Using Sentiment Analysis for Better Placement of Contextual Advertisements

by

Samuel Suraj Bushi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Online advertising is one of the most lucrative forms of advertising, with digital ad-spending expected to reach $83 billion in 2017 [1], making it one of the most important channels of advertising media. Contextual Advertising is a type of online display advertising that takes cues from the content of the triggering page and displays advertisements that are relevant to the current *context*, increasing the probability of an impression conversion, while at the same time avoiding being too annoying or disinteresting to the user. However, on several occasions, the *context* may have a negative connotation, and displaying advertisements that are relevant to it might prove to be detrimental to the advertiser. We refer to such a scenario as an *unfortunate placement*.

In this thesis, we propose APNEA (Ad Positive NEgative Analysis), a light-weight system that extracts the sentiment from the triggering page and associates it with the relevant advertising brands. APNEA uses a sentiment-oriented approach to rank the advertisers based on their associated sentiments such that positively correlated brands are ranked higher than brands that are neutral or negatively correlated, while maintaining the relative order of relevance, thereby avoiding an unfortunate placement.

Experiments show that APNEA helps avoid unfortunate placements while maintaining ad-relevance. It outperforms several baselines in terms of accuracy on human-annotated test data while having a lower run-time, which is crucial for real-time bidding systems.

Acknowledgements

I am eternally grateful to my supervisor Prof. Osmar Zaïane for his invaluable guidance and incredible support both inside and outside the classroom. This thesis would not have been possible without him.

I would also like to thank my parents, my sister and my extended family for always being there for me. Finally, I express my utmost gratitude to my friends in Edmonton, Canada, and all over the world, for their constant support and encouragement.

Contents

List of Tables

List of Figures

List of Acronyms

API	Application Programming Interface
APNEA	Ad Positive NEgative Analysis
CPC	Cost-Per-Click
CPI	Cost-Per-Impression
CPM	Cost-Per-Mille
CTR	Click-through-rate
DASA	Dissatisfaction-oriented Advertising, based on Sentiment Analysis
DOM	Document Object Model
DSP	Demand Side Platform
HTML	Hypertext Markup Language
LODR	Logarithm of Odds Ratio
ML	Machine Learning
POS	Part-Of-Speech
RL	Logarithmic Ratio
RNTN	Recursive Neural Tensor Network
RTB	Real-time Bidding
SD	Sentiment Difference
SM	Sentiment Maximum
SOCA	Sentiment Oriented Contextual Advertising
SSP	Supply Side Platform
SVM	Support Vector Machine
TD	Threshold Difference
TM	Threshold Maximum
URL	Uniform Resource Locator

Chapter 1

Introduction

Online advertising is one of the most lucrative forms of advertising. In 2017 Q1, more than 21 million or 85% of Google's revenue came from Online Advertising operations [2]. Other search engines and web-hosting platforms also leverage the monetary benefits of online advertising to a similar extent.

There are many different forms of online advertising, i.e. text, image, audio, video, email (spam) and so on, each one working in its own different way. According to the authors of [3], out of these different advertising media, textual advertisements occupy "a large part of the market". There are two main channels through which textual advertisements are triggered:

1. *Sponsored Search:* Sponsored search works by using a user's query in a search engine to trigger ads, which are then displayed as a part of the results. This works by understanding the user's information need and matching it with competing advertisers who have a similar advertising agenda. Most of the popular search engines like Google, Bing, Yahoo!, etc. all use sponsored search on their search platforms.

2. *Contextual Advertising:* Contextual Advertising, on the other hand, works based on the content of a web page (also known as a *triggering page*) that a user visits and displays advertisements that are relevant to the *context* of that web page. Several studies have shown that increased relevance indeed improves the click-through-rate of advertisements. [4, 5].

With the rise of search engines in the late 1900s, Sponsored Search has developed before Contextual Advertising (See Chapter 2). Contextual Advertising, therefore, adopted many practices from Sponsored Search, such as characterizing textual ads using bidding phrases [3], which we observe in the current work. However, using bidding key-phrases without any context associated with them can lead to an interesting problem.

1.1 The Problem

Contextual advertising helps by displaying ads that are relevant to the context at hand, thereby increasing the probability of the user clicking the advertisement, while at the same time avoiding being too annoying or disinteresting to the user. However, on several occasions, the context can have a negative connotation, and displaying advertisements that are relevant to this context might cause more harm than good to the advertiser. We refer to such a scenario as an *unfortunate placement* of an advertisement.

