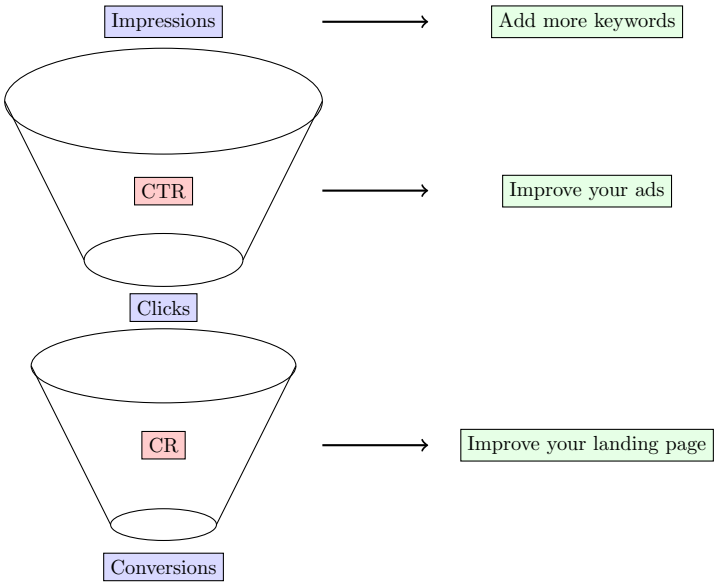


FIGURE 1.1: Impression refers to displaying an ad on a SERP. The number of ad clicks divided by impression is CTR. A subset of the total number of clicks will result in conversion. Total number of conversions divided by clicks is CR. For increasing the number of impressions advertisers should use more precise and popular keywords. In order to increase CTR, advertisers should improve ad quality. In order to improve CR, advertisers should improve the landing page (where the user clicks are directed to) and make it relevant to user needs expressed as user queries.

efficiently. By using historical information about the campaign, conversion optimizer computes the optimal CPC bid per ad each time it is eligible to appear. Advertiser still pays per click but she no longer needs to adjust the bids manually.

Among the eligible ads to impress for a given user query, each search engine can use a different method for assigning the set of eligible ads to the limited number of ad slots. In **first price auctions**, ad position is exclusively determined by advertiser's bid. Ads with higher bids take higher positions. In **second price auctions**, advertisers pay just enough to beat the competition. In other words, advertisers do not pay their full bid amount, but they pay an amount that is equal to the next highest bid. Table 1.1 compares the position and the actual cost among four different advertisers in these two types of auctions.

TABLE 1.1: First price and second price ad auctions. The illustration shows how ad positions and actual costs are determined in each ad auction.

Advertiser	Real Bid*	First Price Auction		Second Price Auction	
		Actual Cost	Ad Position	Actual Cost	Ad Position
A	\$4	\$4	1	\$3	1
B	\$3	\$3	2	\$2	2
C	\$2	\$2	3	\$1	3
D	\$1	\$1	4	–	–

*Indicates amount advertisers are willing to pay when a user clicks on their ads and visits their landing pages.

In the aforementioned auctions, ad positions are determined solely by bids: the highest bid gets the highest ad position. The main problem with this approach is that it does not take the quality of the ads into account. The higher the bid is the more likely it would get the better position regardless of whether the ad meets a quality standard. To offer a quality of service to both advertisers and users, search engines should allocate higher quality ads to higher ad positions. Google considers external and internal factors that impact the quality of an ad and places a given ad in the most suitable position within a SERP.

To track the quality of ads, Google computes a **Quality Score** for each ad. Main factors that affect the quality of an ad are as follows:

- Expected CTR: An important factor in ranking the ads is CTR, which represents users likelihood to click through. Google constantly explores ads by allocating them to ad slots.
- Landing Page Quality: Relevance of the ad to the landing page increases its quality score. If the ad matches the landing page text, it is considered relevant. External factors such as whether the page loads fast, and whether the page is navigable also affect the ad quality score.
- Ad Relevance (Ad Text): By analyzing the text and language in the ad, Google determines how relevant the ad is to the search query.
- Ad Formats: Ad formats should contain information about advertiser's business like phone number, highlighted keywords relevant to queries, or website domains in ad's headline. They convey more information to users and make it more likely that users would click on the ad. Consequently, the format is an important factor in scoring the quality of an ad.

According to how the pairwise entities of CTR, landing page, ad text, and ad formats interact, Google assigns a quality score to the ad as shown in Figure 1.2.

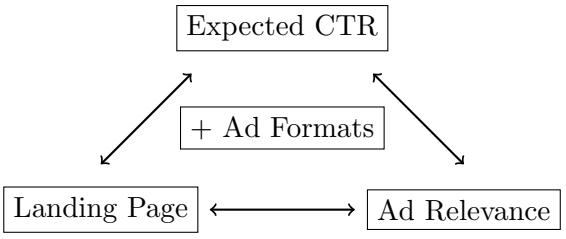


FIGURE 1.2: Main factors affecting ad's quality.

Google ranks ads according to their **Ad Rank** scores, which is computed using quality score and bid. Ad Rank controls both CPC value and ad position as illustrated in Table 1.1. Suppose that ad ranks are computed and collected in the Ad Rank column in Table 1.2. From Table 1.2, we can observe that although the ad A has the highest

bid, it is not going to appear in a SERP. Advertisers pay the minimum amount that is necessary to maintain their positions and it is always less than the amount they are willing to pay, i.e., their real bid. The actual cost the advertiser will pay for a click is the minimum amount needed to beat the Ad Rank of the ad position below. The numbers in Table 1.2 are taken from the example given by Hal Varian, Google's chief economist[2], and are made up values. The intention is to show how ad quality impacts cost and position in Google's version of a second price auction.

TABLE 1.2: Ranking the ads based on Ad Rank. The key components of Ad Rank are ad's bid and quality. This table represents how the auction system works and how ad rank determines ad's position and cost. In this scheme, users see the most relevant ads and advertisers get the most value.

Ad	Real Bid	Ad Rank	Position	Actual Cost
C	\$2	20	1	\$1.73
B	\$3	15	2	\$2.68
D	\$1	8	3	\$0.69
A	\$4	5	4	–

Return on Investment (ROI) is one of the most important performance metrics in search marketing. There are several ways to determine ROI and the most simple and frequently used method is to divide the net profit by the net cost.

$$\begin{aligned} Profit &= Revenue - Cost, \\ ROI &= \frac{Profit}{Cost}. \end{aligned}$$

1.2 Goal and Motivations

Online advertising is a win-win game if and only if players are familiar with the game's rules. A search engine must have a good estimation of an ad's expected CTR value in order to display it in the correct position. The search engine's revenue depends solely on the aggregated clicks since advertisers are charged per user click in PPC model. Consequently, predicting CTR is highly crucial to a search engine.

Advertisers, on the other hand, must think deeper and have a good perspective of whether the user's click would result in a conversion or not. Failing at conversion prediction means a business failure since advertiser is charged for each received click but the click may or may not lead to a conversion. From an advertiser's perspective that target conversions, predicting conversion likelihood before clicks is extremely crucial.

Conversion is a rare event. For a newly created ad, its CR potential is not known in advance till enough users click, some of which may convert eventually. Advertisers have to pay for these clicks whether those clicks convert or not. The challenge is that

[2]Insights on the AdWords Auctions, http://www.youtube.com/watch?v=PjOHTFRaBWA

advertisers do not know a-priori whether a keyword or an ad is likely to convert till it does convert. Estimating CR in advance is essential from an advertiser point of view. By having a promising approximation on CR, advertisers can reach a desirable ROI by correcting the bid and maximum CPC on each keyword. The procedure is described as follows:

What is a visitor's worth?
Each visitor's worth is equal to average profit$\times$CR.
Example - Assume an average sale revenue of \$150 and an average cost of \$100, resulting in a profit of \$50 per sale. Suppose that out of 100 visitors to the website, only 2 of them purchased the good resulting in a CR of 2%. Consequently, each visitor is worth $\$50 \times 0.02 = \$1$. The pay-per-click bid must be less than \$1 in order to gain profit.

Calculating Max. CPC
Given ROI and each visitor's worth of $\$a$, maximum bid should be equal to $a \times (1 - \frac{ROI}{100})$.
Example - If the target ROI is 30% and each visitor's worth is \$1, the maximum bid should be $\$1 \times (1 - 0.3) = \$0.7$.

A good estimation of CR helps the advertiser to manage her campaign more efficiently and increase her ROI. In this study, we are interested in building a model for predicting CR in a search campaign in general, or predicting the CR per keyword or the CR per ad.

1.3 Outline

The goal of current chapter is to familiarize the reader with the terminology used in online advertising and provide a general overview, and our motivation behind this study. The rest of the thesis is structured as follows: a review of related works on online advertising is discussed in Chapter 2. Essential theoretical basis is explained in Chapter 3. The formal definition of the problem and the methodology used for tackling the problem of conversion rate estimation are given in Chapter 4. Experiment results are provided in Chapter 5 along with a thorough analysis of the results. A summary of conclusions along with potential expansions of the work is provided in Chapter 6.

Chapter 2

Literature Review

In this chapter, we review some of the works done in online advertising.

When a user submits a search query, the search engine tries to determine the most relevant search keywords (which reside in adgroups) in search campaigns of many different advertisers to the submitted query and display those ads (in the respective adgroups) that are best responses to the user's query. The "best" ads in PPC model are those that maximize the expected revenue of the search engine.

$$E_{ad}[revenue] \propto CTR_{ad} \times CPC_{ad}.$$

CPC_{ad} is the ad's bid (in a first price auction) or the bid of the next-highest bidder (in a second-price auction), which could be normalized by ad performance (see Table 1.2).

Revenue of the search engine is directly proportional to the advertiser's bid. In PPC model, those ads that receive more clicks and therefore have higher CTRs are more profitable for search engines. The bids are known to the search engine in advance because advertisers determine the bids for the keywords in their search campaigns. Therefore, the problem of ranking the eligible ads in order to assign them to the available ad slots from top to bottom reduces to finding the expected CTR of those ads. There is a rich body of literature on CTR prediction.

In [5], authors use a regression-based approach to estimate CTR for new ads. In their approach, they do not take the search queries into consideration and train their model independent of the queries issued.