For example, in Fig. 1.1, the article describes the account of a couple who receive a package of human body parts that have been wrongly delivered to them, instead of being delivered to a laboratory. Since the article mentions words like `package` and `delivery`, UPS may be bidding on similar words and therefore has won the bid to display its advertisement on this web page. However, this may negatively affect the advertiser (UPS) even though it may not have been involved in this particular incident. Furthermore, the user may be less compelled to click on the advertisement now, because of the negative correlation between the advertisement and the article.

We hypothesize that the root cause of these unfortunate placements is the lack of context associated with the triggering key-phrases. We believe that the sentiment carried by these phrases in the context of the web page will provide valuable information that could be used to mitigate such unfortunate placements. Therefore, we propose to use Sentiment Analysis to extract and analyze the sentiment of these triggering phrases and provide the missing context.

Sentiment Analysis is the "field of study that analyzes people's sentiments

Figure 1.1: An ad for UPS on a news article about a mis-delivered package[1].

towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes" [6]. It is a vast field that encompasses both Opinion Mining and Emotion Mining. Sentiment Analysis has been applied to several domains like healthcare, e-learning, call-centres etc. This work aims at applying Sentiment Analysis to intelligently filter out advertisements that may be harmed if they were matched to a given web page. In other words, given a web-page and a set of advertisements with their bidding key-phrases, our system selects those advertisements that may appear on the web page, without damaging their own brand image.

In the context of text mining, Emotion Mining is "the study of emotions reflected in a piece of text" [7]. It is much more granular than Sentiment Analysis and carries more information. However, we will not be dealing with

[1]Source: Osmar R. Zaïane.

the emotions expressed in a web page in this work but address the same as part of future work.

Modern online advertising architecture is complex and fast. It has a lot of underlying players who play different roles, from publishers who own the advertising inventory to advertisers who seek an audience for their products and services. The entire process of matching a user with a relevant advertisement usually takes place in less than 100 milliseconds [8]. Such strict time constraints make the current task at hand more challenging. In Chapter 2, we describe in detail the different players and the chain of events that take place in Real-time Bidding.

1.2 Thesis Statement

In this work, we hypothesize that we could use the sentiment associated with a bidding phrase to avoid an unfortunate placement, in a reasonable time. Furthermore, we argue that such a system would also preserve existing good matches between positive pages and advertisements. More formally,

Given a web-page P and a set of advertisements ADS, where each advertisement $adv_i \in ADS$ has a list of bidding phrases, $\{p_i^1, p_i^2, ...p_i^j\}$, select relevant advertisements $adv_{win} \subseteq ADS$ such that the sentiment associated with each advertisement $adv_k \in adv_{win}$ is non-negative w.r.t. P.

The challenges that are posed by this problem statement are:

1. The advertisements adv_{win} should be relevant to the context of the page P.

2. The system must be able to piggyback on existing bidding algorithms, without significant additional latency.

3. The system should not compromise good matches, i.e. if an advertisement adv_i is both relevant and positively associated with a web-page P, and adv_i and P are matched together without the Sentiment Analysis component, then even after using sentiment, P and adv_i should be matched by the system.

Through this thesis, we aim at finding answers to the following research questions:

1. Would it be possible to build a system that can avoid an unfortunate placement of an advertisement?

2. Can such a system be fast enough to be practical with respect to modern day advertising technology?

3. Can such a system preserve existing good matches, when the context is non-negative?

To address the above problem and its challenges, we propose the Ad Positive NEgative Analysis (APNEA) system. APNEA is a light-weight system that extracts the sentiment from the triggering page P and associates it with candidate advertisers interested in advertising on P. By providing sentiment as the missing context to the advertisements, the system ensures that unfortunate placements are avoided. Being light-weight and simple, APNEA is fast and can be easily added on top of existing algorithms, without compromising the speed of the bidding process. Finally, advertisers are 'penalized' only by the sentiment context of the web page, therefore a non-negative context should not prevent a relevant advertisement from being displayed, ensuring that existing good matches are not disturbed in APNEA.