In [6], authors state that true CTR is not what search engines should look for. Instead, they need to determine the relative ranks of the top relevant ads for specific query keywords. Building a ranking-based model in this context seems to be more intuitive than a regression model. In this research, unlike [5], authors do not try to predict CTR but to rank the ads in descending order of CTR. They propose a query-dependent approach in order to estimate the ranks of the eligible ads. Taking user queries into account largely affects whether the user would actually click on an ad. They first

try to distinguish good ads from the bad ones and then select ads that have higher CTR. They evaluated human expert tips for improving ad CTR such as displaying keywords in the ad headline, a direct message to users, capitalizing every word in the ad description, offering a free product, using action words, e.g., "Buy" or "Get", and creating a sense of urgency. Their experiment results on historical search logs suggest that the ads that follow the expert tips have higher CTR. Using these tips in a ranking model, the ads with higher ranking CTRs can easily be selected for display to users. The motivation of this work is influenced by the work of [7], who proposed a ranking approach to collaborative filtering.

In [7], authors discuss content-based filtering and collaborative filtering used by recommender systems. In content-based filtering, items and users are represented using a set of features derived from content such as product descriptions or user profiles. Content-based approach matches the products of interest to a user. Conversely, the collaborative filtering (CF) approach uses the user ratings per item rather than the content information. In the latter, a recommendation is given to a user based on the preference of other users. Consequently, the problem in CF is reduced to finding the top-N items to recommend. In CF, one common approach is to first predict the rates and then rank them accordingly. In this paper however, the authors proposed to rank the ratings instead of predicting them. The methods they used were based on a greedy heuristic and a random walk model.

From a search engine perspective, the task is to identify the top-most relevant ads to a user query; therefore, one does not need to care about the exact CTR values. Hence, a ranking-based approach is more promising for carrying out the task. However, this ranking approach is not easily applicable to the problem of conversion prediction. Advertiser needs to know how much profit a chosen keyword will bring to her business. Having a good approximation of CR per keyword can immensely benefit the advertiser (as in only advertising for the keywords with higher CRs). A ranking approach is useful when the advertiser wants to select a keyword for campaigning among a large set of candidate keywords. In short, we are interested in finding a decent CR estimation per keyword rather than ranking keywords among each other.

For search engine, placing relevant ads in higher positions boosts the revenue by increasing the number of received clicks. Ads in lower positions are less likely to be viewed by users and tend to lower CTRs. CTR of an ad typically decreases with display position due to reduced visual attention[1]. Consequently, placing the best performing ads in top positions increases both the revenue and user satisfaction. On the other hand, the impact of ad position on CR is not the same as on CTR. The impact of ad position on CTR and CR is discussed in [8]. According to their findings, CTR decreases

[1]Eye Tracking Study, http://eyetrackingupdate.com/,

as expected with position. However, contrary to conventional wisdom, the top positions are associated with lower revenue relative to lower (and less expensive) positions. The results in this paper suggest that the top positions have lower CRs compared to middle positions.

To manage campaigns more efficiently, advertisers can vary campaign settings by using different variants of keywords, bid choices, and text options. Then, they can monitor user's purchase behaviour in different campaign settings and finally pick the best setting in performance. If advertisers can determine purchase intent in user queries, they can handcraft more relevant search keywords to users. Discovering the latent thematic structure among user queries uncovers the possible information needs of users and help us to target users with purchase intent in an easy and more organized way. The authors in [9] used a dynamic LDA-based topic model for learning the hidden themes in search terms. Once themes are identified, advertisers can manage themes instead of raw keywords. Our study uses the idea of discovering latent thematic structure as in [9] to find out which keywords are more likely to convert and further use this information to build a prediction model.

Chapter 3

Probabilistic Models

3.1 Topic Models and LDA

By the progressive rate of information in different domains (in forms of text archives, image archives, and other forms of data), it is getting impossible to follow every proposed and published information. We need tools to help us to organize & summarize these vast amounts of information, make them easily accessible, and search through them. Topic modeling is one of the interesting research areas in machine learning that provides algorithmic tools to organize and exploit electronic archives. In fact, topic modeling is an application of hierarchical Bayesian models applied to grouped data like documents and images.

Topic modeling utilizes probabilistic models to generate a model for discovering the hidden thematic structure in a set of unstructured collection of documents. In a collection, documents contain different while overlapped subjects and themes. Using topic models, we can uncover the hidden thematic structure in the collection and further interpret each document based on the revealed hidden thematic pattern, which is called *topic*. A *topic* is interpreted as a list words in a fixed vocabulary sorted based on probability of their occurrence in the topic. In other words, a topic is a probability distribution of terms in a fixed vocabulary for the collection. Topics are usually represented by their top most probable words in them, since the vocabulary is sometimes so large.

Topic modelling is widely used to organize and arrange documents based on their content and is applied in many contexts including newspaper archives [15], scientific abstracts [25], and emails. Using topic modelling, we can find the trends within topics ([26] and [25]) and observe how words in a topic change through time [27]. Moreover, we can model the connections between topics and see how similar or close the individual topics are to each other [25]. Figure 3.1 demonstrates five random topics of a topic model fit to JSTOR's archive of the journal *Science*, [20].

The model used to discover the topics over the collection in Figure 3.1 is Latent Dirichlet Allocation (LDA).

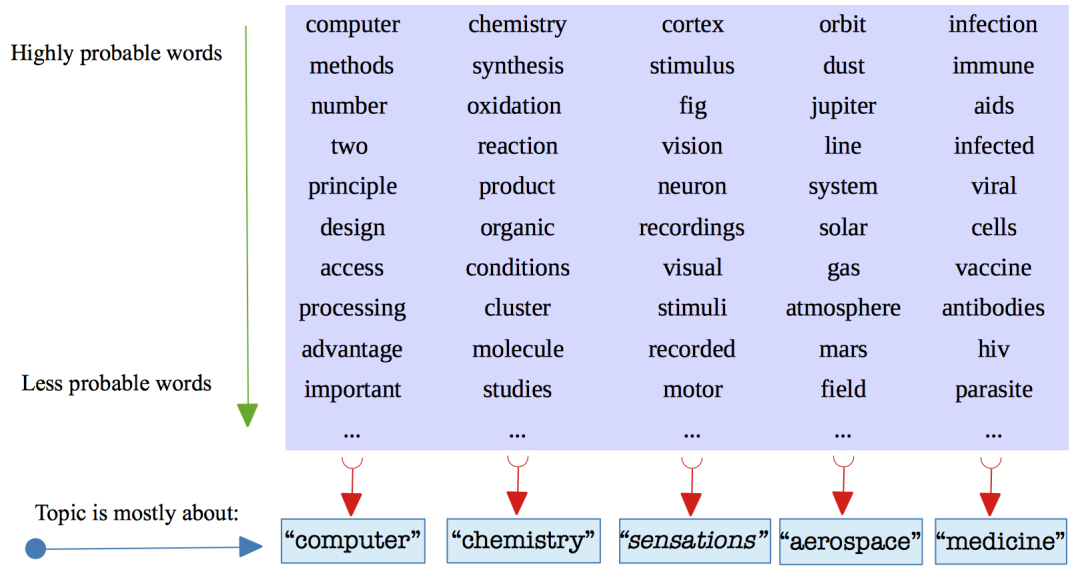


FIGURE 3.1: Five topics from a 50-topic model fit to Journal of *Science* from 1980-2002. Each topic is represented with its 10 top most frequent words. The words in a topic are thematically coherent.

3.1.1 Introduction to LDA

The basic intuition behind LDA is related to mixed membership models; a single document is a mixture of a specific number of topics with corresponding proportions, [16]. The inputs to an LDA model are documents. Theoretically, a document is a sequence of words. In LDA, documents are represented as bags of words (the position and order of words in document are not taken into account). The output of LDA is the discovered hidden structure of the inputs. The hidden structure is composed of topics exhibited from documents, per-document topic distributions, and per-document per-word topic assignments. The document's topic distribution can further be used for labelling the document and the topics information can be used for IR tasks.

LDA is a generative probabilistic model where we have two sets of variables: observed variables, i.e., bags of words per document, and hidden varibales, i.e., distribution of words per topic, distribution of topics per document, and topic assignments per-document per-word. The documents are not labeled and topics are unknown. The goal is to compute the conditional probability of hidden variables given the observed variables which is called posteriori.

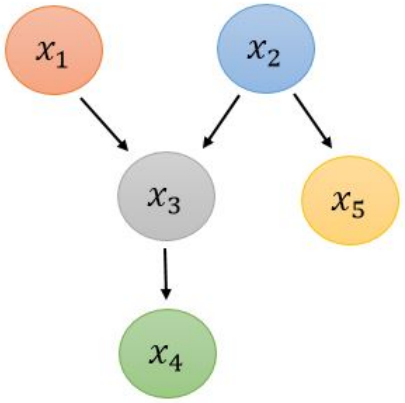


FIGURE 3.2: An example of a Bayesian Network with five random variables. By applying chain rule to this graph, we can write the following joint distribution of variables:

$$\begin{aligned} P(x_1, \ldots, x_5) &= \prod_{i=1}^{5} P(x_i | Par_G(x_i)) \\ &= p(x_1) \times p(x_2) \times p(x_3|x_1, x_2) \times p(x_4|x_3) \times p(x_5|x_2). \end{aligned}$$

3.1.2 Graphical Representation of LDA

To have a better perspective of the variables involved in LDA and their relationships and dependencies, LDA is represented as a directed graphical model, [17].

3.1.2.1 Bayesian Networks

Graphs are widely used to represent complex systems with large number of random variables where nodes are random variables and edges represent probabilistic relations among the random variables. A Bayesian Network (BN) is a directed acyclic graph, G, whose nodes represent the random variables $x_1, \ldots, x_n$. For each node x_i, a conditional probability distribution is defined as $P(x_i|Par_G x_i)$. The BN represents a joint probability distribution via the chain rule as follows:

$$P(x_1, \ldots, x_n) = \prod_i P(x_i | Par_G(x_i)). \tag{3.1}$$

An example of a BN with five random variables along with the open form of joint distribution is demonstrated in Figure 3.2.