Experiments show that APNEA helps avoid unfortunate placements while maintaining ad-relevance. It outperforms several baselines in terms of accuracy on human-annotated test data while having a lower run-time, which is crucial for real-time bidding systems. Through this work, we also contribute two datasets to the scientific community for further research in this specific area: a dataset of advertisements, with bidding key-phrases and a manually annotated test dataset of page-ad matching ground-truth.

The rest of this work is organized as follows: In Chapter 2 we introduce Real-Time Bidding and Sentiment Analysis in brief and discuss related works. In Chapter 3, we give a detailed description of the APNEA system and the different sentiment analyzers used. In Chapter 4, we introduce the datasets

used and evaluate the proposed system against a set of baselines and explore how different features of APNEA affect its performance on the test set and analyze the run-time of the system. Finally, we conclude the current thesis and shed some light on future work in Chapter 5.

Chapter 2

Background

2.1 Online Advertising

Online Advertising is a complex process that involves many players. Before we go into the details of how an advertisement is served online, we define a few basic terms. Most of these definitions are adapted from [8], while a few have been taken from Wikipedia [9, 10, 11, 12].

2.1.1 Definitions

Online Advertising: Online advertising is a form of marketing and advertising which uses the Internet to deliver promotional marketing messages to consumers. As discussed earlier, online advertising can take on many forms ranging from text and images to video and emails.

Display ads: Display advertising is a subset of Online Advertising that deals with advertising on websites. It includes many different formats such as text, images, flash, video, and audio.

Advertising Inventory: A notion of the advertising volume regarded as the virtual assets owned by the publisher. It is also known as Ad space.

Ad Impression: An impression is the unit of ad inventory, i.e. an ad display opportunity.

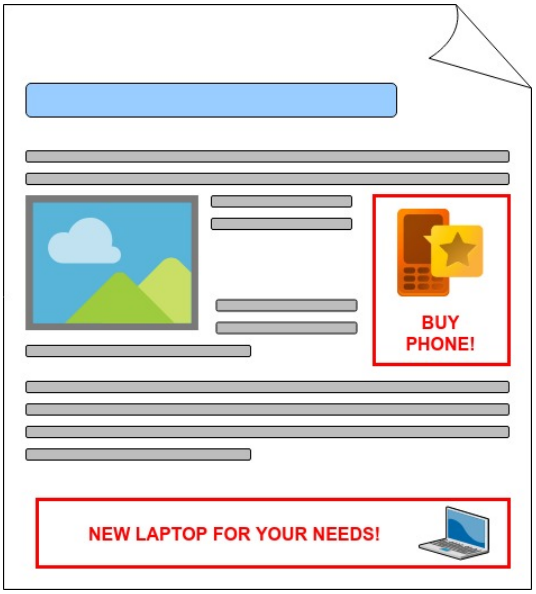

Figure 2.1: Example of Display Ads.

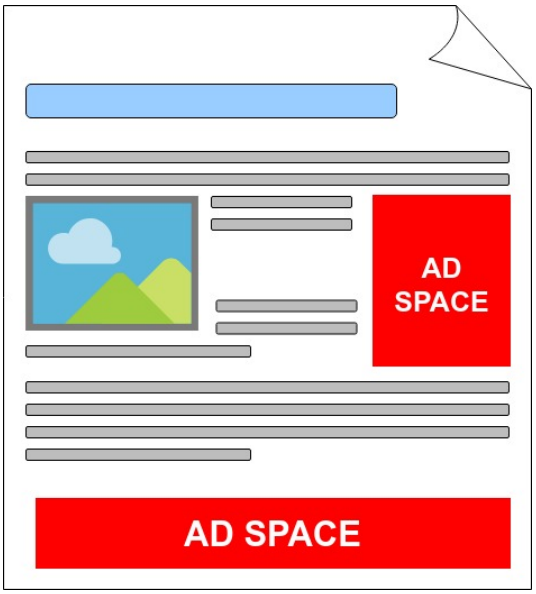

Figure 2.2: Example of Ad Inventory.

Ad Network: An ad network is a company that connects advertisers to websites that want to host advertisements. The key function of an ad network is aggregation of ad space supply from publishers and matching it with advertiser demand.

Ad Exchange: A marketplace which connects the media sellers (publishers) and buyers (advertisers) via network message parsing

with a predefined protocol and selects a buyer for each sold media inventory (ad impression) by auctions.