By having the joint probability, we can answer different kinds of inferences and queries on our graph model. For instance, given the states of some variables (evidences), we want to find the state probability of other variables which is called conditional probability queries. In mathematical forms, if the state of a set of variables is known, $\mathbf{E} =$

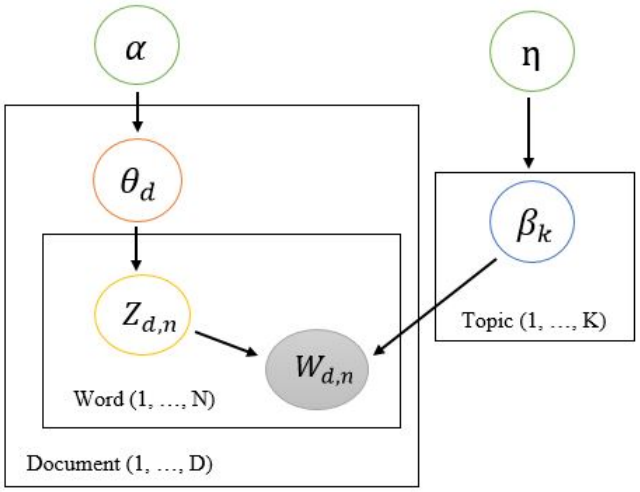


FIGURE 3.3: LDA is represented as a directed acyclic graph. The parameters α and η are Dirichlet hyper-parameters. The distribution β_k for $k = 1, \ldots, K$ represents the k^{th} topic distribution over words with a Dirichlet prior with parameter η. The distribution θ_d for $d = 1, \ldots, D$ represents the d^{th} document distribution over topics with a Dirichlet prior with parameter α. The assignment $Z_{d,n}$ represents the topic assignment for the $n^{th} word$ in d^{th} document. The observation $W_{d,n}$ represents the n^{th} word in d^{th} document.

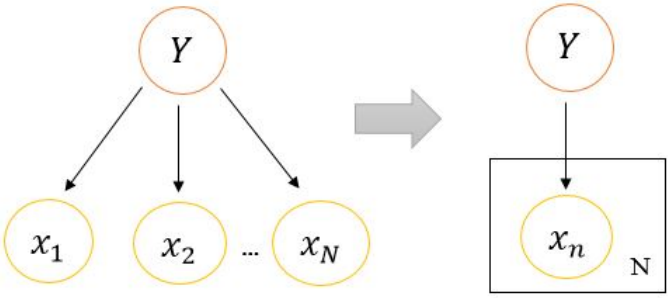


FIGURE 3.4: Plates for representing replication.

e, and the query is to find the probability of a subset of other variables, **y**, then in order to find the conditional probability query, we have to compute $P(\mathbf{y}|\mathbf{E} = \mathbf{e})$.

There are algorithms for finding an exact or approximate solution to such queries, like variable elimination, belief propagation, variational approximation, and random sampling instantiations. We will elaborate more on the sampling techniques in Section 3.1.3 for finding an approximate solution to the conditional queries.

Assume that we have a collection of D documents where each document consists of N vocabulary words. We are interested in detecting K topics using this document collection. Figure 3.3 best illustrates the LDA in a graphical form where we have K topics, D documents, and N words in each document; the plates in the graphical LDA correspond to the replication operation in the plate notation as shown in Figure 3.4.

Assume that topic distributions (β_k), topic proportions per-document (θ_d), and topic assignments per-word per-document ($Z_{d,n}$) are known. Using this LDA model, we can

generate a document by generating words of the document as follows: we randomly sample a K-topic proportion that will compose the document. Each word of the document is assigned to a topic. Using the topic proportion of the document, we sample a topic. Once the topic is chosen, we can sample a word from the multinomial distribution of the selected topic. The sampled word becomes the observed word. In this way, we generate a whole set of documents for a text collection. This is why LDA is called a generative probabilistic model.

Given the bayesian network in Figure 3.3, the joint probability of observed and hidden variables is as follows:

$$p(\beta_{1:K}, \theta_{1:D}, Z_{1:D}, W_{1:D}) = \Big(\prod_{k=1}^{K} p(\beta_k|\eta)\Big)\Big(\prod_{d=1}^{D} p(\theta_d|\alpha)\Big(\prod_{n=1}^{N} p(Z_{d,n}|\theta_d)p(W_{d,n}|Z_{d,n}, \beta_{1:K})\Big)\Big). \tag{3.2}$$

The joint probability of hidden and observed variables in Equation 3.2 can be computed from the generative process.

- $p(\beta_k|\eta)$ and $p(\theta_d|\alpha)$: Topics and topic proportions are K-dimensional and N-dimensional Dirichlet respectively.

$$p(\beta_k|\eta) = Dir(\eta), \tag{3.3}$$
$$p(\theta_d|\alpha) = Dir(\alpha). \tag{3.4}$$

  Dirichlet distribution is presented in more detail in Section 3.1.2.2.

- $p(Z_{d,n}|\theta_d)$: Given the topic proportion θ_d of document d, the probability of assigning a topic $Z_{d,n}$ to the n^{th}word of the document d is the proportion of that topic $Z_{d,n}$ in θ_d as shown in Figure 3.5.

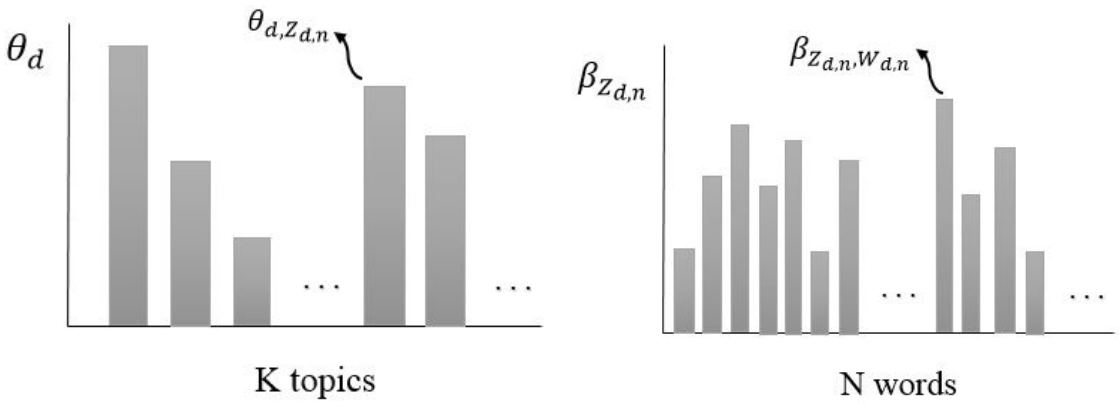
$$p(Z_{d,n}|\theta_d) = \theta_{d,Z_{d,n}}. \tag{3.5}$$

- $p(W_{d,n}|Z_{d,n}, \beta_{1:K})$: Given the topic assignment $Z_{d,n}$ of the n^{th} word of the document d and the assigned topic's distribution itself $\beta_{Z_{d,n}}$, the probability of the word $W_{d,n}$ being observed is the proportion of the word $W_{d,n}$ in $\beta_{Z_{d,n}}$ as shown in Figure 3.5.

$$p(W_{d,n}|Z_{d,n}, \beta_{1:K}) = \beta_{Z_{d,n},W_{d,n}}. \tag{3.6}$$

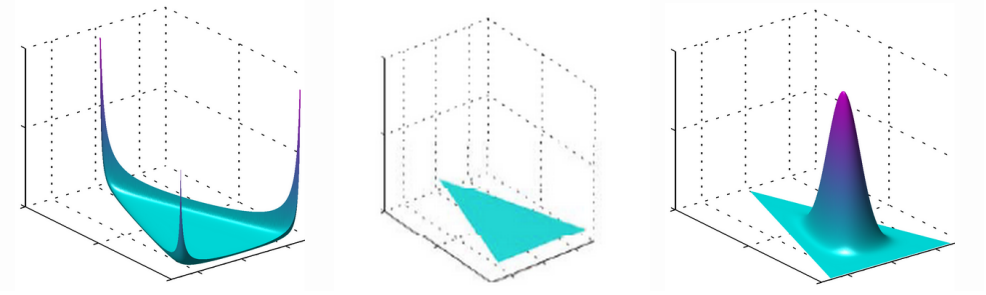
3.1.2.2 Dirichlet Distribution

Dirichlet distribution is named after one of the prominent mathematicians in 1800's, Johann Peter Gustav Lejeune Dirichlet. In Bayesian statistics, Dirichlet distribution



(A) Document d topic distribution. (B) Topic $Z_{d,n}$ word distribution.

FIGURE 3.5: Generation of a document using LDA.



(A) $\{\alpha_i\} = 0.1$ (B) $\{\alpha_i\} = 1$ (C) $\{\alpha_i\} = 10$

FIGURE 3.6: Dirichlet distributions for a 3-dimensional variable where the vertical axis corresponds to the density value and the horizontal axes correspond to the plane coordinates of the simplex. The distribution is a uniform distribution for $\alpha = 1$. For $\alpha > 1$, the maximum lies somewhere near the center. For $\alpha < 1$, the maximum lies somewhere near the corners of the triangle.

is often used as a prior distribution and is denoted by $Dir(\alpha)$ and is parameterized by a vector of positive and αs. For a discrete n-dimensional random variable θ, $\theta = (\theta_1, \ldots, \theta_n)$ is distributed under Dirichlet distribution with parameter α, and the density function $p(\theta)$ is defined as follows:

$$\theta \sim Dir(\alpha) \quad \Leftrightarrow \quad p(\theta|\alpha) = \frac{1}{\beta_\alpha} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1} \mathbf{1}(\theta \in S), \tag{3.7}$$

where $\frac{1}{\beta_\alpha} = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)}$, $\Gamma(n) = (n-1)!$, and $\mathbf{1}(\theta \in S)$ is the indicator function over the probabilistic simplex S. A probabilistic simplex is a set of n-dimensional vectors that sum to one; $S = \{x \in \mathbb{R}^n : x_i > 0 \text{ and } \sum_i x_i = 1\}$.

Dirichlet distribution is an exponential family distribution over a probabilistic simplex (it is a distribution over probability distribution). For example in a three dimensional space, the possible values for θs lie on a triangle surface with $\theta_1 + \theta_2 + \theta_3 = 1$. With different values for αs, different kinds of distributions can be obtained. Figure 3.6 demonstrates Dirichlet distributions for a 3-dimensional variable with three different αs.

One interesting property of Dirichlet distribution is **conjugacy**. If a data point comes from a multinomial distribution, and the prior distribution of the parameter for the data point (the vector of probabilities that generates the data point) is distributed as a Dirichlet, then the posterior distribution of the parameter is also a Dirichlet. In mathematical forms, we have:

$$\begin{aligned} \theta &\sim Dir(\alpha), \\ Z_n &\sim Mult(\theta), \\ \rightarrow \quad p(\theta|Z_{1:N}) &= Dir(\alpha + n(Z_{1:N})), \end{aligned} \tag{3.8}$$

where $Mult(\theta)$ denotes a multinomial distribution over θ and $n(Z_{1:N})$ denotes the total number of topic assignments in a document per topic. For example, $n_{11} = 3$ means that the 11th topic occurs 3 times in the document.

3.1.2.3 LDA Posteriori

Posterior or the conditional probability of the hidden variables given the observed variables is computed as:

$$p(\beta_{1:K}, \theta_{1:D}, Z_{1:D}|W_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, Z_{1:D}, W_{1:D})}{p(W_{1:D})}. \tag{3.9}$$

The numerator is the joint probability of the observed and the hidden variables as in Equation 3.2. The denominator is the probability of the observed variables. This factor is the marginal probability of the observed variables and can be computed as follows:

$$p(W_{1:D}) = \sum_{\beta_{1:K}, \theta_{1:D}, Z_{1:D}} p(\beta_{1:K}, \theta_{1:D}, Z_{1:D}, W_{1:D}). \tag{3.10}$$

However, this summation is not tractable in practice because we have a very large number of possible topical structures and the sum is over all possible assignments of words in all documents to each topic. In Section 3.1.3, we discuss how to approximate this factor.

3.1.3 Approximation of Posterior Inference

In summary, LDA infers the following values from a collection of documents:

- Per-word topic assignment $Z_{d,n}$,
- Per-document topic proportions θ_d, and
- Per-topic word distributions β_k.

The posterior expectations from the LDA model can be used to perform end-user tasks such as finding similar documents, finding similar topics, and summarizing documents.

There are different algorithms to approximate these posterior inferences, such as mean field variational methods ([18] and [19]), expectation propagation ([21] and [22]), collapsed variational inference ([23] and [24]). We used Markov chain Monte Carlo algorithm ([30] and [31]) and collapsed Gibbs sampling ([25]) for doing the inferences. There are evaluation metrics proposed in [28] and [29] for comparing the performance of inferences.

3.1.3.1 Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo is an algorithm to sample by using a markov chain from a distribution that is intractable to directly sample from. A markov chain defines a probabilistic transition model $T(x \to x')$ over states x such that:

$$\text{for all x:} \quad \sum_{x'} T(x \to x') = 1. \tag{3.11}$$

Given the current states x, the transition model determines the likelihood of the transition from state x to state x'. The next state x' becomes the current state, and the process repeats. The temporal dynamics of this process is represented by:

$$P^{(t+1)}(x^{(t+1)} = x') = \sum_{x} p^{(t)}(x^{(t)} = x) T(x \to x'). \tag{3.12}$$

A markov chain is associated with a unique stationary distribution Π, where the probability of being in a state is the same as the probability of transitioning to this state from a randomly sampled predecessor. Formally, we have:

$$\begin{aligned} p^{(t)}(x') \simeq p^{(t+1)}(x') &= \sum_{x} p^{(t)}(x) T(x \to x'), \\ \Pi(x') &= \sum_{x} \Pi(x) T(x \to x'). \end{aligned} \tag{3.13}$$

MCMC algorithm uses a markov chain T whose stationary distribution is close enough to our target distribution P which is intractable to sample from directly. The first step is to sample an initial state from an arbitrary distribution and then use the transition model in Equation 3.12 for generating new state, i.e., a new sample. Since the chain converges to a stationary distribution, we would eventually have a sample that is very close to a sample from P. Samples at the beginning are far from P. We have to repeat the process till it converges to the stationary distribution of the chain and that would be the point where the real samples are being collected. This state is called **mixing**. After this point, samples from the chain mixes are from stationary distribution Π. MCMC algorithm is summarized in Algorithm 1. Note that we can run multiple markov chains in parallel in MCMC algorithm to speed up sampling process.

Gibbs sampling is one of the most powerful sampling method for generating samples that are close to samples from the original distribution. The idea in Gibbs sampling is to sample each variable given the current state of other variables. Algorithm 2 summarizes the procedure in Gibbs sampling.

Initialize Sample $x^{(0)}$ from an arbitrary distribution;
repeat
  Generate $x^{(t+1)}$ from $T(x^{(t)} \rightarrow x')$;
  Check if mixing has occurred (comparing window statistics in chain);
  $t := t + 1$;
until *mixing*;
repeat
  S = $\emptyset$ (set of samples);
  Generate $x^{(t+1)}$ from $T(x^{(t)} \rightarrow x')$;
  $S := S \cup x^{(t+1)}$;
  $t := t + 1$;
until *obtain enough samples*;
Let $S = \{x[1], ..., x[M]\}$ (M collected samples);
Estimate $E_p \simeq \frac{1}{M} \sum_{m=1}^{M} x(M)$ (empirical expectation);

Algorithm 1: Markov Chain Monte Carlo (MCMC) algorithm.

Original distribution: $P(x_1, ..., x_n)$;
Markov chain state space: all variables $\{x_1, ..., x_n\}$;
Initialize: $x^{(0)}$;
repeat
  for *i = 1:n* **do**
    Sample $x_i \sim P(x_i|x_{-i})$
    (x_{-i} indicates the set of all variables except for x_i);
  end
  Set $x' = x$;
until *obtain enough samples*;

Algorithm 2: Realization of a Markov chain using Gibbs sampling.

3.1.3.2 LDA Posterior Approximation

The state space of Gibbs sampling on LDA posterior in Equation 3.9 includes θ and $z_{1:N}$ since words are observed and we assume that topics are fixed. For each document, the sampling is done in two steps:

Step 1. Sample θ given the current state of hidden variables $Z_{1:N}$ and observations $W_{1:N}$. According to the graphical representation of LDA in Figure 3.3, θ is independent of W given Z. Due to the conjugacy of Dirichlet (see Equation 3.8), θ can be sampled from the following distribution:

$$p(\theta|Z_{1:N}, W_{1:N}) = P(\theta|Z_{1:N}) = Dir(\alpha + n(Z_{1:N})), \tag{3.14}$$

where α is the hyper-parameter of θ prior distribution and $n(Z_{1:N})$ denotes assignment counts per-topic in a document.

Step 2. Sample Z_i given the value of other Z's words and θ. This conditional probability can be written as (to be normalized afterwards):

$$p(Z_i = z_i | Z_{-i}, W_{1:N}, \theta) \propto p(Z_i = z_i|\theta) \times p(W_i = w_i|\beta_{1:K}, Z_i). \tag{3.15}$$

A modified version of Gibbs sampling is called **collapsed Gibbs sampling**, which converges faster than the normal Gibbs sampling. In collapsed Gibbs sampling, the state space of Markov chain reduces to just $Z_{1:N}$ by marginalizing over the topic proportions θ due to the fact that θ is a conjugate prior for $Z_{1:N}$. Therefore, the conditional probability for sampling $Z_{1:N}$ takes the following form:

$$p(Z_i = z_i | Z_{-i}, W_{1:N}) \propto p(W_i = w_i | Z_i = z_i, Z_{-i}, W_{-i}) p(Z_i = z_i | Z_{-i}), \tag{3.16}$$

where Z_{-i} is the assignments of all Z_k such that $k \neq i$. The above proportion is derived from the following property in probability theory:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \propto P(B|A)P(A)$$

It is shown in [16] that the proportion in 3.16 can be rewritten in the following form:

$$p(Z_i = z_i | Z_{-i}, W_{1:N}) \propto \frac{n_{-i,j}^{(w_i)} + \eta}{n_{-i,j}^{(.)} + N\eta} \times \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,.}^{(d_i)} + K\alpha}, \tag{3.17}$$

where $n_{-i,j}^{(w_i)}$ is the total count of word i assigned to topic j not including the current one, $n_{-i,j}^{(.)}$ is the total number of words assigned to topic j not including the current one, $n_{-i,j}^{(d_i)}$ is the number of words from document d_i assigned to topic j not including the current one, and $n_{-i,.}^{(d_i)}$ is the total number of words in document d_i not including the current one.

Algorithm 3 provides the pseudocode for posterior approximation of LDA using collapsed Gibbs sampling.

Parameters and Variables :

Symbol	Definition	Description
$\boldsymbol{Z}_{D\times N}$	$\boldsymbol{Z}[d,n] = z_{d,n}$	$\boldsymbol{Z}[d,:] \equiv$ per-word per-document topics assignments
$\boldsymbol{\theta}_{D\times K}$	$\boldsymbol{\theta}[d,k] = \theta_{d,k}$	$\boldsymbol{\theta}[d,:] \equiv$ per-document topics proportions
$\boldsymbol{\beta}_{K\times N}$	$\boldsymbol{\beta}[k,n] = \beta_{k,n}$	$\boldsymbol{\beta}[k,:] \equiv$ per-topic words proportions
$\boldsymbol{F}_K$	$\boldsymbol{F}[k] = f_k$	$f_k \equiv$ frequency of topics in the collection

D = length(Collection);
K = length(Topics);
N = length(Document);
$\boldsymbol{Z}_{D\times N} = zeros(D, N)$;
$\boldsymbol{\theta}_{D\times K} = zeros(D, K)$;
$\boldsymbol{\beta}_{K\times N} = zeros(K, N)$;
$\boldsymbol{F}_K = zeros(K)$;

Initialize:
for *d = 1:D* **do**
  $\boldsymbol{Z}_N = zeros(1, N)$;
  for *n = 1:N* **do**
    $z \leftarrow$ randomly assign a topic to word n;
    $\boldsymbol{Z}_N[n] = z$;

    $\boldsymbol{\theta}_{D\times K}[d, z]$+=1; # add 1 to the count of topic z in document d
    $\boldsymbol{\beta}_{K\times N}[z, n]$+=1; # add 1 to the count of word n in topic z
    $\boldsymbol{F}_K[z]$+=1;
    # add 1 to the count of topic z (keep the count of topic z in the whole collection)
  end
  $\boldsymbol{Z}_{D\times N}[d,:] \leftarrow \boldsymbol{Z}_N$;
end

Posterior Approximation:
for *d = 1:D* **do**
  for *n = 1:N* **do**
    # remove word n from its associated topic
    $z = \boldsymbol{Z}_{D\times N}[d, n]$; # find assigned topic to word n
    $\boldsymbol{\theta}_{D\times K}[d, z]$ -= 1; # subtract 1 from the count of topic z in document d
    $\boldsymbol{\beta}_{K\times N}[z, n]$ -= 1; # subtract 1 from the count of word$n$ in topic z
    $\boldsymbol{F}_K[z]$ -= 1;
    # subtract 1 from the count of topic z (keep the count of topic z in the whole collection)

    # assign new topic to word n based on Equation 3.17
    $\tilde{\boldsymbol{P}}_z = \frac{\boldsymbol{\beta}_{K\times N}[:,n]+\boldsymbol{\eta}}{\boldsymbol{F}_K[z]+N\times\eta} \times (\boldsymbol{\theta}_{D\times K}[d,:] + \alpha)$;
    $\boldsymbol{P}_z \leftarrow$ normalize $\tilde{\boldsymbol{P}}_z$;
    $z_{new} \leftarrow$ from distribution $\boldsymbol{P}_z$ choose a topic;

    # update the values
    $\boldsymbol{Z}_{D\times N}[d, n] = z_{new}$;
    $\boldsymbol{\theta}_{D\times K}[d, z_{new}]$+=1;
    $\boldsymbol{\beta}_{K\times N}[z_{new}, n]$+=1;
    $\boldsymbol{F}_K[z_{new}]$+=1;
  end
end

Algorithm 3: LDA and posterior approximation.