Supply-side Platform: A supply-side platform (SSP) is a platform which serves publishers to manage the ad inventory of the sites.

Demand-side Platform: A demand-side platform (DSP) is a platform which serves advertisers to manage their campaigns and submits real-time bidding responses for each bid request to the ad exchange via computer algorithms.

Click-through-rate: In online advertising, click-through-rate (also known as CTR), is the ratio of the number of users that have clicked on an advertisement, to the number of users to whom the advertisement has been served. It is used as a metric to measure the effectiveness of an advertisement.

2.1.2 History and Evolution of Online Advertising

The first 'true' banner ad was run by HotWired, on October 24th, 1994 for AT&T. Since then online advertising has grown into a multi-billion industry with huge players [13].

The concept of Sponsored Search was the first of the different technologies to be developed in the domain of Online Advertising owing to the ubiquity of information need and the simplicity in satisfying that need through marketing partners. Sponsored Search was developed by Bill Gross of Idealab in 1998 which eventually was acquired by Yahoo! in 2003 [14]. In 2002, Google started AdWords with Generalized Second Price quality-based bidding and was followed by Yahoo! in 2007 [13]. Sponsored Search allows advertisers to bid on keywords in a user's search query and display their advertisement as part of the results. This satisfied the information need of the user, while at the same time providing audience to the advertisers and generating revenue for the search engine hosting the advertisements [13].

In April 2003, Google bought Oingo, which has already developed a pro-

prietary search algorithm built on word meanings [13]. Other networks like Yahoo! Publish Network and Microsoft adCenter offered a competitive service. Contextual advertising platforms helped publishers to sell the ad inventory on their web pages to these networks for revenue. Sponsored Search can also be thought of contextual advertising, the context being the user query. It has been emphasized more because of the "large market, early development and research attention" [13].

In 2005, ad exchanges that held auctions for impressions were created, like DoubleClick Advertising Exchange, adBrite, etc. These exchanges helped the publishers and the advertisers to bid based on user behavioral profile and reach a larger number of bidders. The process of holding auctions, where the advertisers bid in real-time for each ad inventory is called Real-time Bidding (RTB) [15]. Other platforms that helped facilitate the auctioning and bidding processes like DSPs and SSPs began to be developed shortly after [13].

Real-time bidding poses several challenges owing to the large traffic of impressions being sold every second. DSP Fikisu claims that it processed 32 billion ad impressions daily [16], surpassing the average of 1.2 billion trades on the NYSE trades daily. Therefore, it is safe to say that RTB is larger than the financial market in terms of trading quantity [13]. Since the impressions are bought and sold real-time, this facilitates user behavior targeting and personalization [17]. Furthermore, RTB has become a field of interest for various research groups such as in Information Retrieval to address the problem of ad relevancy [17], Data Mining to mine bidding patterns from a large stream of observed bids [18], and Machine Learning to learn models that optimize the campaign performance by bid estimation through click-through-rate (CTR) prediction [19].

The evolution of Sponsored Search and Contextual Advertising, and later Real-Time Bidding has led to the rise of several revenue models in the online advertisement industry: Cost-Per-Click (CPC), Cost-Per-Mille (CPM) and Cost-Per-Impression (CPI). There are other models such as Cost-Per-Action (CPA) where the target action is not just a mere impression or an ad-click, but the execution of a specific task, like the user buying a product or signing

up to a website, etc., which we will not be discussing here.

- The CPC model charges the advertiser for every time an ad has been clicked.
- The CPI model charges for every time an impression has been made (i.e. an ad has been delivered), irrespective of whether the ad has been clicked or not.
- The CPM model, where *Mille* is Latin for 'thousand', charges the advertiser for every thousand impressions.

Not surprisingly, the mechanism behind each revenue model is different, with CPC models having a lot of proprietary algorithms that rank the advertisements, according to their quality[1]. Real-time bidding uses CPI or CPM models as the inventory is bought and sold in real-time, with user behavioral targeting.