Chapter 4

Methodology

4.1 Problem Statement

The goal is to predict CR of keywords using a set of their features. A promising approach is to train a predictor which learns how to predict CR of new keywords based on the previous seen keywords. In machine learning, this predictor is called hypothesis h. For further use, we utilize the following notations.

$$
\begin{aligned}
f^{(i)} &: i^{th} \text{ input variable, i.e., } i^{th} \text{ keyword's features,} \\
y^{(i)} &: i^{th} \text{ target variable, i.e., } i^{th} \text{ keyword's CR,} \\
(f^{(i)}, y^{(i)}) &: \text{a training example,} \\
f_j &: j^{th} \text{ feature,} \\
n &: \text{number of features,} \\
m &: \text{number of training examples used for training.}
\end{aligned}
$$

Given training examples, the goal is to learn a function $h : F \to Y$ such that $h(f)$ is a good predictor for the corresponding value of y. The process is depicted in Figure 4.1. In order to train our hypothesis, the next step is to extract a set of features.

4.2 Feature Extraction

There are many feature categories we can use. Determining the right set of features is important to learn a good predictor.

4.2.1 Numeric Features

For keywords, we extract numeric features from data that are likely to result in better prediction of conversions. We briefly describe these features.

- Clicks: Total number of user clicks,

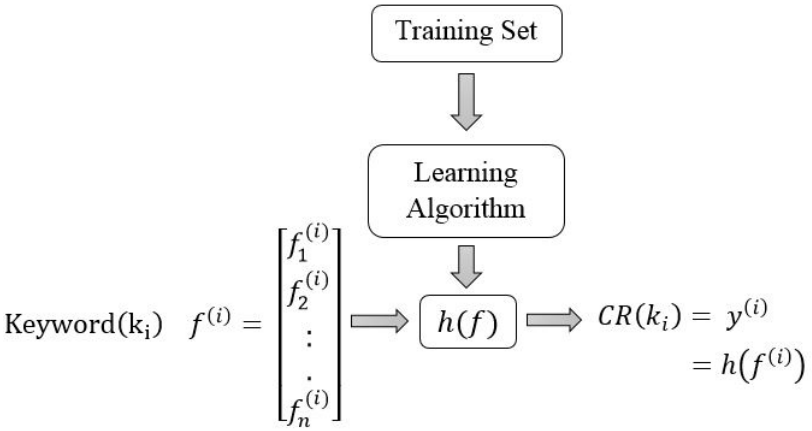


FIGURE 4.1: How to apply the Learning algorithm ($f_j^{(i)}$ represents the j^{th} feature of the i^{th} keyword).

- Avg. CPC: Average amount that advertiser has been charged for user clicks. This amount is the total cost of all clicks divided by the total number of clicks received. Average CPC is not the same as maximum CPC (bid), which is the maximum amount advertiser is willing to pay for a click. For example, if an ad receives two clicks (the first costing \$0.20 and the second costing \$0.40), the average CPC for those clicks is \$0.30,
- Avg. position: Average of position in SERPs,
- Bounce rate: Percentage of users who visit the page and leave immediately without taking any further action, and
- Quality score: Score given by the search engine to keyword. The score influences both position and CPC.

4.2.2 Average CR of Keywords in Same Ad-group

The conversion of keywords in the same ad-group are similar. Therefore, a potential feature that can predict the target CR is to use CRs of all other keywords in the same ad-group. A naive approach is to compute the average of all CRs in the ad-group. For each keyword, this feature is calculated using all the keywords in the same ad-group excluding the keyword.

4.2.3 Average CR of Keywords in Same Campaign

Similar to the previous feature, another potential feature is to average the CRs of all other keywords in the same campaign.

TABLE 4.1: Different match types for keywords in sponsored search advertising. The definitions are from Google Adwords Support [1].

Match type	Special symbol	Example keyword	Ads may show on searches that ...	Example searches
Broad match	none	java class online	include misspellings, synonyms, related searches, and other relevant variations	java tutorial
Broad match modifier	+keyword	+java +class +online	contain the modified term (or close variations, but not synonyms) in any order	online class for java
Phrase match	"keyword"	"java class online"	are a phrase, and close variations of that phrase	available java class online
Exact match	[keyword]	[java class online]	are an exact term and close variations of that exact term	java class online
Negative match	-keyword	-java	are searches without the term	python tutorial

4.2.4 Match Types

There are three main match types for matching keywords to queries: broad match, phrase match, and exact match. Each match type is explained in Table 4.1. The conversion performance of keywords varies with match type. In order to accommodate the preconditions for all match types, we leave keywords as they are without pre-processing them (such as removing punctuation, or removing stop words).

4.2.5 Non-linear Features

Instead of working with linear features only, one can engineer non-linear features via $\log(f_i + 1)$ and f_i^2 where f_i is the i^{th} feature. Fabricating non-linear features has two main benefits: the number of features are increased and more complex functions can be fit to the data.

4.2.6 Topic based Features

Documents are more than bags of words; each has a main theme that governs it. Keywords are treated as short documents. If a keyword A results in a conversion, another keyword B that shares the same thematic structure with A might result in conversion as well. Topics or themes represent the hidden thematic structure of keywords, and they can be extracted using LDA. For more information on LDA, refer to Section 3.1.

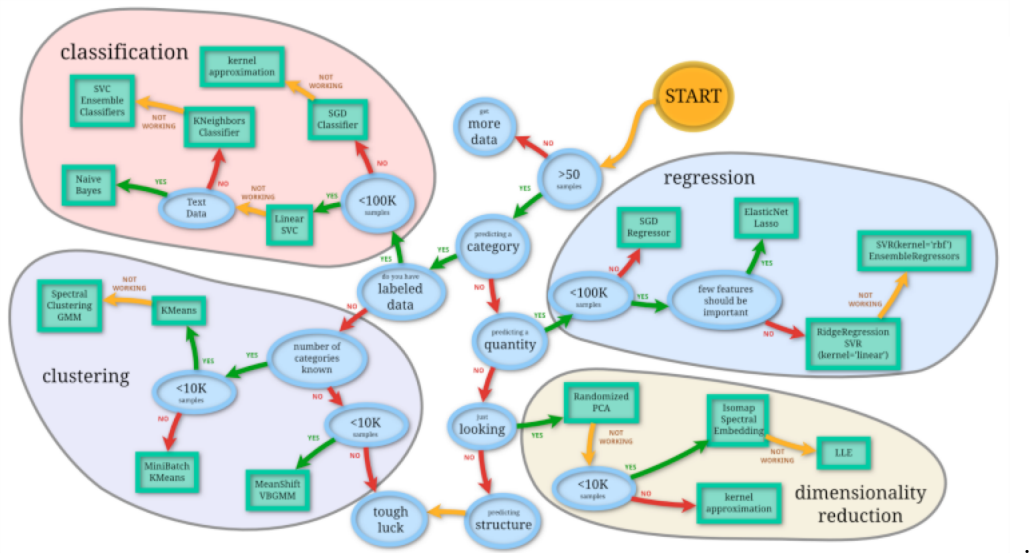


FIGURE 4.2: A flow chart for choosing the right machine learning algorithm to use in different application characteristics.

Once a K-topic model is computed on the keywords, each keyword will have a corresponding topic distribution of K topics, i.e. K probability values that sum up to 1. This K dimensional feature is used in our CR model.

4.3 Model Construction

Figure 4.2 provides a general guideline on how to tackle machine learning problems. The flow chart is provided by the scikit-learn community [32]. In our case, predicting CR is a regression problem. The term regression refers to the fact that the target variable is a continuous value (CR can take real values between zero and one). We use conversion data to find an association between keyword features (input) and CR (output).

Given the notation above, fs are n-dimensional vectors in $\mathbb{R}^n$. For instance, $f_1^{(i)}$ might be the quality score of the i^{th} keyword in the training set. To perform supervised learning, we must decide how to represent hypothesis h. We use a linear model to approximate y as a linear function of f. Since we are using non-linear forms of our basic features, our model also accommodates those cases where there is a non-linear relationship between our CR target and keyword features.

$$\begin{aligned} h(f) &= w_0 + w_1 f_1 + ... + w_n f_n, \\ &= \sum_{i=0}^{n} w_i f_i =< \mathbf{w}, \mathbf{f} >, \end{aligned}$$

where w_i is the weight and $f_0 = 1$ to simplify the notation and n is the size of our feature space (excluding f_0).

In order to learn ws, we want to make $h(f)$ as close as possible to y for our training examples. Formally, we choose the following least-squares cost function $J(w)$:

$$J(w) = \frac{1}{2} \sum_{i=1} m(h(f^{(i)}) - y^{(i)})^2.$$

In order to minimize the cost function over w, one can use **Gradient Descent** algorithm, which starts with some initial vector **w** and repeatedly performs the following simultaneous update:

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w), \qquad \text{for} \quad j = 0, ..., n \tag{4.1}$$

where α is called the learning rate and controls the convergence rate. Algorithm 4 outlines the gradient descent algorithm.

Data: $f^{(i)}$ and $y^{(i)}$, for $i = 1, ..., m$;
Result: w_j, for $j = 0, ..., n$;
Initialize vector of w; **repeat**
  $w_j := w_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h(f^{(i)})) f_j^{(i)}$; # simultaneous update for $j = 0, ..., n$
until *convergence*;

Algorithm 4: Gradient Descent Algorithm.

In each step of the gradient descent update, the weights are updated according to the gradient of the cost function with respect to all training examples. In other words, the algorithm needs to scan through the entire training set before taking a single step. It is a costly operation for a large training set and slows down the convergence. There is a faster alternative to this algorithm, which is called **Stochastic Gradient Descent**. In stochastic gradient descent, the update considers one training example at a time for computing the gradient. Algorithm 5 outlines the stochastic gradient descent algorithm.

Data: $f^{(i)}$ and $y^{(i)}$, for $i = 1, ..., m$;
Result: w_j, for $j = 0, ..., n$;
Initialize vector of w;
repeat
  for $i = 1 : m$ **do**
    $w_j := w_j + \alpha \times (y^{(i)} - h(f^{(i)})) f_j^{(i)}$; # simultaneous update for $j = 0, ..., n$
  end
until *convergence*;

Algorithm 5: Stochastic Gradient Descent algorithm.

We use the least-squares function as our cost function and use stochastic gradient descent for the minimization of this cost function. In order to compare the CR prediction of our model, we use the average CR of all keywords in the training set as the base model.

Chapter 5

Experiments

In this chapter, we present our experiment setup and the results of our empirical evaluation.

5.1 Experiment Settings

5.1.1 Data Description

The search campaign data used in our experiments is from a SaaS startup that offers courses online and corresponds the time period from Dec 2013 to July 2014. There are 52,000 keywords in 260 campaigns.

Figure 5.1 illustrates the relationship between some keyword attributes. As we can see in Figure 5.1a, keywords with higher number of clicks tend to have low average position (average position less than two). This is along with the fact that higher positions attract more visual attention and result in getting more clicks. Figure 5.1b suggests that for conversion rate, the effect of position is not the same as of clicks. Keywords with relatively high CR tend to have average position of more than two.

Figure 5.1c reveals the average position versus quality score of keywords. The CR performance is depicted in Figure 5.1d for campaigns, in Figure 5.1e for keywords. Figure 5.1f depicts the CR distribution of keywords.

The cumulative distributions of CR over keywords and campaigns are demonstrated in Figure 5.2.

The plots in this section were drawn on a subset of data where keywords with less than 10 clicks were removed. There are many keywords with zero conversions, and this results in the tighter probability distributions and plots.

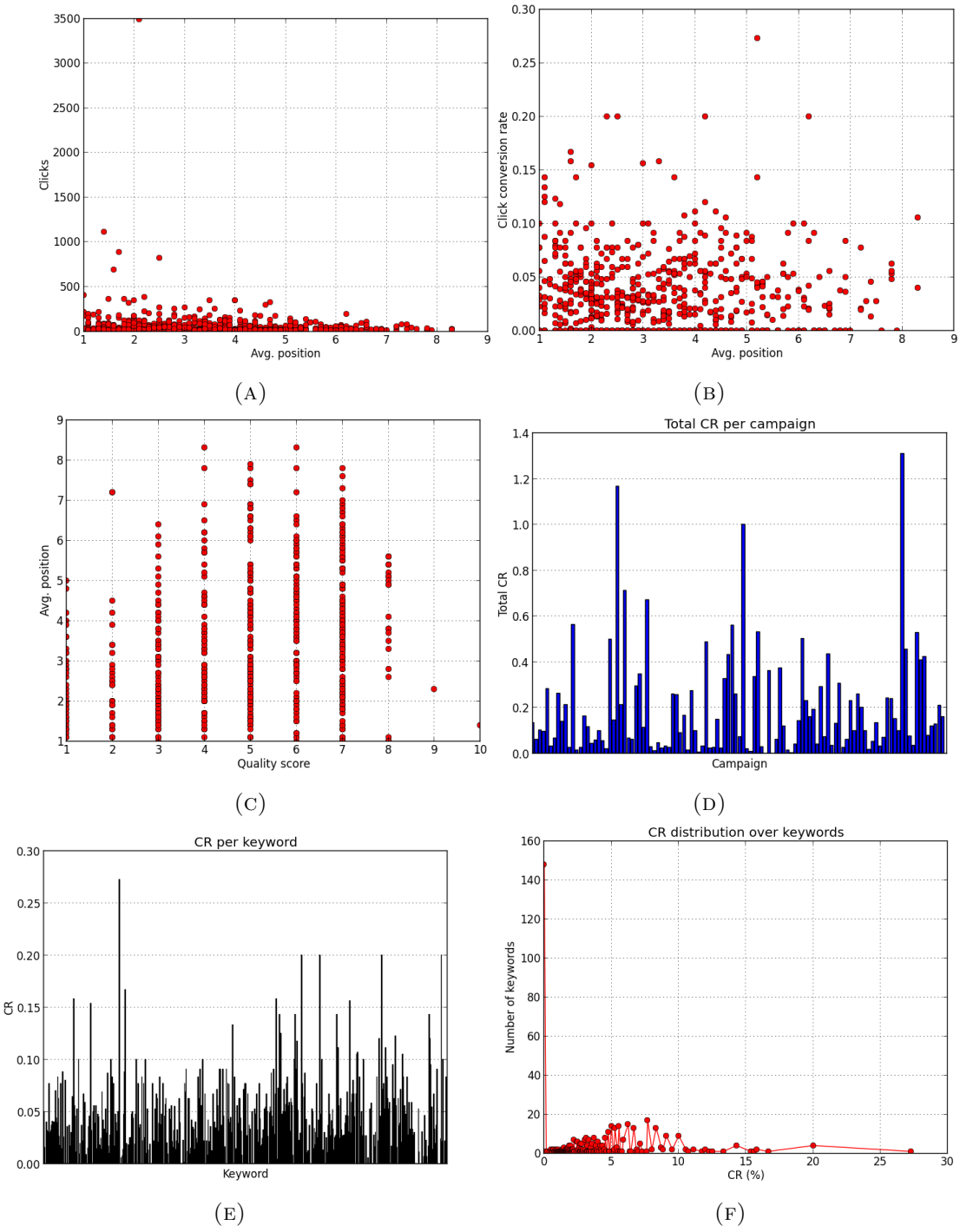


FIGURE 5.1: These plots were drawn on a subset of data where keywords with less than 10 clicks were removed. (a) Number of clicks versus average position of keywords. The lower the average position is, the more clicks are received. The lowest value for position is one. (b) Click conversion rate versus the average position over keywords. (c) Average position versus quality score of keywords. The position is determined by the quality score given by the search engine, and the specific bid used for the keyword. In cases where bids are the same for different keywords, the one with the higher quality score gets a higher position. (d) Total conversion values in all the campaigns. (e) CR of all keywords. (f) CR distribution over keywords.

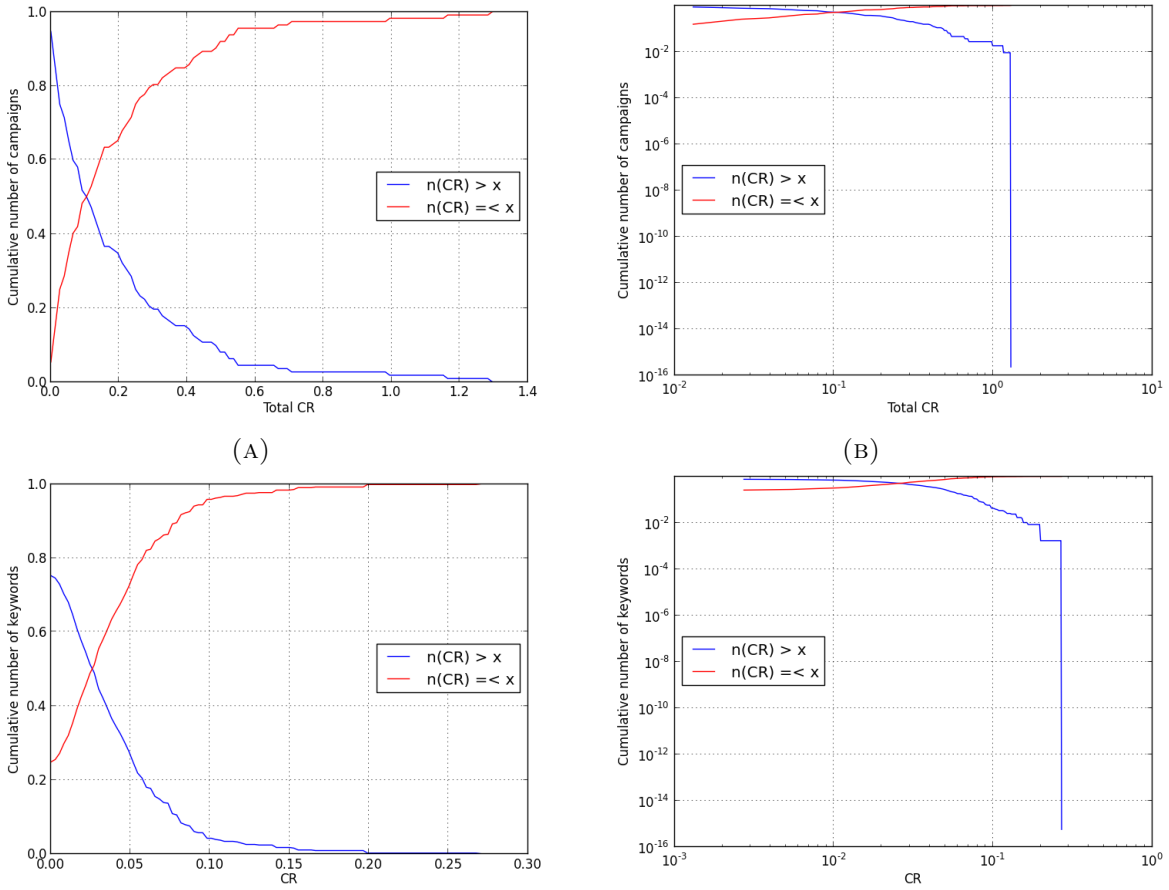


FIGURE 5.2: The plots in red are cumulative distributions where keywords with less than or equal to a threshold are included. Where as in blue plots, the keywords with CR greater than a threshold are included. The number of bins is set to 100 for these plots. (a) Probability distribution of CR over campaigns. (b) Probability distribution of CR over campaigns in logarithmic scale. (c) Probability distribution of CR over keywords. (d) Probability distribution of CR over keywords in logarithmic scale.