2.1.3 Cost-Per-Click Model for Sponsored Search

The Cost-Per-Click model is more commonly employed by Ad Networks, like Google Ads, Bing Ads, etc. for Sponsored Search, where the conversion of an impression to a click is of higher priority. Therefore, these networks use complex and secretive algorithms to rank ads and determine the prices to be paid per click. Google assigns a 'quality score' for each ad, which depends on a number of factors like ad quality, landing page quality, ad format, etc. and uses it to rank the ads. The basic steps behind a typical bidding process in Sponsored Search are as follows [20, 21]:

1. Each advertiser places a max-bid for a particular advertisement in a campaign. The Ad rank of an advertisement is determined as a combination of the quality score and the max-bid of the advertiser, say by taking the product of the two:

$$AdRank_i = QualityScore_i * MaxBid_i$$

[1] https://adwords.googleblog.com/2013/10/improving-ad-rank.html

Figure 2.3: Calculating the Actual CPC. [2]

2. The advertisements are displayed in decreasing order of ad rank, with the CPC for each advertiser A_i determined by the formula:

$$ActualCPC_i = AdRank_{i+1}/QualityScore_i + \$0.01$$

where $AdRank_{i+1}$ is the AdRank of the advertiser below the i^{th} advertiser and $QualityScore_i$ the quality score of the i^{th} advertiser.

Since the ad rank of entities below the i^{th} advertiser would be less than $AdRank_i$, while a better $QualityScore_i$ will reduce the Actual CPC, the above formula has the effect of higher quality advertisements being placed higher up and costing less, encouraging the advertisers to improve their ad-quality.

2.1.4 Cost-Per-Impression Model for Real-Time Bidding

As mentioned earlier, there are many players in Real-Time Bidding and the process is quite complex. Nevertheless, the simplified steps in delivering an advertisement to a user are illustrated in Fig. 2.4, and are as follows [13, 22, 23]:

1. When a browser loads a web-page with ad inventory, the browser requests the URL associated with the ad space for an advertisement to be

[2]Source: https://searchengineland.com/new-adwords-ad-ranking-formula-what-does-it-mean-174946

displayed.

2. This request goes to the web-page publisher's ad server, which then decides whether this ad space could be served to some of its premium customers, who buy the inventory directly from the publisher.

3. If no such premium buyers are available, the publisher ad server passes the request to a Supply Side Platform (SSP) which further processes the request, performing checks like whether the SSP has seen this consumer before, or if the SSP does not have adequate information it may even request a Data Provider to get to know more about the consumer.

4. Once the SSP has determined that it has enough information, it forwards the request to an ad exchange, which has been busy connecting with other SSPs, DSPs and ad exchanges.

5. The Ad exchange may have a pre-cached bid, which is a bid by an advertiser to buy a certain number of impressions if the consumer meets certain criterion. If there are no pre-cached bids, then the ad exchange conducts an auction where its connected DSPs can participate.

6. The DSPs place bids, depending on which advertisers are associated with them. The Ad exchange picks the winning bid, usually by second price auctioning and sends it back up the hierarchy, through the SSP to the publisher and finally to the browser.

7. The browser then uses this information to contact an Ad Server, which marks the request of the advertisement as one impression (and charges for it) and sends the necessary data, images, etc. to the browser which are finally displayed to the consumer.

DSPs process all the information regarding the user and the current URL and decide whether to bid or not for the current ad space. These decisions are taken through statistical analysis of past bids, and with the help of complex artificial intelligence systems [13]. More precisely, DSPs decide whether to bid