5.1.2 Pre-processing Data

5.1.2.1 Cleaning Data

Non-word characters were removed. Non-numeric characters from the numeric parts of the data, e.g, % sign for bounce rate, were removed. Moreover, we want to estimate the true CR, but in most of the cases the data contains a small number of clicks and conversions. For keywords with too few clicks, the predicted CR might be far off from the true CR. This results in much noise in our training and testing process. To address this issue, we filtered out keywords with less than 10 clicks.

5.1.2.2 Scaling Attributes

Scaling features causes the algorithm to converge in less iterations. There are two main approaches for scaling the attributes:

1. Min-max normalization: real data is linearly transformed such that the minimum value is set to zero and the maximum value is set to one.

$$f_j := \frac{f_j - min(f)}{max(f) - min(f)},$$

where $min(f)$ and $max(f)$ are the minimal and maximal values for each attribute respectively. With this method of scaling, all values fall in between zero and one.

2. Z-score normalization: real data is linearly transformed such that the mean of the transformed data is equal to zero with unit standard deviation. The Z-score normalization is also called Gaussian normalization with zero mean and unit variance. With this method of scaling, the whole data does not necessarily lies in a particular interval. Each feature is scaled as follows:

$$f_j := \frac{f_j - \bar{f}}{\sigma},$$

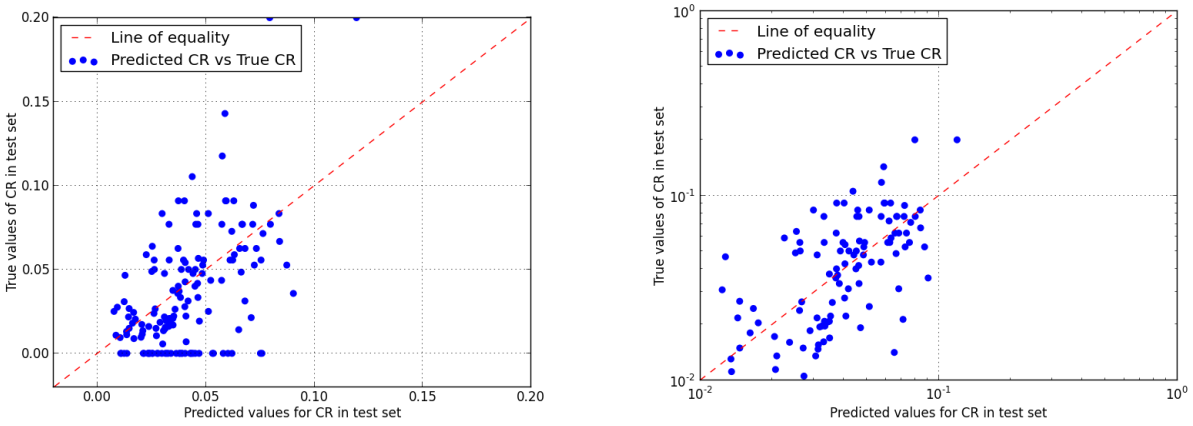
where $\bar{f}$ denotes the mean value and σ denotes the standard deviation of the feature.

Both scaling methods were used for scaling our features. Our experiments suggested that Z-score is a better scaling metric for our dataset. Since our data is very sparse, a non-zero value is of great value while min-max normalization sets most of the values to zero and makes CR prediction impractical. The mean and standard deviation per feature in the training set were used for scaling the same feature in the testing set.

5.1.2.3 Training, Cross Validation, and Testing

We split the data into training and testing sets by a split ratio. Then, we trained the model on the training set and tested it using the test set. Due to the sparsity of the dataset, a large number of zero CRs can fall in one set while the other set contains more non-zero values. This could affect the model and its accuracy. In order to address this issue, we created a list of non-zero CRs and a list of zero CRs. Then, we divided each of the lists into training and testing sets by the split ratio (the lists were shuffled before the splits). Finally, we joined the training sets of zero CRs and non-zero CRs into one final training set and one final testing set.

By splitting the data into two sets, the model we built correlated with the choice of the training samples. As the training set changes, the model would change slightly. In order to address this issue, we sampled the keywords for training and testing according to the split ratio and then carried out the experiment for multiple times. We computed the error in each experiment. The final reported error was the average of all errors. This methodology is similar to k-fold cross validation but has a subtle difference. The k-fold cross validation is used for validating the model and its parameters. It splits the training set into training and validation sets. The error, however, is calculated on the untouched



(A) True CR vs. predicted CR.

(B) True CR vs. predicted CR on a logarithmic scale.

FIGURE 5.3: Evaluating the performance of the model by comparing true CR and predicted CR with respect to the line of equality. The features used for these plots were numeric features, average CR of ad-group and campaign, topic-based, and nonlinear features. The closer blue dots are to the line y = x, red line, the better the performance of the model is.

testing set. Therefore, we run the experiment multiple times for different choices of the training and testing sets and averaged the error over all the runs.

5.1.3 Evaluation Rules

5.1.3.1 Evaluation Protocols

Data was split into training and testing sets by a split ratio of 0.75 (75% of data went into training set and the 25% remaining went into test set).

5.1.3.2 Evaluation Metrics

Mean squared error (MSE) and R^2 score were used as evaluation metrics for our model. MSE measures the average of the squares of the difference between the estimate and the actual value. R^2 score represents how well the observed outcomes are replicated by the model as a proportion of the total variation of outcomes explained by the model. The interested reader can refer to Appendix A for more details on MSE, R^2 score, and other useful measures.

5.2 Experimental Results

The results of training a linear model using SGD regression function are given in Figure 5.3. It shows how well the model predicts the true CRs. Tables 5.1, 5.2, and 5.3 shown the prediction errors made on a test set with different choices of features.

TABLE 5.1: Performance evaluation of the model for a single feature.

Features	SGD with regularization		
	MSE (*e-3)	R^2 score	%Improvement
$\overline{CR}$	1.287	-	-
Match Type	1.296	-0.008	-0.81
Topics ($K = 70, \alpha = 1$)	1.198	0.068	6.83
Avg CR of ad-group	1.199	0.068	6.84
Avg CR of campaign	1.162	0.096	9.70
Numeric	1.058	0.178	**17.80**

TABLE 5.2: Performance evaluation of the model under Topic-model derivatives.

Features	SGD with regularization		
	MSE (*e-3)	R^2 score	%Improvement
$\overline{CR}$	1.287	–	–
Topics ($K = 50, \alpha = 0.01$)	1.27	0.011	1.14
Topics ($K = 50, \alpha = 1$)	1.206	0.062	6.18
Topics ($K = 70, \alpha = 1$)	1.198	0.068	**6.83**

TABLE 5.3: Performance evaluation of the model under different choices of feature combinations. The numeric features were always used in the combinations. The method $\overline{CR}$ corresponds to the base model and it represents the average CR obtained in the training set.

Features	SGD with regularization		
	MSE (*e-3)	R^2 score	%Improvement
$\overline{CR}$	1.287	–	–
Numeric	1.058	0.178	17.8
+Topics ($K = 70, \alpha = 1$)	1.039	0.192	19.20
+ Non-linear of numerics (f_i^2 and $\log(f_i + 1)$)	1.015	0.210	21.13
+ Avg CR of ad-group & campaign	0.957	0.256	25.64
+ Avg CR of ad-group & campaign & Topics ($K = 70, \alpha = 1$) & Non-linear	0.941	0.268	**26.81**

Chapter 6

Conclusion

In this study, we built a prediction model to estimate true CRs of unknown keywords based on the available statistics for existing keywords in an online campaign. We separated the available data into two sets as training and testing for learning a model of CR. The training set contained 75% of the whole data and was used to build the CR model. The remaining part of the data was used for testing the performance of the model built.

We used a regression linearized learning model for predicting CR. In order to build our CR model, we considered a set of parameters that affect CR of keywords. We used stochastic gradient descent algorithm to learn the parameter weights.

Per keyword advertised for ample time, an advertisement broker, e.g., Google Adwords and Microsoft Bing provides quality score, bounce rate, average position, and number of clicks. These data points can readily be used for the model. However for new keywords, there is not much performance data available because they either have not been advertised yet or they are advertised but there is not enough time passed to accrue meaningful statistics. Since keywords consist of terms and terms are more likely to have appeared in previous seen and advertised keywords, one can use the available statistics for the terms instead. This insight led us to use probabilistic probabilistic inference for extracting meaningful features on keywords. Advanced features such as thematic structure (topical structure) improved the performance of our CR prediction model.

Advertisers themselves provide supervision by placing keywords with similar characteristics in an ad-group or a campaign. An ad-group is a collection of ads and keywords. We made use of this supervision by using average CR per ad-group and per campaign as a keyword feature.

Since we used a linear regression hypothesis for our model, we increased the complexity of our model by adding nonlinear features, e.g., by taking the logarithm (plus 1) of a numeric feature and by taking the square of a numeric feature. Such nonlinearity improved our model's predictive power.

An interesting finding is on the performance of keyword match type. Each keyword

has a match type. One could expect that the exactly matched keywords result in better conversion since the keyword matches the user query exactly. Hence, it is more likely that the user will convert. However, our results contradicted our naive expectation. Keywords with match type as broad-match modifier (i.e., +keyword) tend to convert much better than keywords with other match types.

Our major contribution to the field of performance marketing is that we constructed a hybrid prediction model that combined text and numeric features per keyword. The resulting CR model achieved a superior predictive power compared to using text features only or to using numeric features only.

Appendix A

Terms and Evaluation Metrics

A.1 Evaluation Metrics Used in Ranked Results

- Kendall Rank Correlation Coefficient (Kendall Tau Distance) [33]:

It counts the number of pairwise agreements and disagreements between two ranking lists.