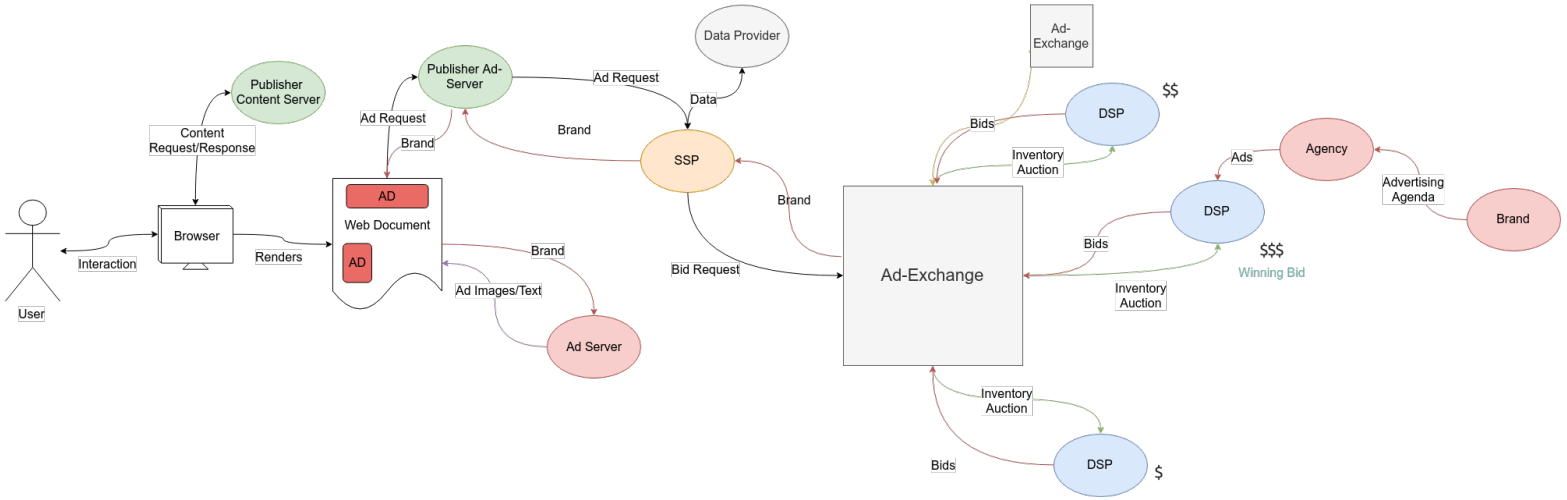

Figure 2.4: Different Agents involved in Real-Time Bidding

on a particular ad inventory based on the predicted click-through-rate of the advertisement [13, 24, 25]. However, we do not have the data to build such a system. Several studies have shown that ad-relevance increases click-through-rate in advertisements [4, 5]. Therefore, we opt to use the relevance of an advertisement to the triggering page as a proxy for the click-through-rate and decide whether an advertiser would bid on the triggering page.

This entire process of real-time bidding takes place within 100 milliseconds. Therefore, any proposal must be lightweight and fast enough to piggyback on existing bidding algorithms, without compromising their speed.

2.2 Sentiment Analysis

Sentiment Analysis is a well-researched topic in data mining. Since the advent of the internet, ever-increasing amounts of data is publicly available. Mining useful information such as the sentiment of people from this data has become an important problem and has been used for many applications. Research in Sentiment Analysis and Emotion Mining has been progressing at a rapid pace, and its applications are becoming more and more diverse. In this section, we will define Sentiment Analysis, and discuss the current research practices in this community.

2.2.1 Definitions

The Cambridge Dictionary defines *Sentiment*[3] as 'a thought, opinion, or idea based on a feeling about a situation, or a way of thinking about something', and *Opinion*[4] as 'the thoughts or beliefs that a group of people have'. Yadollahi et al. [7] aptly redefine Sentiment as "an opinion or idea colored by an emotion". Under this definition, Sentiment Analysis encompasses both Opinion Mining and Emotion Mining. Sentiments can be either *positive*, *negative* or *neutral*.

Liu [6] defines Sentiment Analysis as "the field of study that analyzes people's sentiments towards entities such as products, services, organizations,

[3]https://dictionary.cambridge.org/dictionary/english/sentiment
[4]https://dictionary.cambridge.org/dictionary/english/opinion

individuals, issues, events, topics, and their attributes". Opinion Mining is the field of study that aims at extracting the opinion carried by an individual on a particular topic. For a more in-depth discussion on the differences between sentiments, opinions, emotions, and subjectivity, the readers are directed to [7, 6].

There is no common consensus on the definitions and the relations between the fields of Sentiment Analysis and Opinion Mining. Yadollahi et al. [7] describe Sentiment Analysis as the field that encompasses Emotion Mining and Opinion Mining, whereas Liu [6] uses the terms interchangeably. In this thesis, we would use the above definitions when referring to these fields, adhering to the hierarchy described in [7].

Sentiment Analysis has been extensively applied to various applications, ranging from gathering user sentiment about movies [26, 27], restaurants [28, 29] and products [30, 31] to predicting box office revenues [32], helping detect mental disorders from text [33], improving the user experience for students in e-learning environments [34] and even contextual advertising [35, 36].

There can be two distinct tasks within Sentiment Analysis.