Let $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$ be two different ranking lists for n objects. Pairs of $(x_i, y_i), (x_j, y_j)$ for $i, j = 1, ..., n$ are called concordant if the order of rankings are the same for both of the elements. In mathematical terms, the two pairs are concordant if $x_i > x_j$ and $y_i > y_j$ or if $x_i < x_j$ and $y_i < y_j$. The pairs are called discordant if the order of rankings are not the same for both of the elements. In mathematical terms, the two pairs are discordant if $x_i > x_j$ and $y_i < y_j$ or if $x_i < x_j$ and $y_i > y_j$. If $x_i = x_j$ or $y_i = y_j$ they are neither concordant nor discordant. The Kendal Tau Coefficient is defined as:

$$\tau = \frac{\text{(number of concordant pairs) - (number of discordant pairs)}}{\text{(total pair combinations)}}.$$

The value of τ ranges from -1 to $+1$; τ equals to 1 means perfect agreement between two rankings. τ equals to -1 means the two rankings totally disagree with each other; one ranking is the reverse of the other. Total pair combinations is equal to $n(n-1)/2$. Example: We have the following two ranking lists for five objects.

$$list_1 : 3, 2, 5, 1, 4$$
$$list_2 : 3, 1, 5, 4, 2$$

There are $\frac{5\times4}{2} = 10$ pair combinations. Out of these 10 pairs, 6 are concordant and the rest are discordant. In this case, τ is equal to $+0.2$.

- Discounted Cumulative Gain (DCG) [34]:

Using a graded relevance scale of documents in a search engine result set, DCG measures the usefulness or gain of a document based on its position in the result list.
Let the ratings of n documents be $r_1, r_2, ..., r_n$. The Cumulative Gain (CG) and Discounted Cumulative Gain (DCG) of a document at position p are defined as follows.

$$CG_p = r_1 + r_2 + ... + r_p,$$
$$DCG_p = r_1 + \sum_{i=2}^{p} \frac{r_i}{\log_2^i}.$$

An alternative formulation for DCG is $\sum_{i=1}^{p} \frac{2^{r_i}-1}{r\log_2(1+i)}$ [37]. Nowadays, Normalized-DCG (NDCG) is more common and is computed from normalizing DCG at rank n by DCG of ground truth rankings of documents.
Example: Consider the ground truth ranking of five documents, D_1 to D_5, on $0-3$ ranking scale as in Table A.1. NDCG is computed for an arbitrary ranking function.

Table A.1: Computation of NDCG.

Grand Truth		Ranking Function	
Document Order	Rating	Docuemnt Order	Rating
d_1	3	d_3	2
d_2	2	d_1	3
d_3	2	d_4	1
d_4	1	d_5	0
d_5	0	d_2	2

$$DCG_{gt} = 3 + \frac{2}{\log_2^2} + \frac{2}{\log_2^3} + \frac{1}{\log_2^4} + \frac{0}{\log_2^5} = 6.76,$$
$$DCG_{rf} = 2 + \frac{3}{\log_2^2} + \frac{1}{\log_2^3} + \frac{0}{\log_2^4} + \frac{2}{\log_2^5} = 6.49,$$
$$NDCG_{rf} = \frac{6.49}{6.76} = 0.96.$$

A.2 Evaluation Metrics Used in Regression Predictions

- Kullback-Leibler Divergence [35]:

Kullback-Leibler divergence or relative entropy is a non-symmetric measure of the difference between two probability distributions P and Q. The KL-divergence of Q from P, denoted by $D_{KL}(P||Q)$, is a measure of the information lost when Q is used to approximate P. This evaluation metric is used in [5].

$$D_{KL}(P||Q) = \sum_i P_i \times \log_2 (\frac{P_i}{Q_i}).$$

- Mean Squared Error (MSE) [36]:

It measures the average of the squares of the difference between the estimator and what is estimated. If $\bar{Y}$ is a vector of n predictions and Y is the vector of true values, MSE is calculated using the following formula.

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} (\bar{Y}_i - Y_i)^2.$$

There are also other metrics related to MSE including **MAE** -Mean Absolute Error- and **RMSE** -Root Mean Squared Error.

- Coefficient of Determination (R^2 score):

This measure illustrates how well observed outcomes are replicated by the model, as the proportion of total variation of outcomes explained by the model. In other words, it indicates how well data fits a statistical model. Consider a data set of n observations where y_i and f_i denote the observed value and associated model value (predicted value) for i^{th} observation, respectively. The mean of observed values is $\bar{y} = \frac{1}{n} \times \sum_{i=1}^{n} y_i$. Following is the definition of coefficient of determination. R^2 score of one indicates that model perfectly fits the data.

$$R^2 = 1 - \frac{(y_i - f_i)^2}{(y_i - \bar{y})^2}.$$

Bibliography

[1] R. Breuer, M. Brettel, and A. Engelen, "Incorporating long-term effects in determining the effectiveness of different types of online advertising", *Springer Science + Business Media*, Vol. 22, pp. 327340, 2011.

[2] G. Sanje and I. Senol, "The Importance of Online Behavioral Advertising for Online Retailers", *International Journal of Business and Social Science*, Vol. 3 No. 18, pp. 114-121 , 2012.

[3] H. Haans, N. Raassens, and R. Hout, "Search engine advertisements: The impact of advertising statements on click-through and conversion rates", *Springer Science + Business Media*, Vol.24, pp.151163, 2013.

[4] R. F. Wilson and J. B. Pettijohn, "Search engine optimisation: A primer on keyword strategies", *Journal of Direct, Data and Digital Marketing Practice*, Vol. 8, No. 2, 2006.

[5] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: estimating the click-through rate for new ads", *Proceedings of the 16th International Conference on World Wide Web*, pp. 521-529, 2007.

[6] D. H. Hu, E. W. Xiang, and Q. Yang, "Get more clicks!", *Hong Kong University of Science and Technology*, pp.1-4, 2009.

[7] N. N. Liu and Q. Yang, "Eigenrank: a ranking-oriented approach to collaborative filtering", *SIGIR*, pp. 83-90, 2008.

[8] A. Agarwal, K. Hosannagar, and M. D. Smith, "Location, Location, Location: An analysis of profitability of position in online advertising markets", *Journal of Marketing Research (JMR)*, Vol. 48, No. 6, pp. 1057-1073, 2011.

[9] A. Bulut, "TopicMachine: Conversion prediction in search advertising using latent topic models", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 11, pp. 2846–2858, 2014.

[10] Ch. D. Manning, P. Raghavan, and H. Schutze, "An introduction to information retrieval", *New York, NY, USA: Cambridge Univ. Press*, 2008.

[11] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for IDF", *Journal of Documentation*, Vol. 60, No. 5, pp. 503-520, 2004.

[12] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retreival", *Journal of Documentation*, Vol. 28, pp. 11-21, 1972.

[13] G. K. Zipf, "Human behavior and the principle of least effort", *Addison-Wesley Press*, 1949.

[14] C. E. Shannon, "A mathematical theory of communication", *Journal of Bell System Technical*, Vol. 27, pp. 379-656, 1948.

[15] X. Wei and B. Croft, " LDA-based document models for ad-hoc retrieval", *SIGIR*, 2006.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Drichlet Allocation", *Advances in Neural Information Processing Systems 14*, 2002.

[17] H. Attias, "A variational Bayesian framework for graphical models", *Advances in Neural Information Processing Systems*, Vol. 12, 2000.

[18] D. M. Blei and M.l. Jordan, "Variational inference for Dirichlet process mixtures", *Bayesian Analysis*, Vol. 1, pp. 121-144, 2005.

[19] M. J. Beal, "Variational algorithms for approximate Bayesian inference", *PhD. Thesis, Gatsby Computational Neuroscience Unit, University College London*, 2003.

[20] D. M. Blei and J. D. Lafferty, "Topic Models", *Text Mining: Classification, Clustering, and Applications,* London, U.K.:*Chapman & Hall/CRC Press*, pp. 71-94, 2009.

[21] T. P. Minka and J. Lafferty, "Expectation-propagation for the Generative Aspect Model", *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pp.352-359, 2002.

[22] T. P. Minka, "Expectation propagation for approximate Bayesian inference", *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pp. 362-369, 2001.

[23] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast Collapsed Gibbs Sampling for Latent Dirichlet Allocation", *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 569-577, 2008.

[24] Y. W. Teh, D. Newman, and M. Welling, "A Collapsed Variational Bayesian inference algorithm for Latent Dirichlet Allocation ", *Advances in Neural Information Processing Systems*, Vol. 19, 2007.

[25] T. L. Griffiths and M. Steyvers, "Finding scientific topics", *Proc Natl Acad Sci U S A, 101*, pp.52285235, 2004.

[26] L. Bolelli, S. Ertekin, C. L. Giles, "Topic and trend detection in text collections using Latent Dirichlet Allocation", *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, pp. 776-780, 2009.

[27] L. AlSumait, D. Barbara, and C. Domeniconi, "On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking", *ICDM '08. Eighth IEEE International Conference on Data Mining*, Vol. 8, pp. 3-12, 2008.

[28] Mukherjee and D. M. Blei, "Relative performance guarantees for approximate inference in latent Dirichlet allocation", *D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, Adv. in Neural Inform. Processing Syst.*, Vol. 21, pp, 11291136. 2009.

[29] A. Asuncion, M. Welling, P. Smyth, and Y.W. Teh, "On smoothing and inference for topic models", *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 27-34, 2009.

[30] W. Gilks, S. Richardson, and D. J. Speigelhalter, "Markov chain Monto Carlo in practice", *Chapman & Hall*, 1996.

[31] R. M. Neal, , "Markov chain sampling methods for Dirichlet process mixture models", *Journal of Computational and Graphical Statistics*, Vol. 9, No. 2, pp. 249-265, 2000.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M.Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, Vol. 12, pp. 2825-2830, 2012.

[33] M. G. Kendall, "A new measure of rank correlation", *Biometrika*, Vol. 30, pp. 81-93, 1938.

[34] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of IR techniques", *ACM Transactions on Information Systems*, Vol. 20, No. 4, pp. 422-446, 2002.

[35] S. Kullback and R. A. Leibler, "JSTOR: The Annals of Mathematical Statistics", *The Annals of Mathematical Statistics*, Vol. 22, No. 1, pp. 79-86, 1951.

[36] E. L. Lehmann and G. Casella, "Theory of point estimation", *Springer*, Vol. 41, 1998.

[37] Ch. Burges, T. Shaked, E.Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender,, "Learning to rank using gradient descent", *Proceedings of the 22nd international conference on Machine learning ICML 05*, pp. 89-96, 2005.