- **Sentiment Detection:** This task deals with identifying if sentiment is expressed by a document. This is a binary classification problem.
- **Sentiment Identification:** This task deals with identifying whether a sentiment associated with a document is a *positive* or *negative* sentiment. This too is a binary classification problem. In a few cases, an additional *neutral* dimension is added to this analysis, which then makes it a ternary classification problem.

Although it is important to know the distinction between these two tasks, it is common practice to combine them into the same mathematical model for the sake of simplicity. Also note that, in the above definitions, a document may be a user review, a web page, an essay written by a student, etc. which will be dependent on the corresponding application.

2.2.2 Levels of Analysis

According to Liu [6], there are three main levels of Sentiment Analysis:

- *Document Level Analysis*, which aims to extract the sentiment carried or the opinion expressed by an entire document. The most wide-spread and well-known application is the analysis of user-reviews on online e-Commerce platforms. In this case, a document is a user-written review regarding a particular product or service. The main assumption behind document-level analysis is that the entire document conveys the same sentiment towards the object of interest.

- *Sentence Level Analysis* aims at extracting sentiment from sentences. Sentence-level analysis provides one more level of granularity than document level analysis. By analyzing each sentence independently, we can account for varying sentiment across different sentences and finally use an augmentation mechanism to consolidate all the extracted sentiments. However, a sentence may convey different sentiments towards different aspects of an object. For example, the sentence "I like the softness of this mattress, but it is too expensive." conveys a positive sentiment towards the softness of the mattress, but a negative sentiment towards its price.

- *Entity level Sentiment Analysis* aims at extracting sentiment-target pairs from a document. This is the most fine-grained level of analysis, that helps to identify the target of a particular sentiment. This helps to alleviate the problems in Document-level and Sentence-level analysis. The targets in this level of analysis can be different objects of interest, for example in a document reviewing different models of a car, the targets would be the different models and the sentiment would be the sentiment expressed towards each model. In other cases, the targets could be different aspects of the same object, like the mattress in the previous example. This helps us to automatically extract the different "pros and cons" of the object of interest.

In conclusion, each level of analysis has its own shortcomings: Document

Level Analysis is much more generic and therefore does not give the same granularity as sentence or entity level analysis. Entity level analysis, on the other hand, is highly granular and aims at extracting sentiment target pairs. Although this can be immensely useful, it is usually too complex for most real-world applications. Sentence level analysis is an intermediate compromise to document level and entity level analysis.

In addition to the different levels of Sentiment Analysis, there are many approaches for extracting sentiment, ranging from sentiment lexicons [34, 32] and Support Vector Machines (SVMs) [35, 37] to Deep Learning Models [38, 39].

2.2.3 Lexicon-based Sentiment Analysis

A simple yet effective way to extract sentiment from a document is to maintain a sentiment lexicon and use it to identify the sentiment carried by different words. Any word that is commonly used to express some sentiment, is known as a sentiment word [6]. E.g.: *good*, *bad*, *poor*, *excellent*, etc. A sentiment lexicon, in its essence, is a list of sentiment words, with their associated sentiments; *positive* or *negative*. Typically, words that do not fall into either category are considered to carry a *neutral* sentiment. The main advantage of using Lexicon-based methods is that there is no longer a need for a large annotated dataset. Annotated datasets take up a lot of time and human effort, especially in domains outside of user reviews, where the availability of such datasets is limited [40]. Besides, Lexicon-based methods are easier to use and expand.

Although sentiment lexicons serve as strong baselines, they are far from sufficient to effectively handle the task of sentiment extraction and identification. Lexicons cannot take into consideration the context of the sentence, as they do not delve into the semantics of the sentence. Lexicon-based methods suffer from low recall due to their dependence on the presence of sentiment words in the input document [41]. Also, not all sentences that contain sentiment words carry sentiment. Therefore, there is a need to go beyond lexicon-based approaches.

2.2.4 Machine Learning based Sentiment Analysis

Machine Learning (ML) models like Bayes Networks [42, 43], SVMs [35, 44, 32, 37, 45], Decision Trees and Random Forests[35, 44, 32, 37], Hidden Markov Models [43], Topic Modelling based approaches [46], Ensemble methods [44], log-linear classifiers [33], etc. have been used in various applications of Sentiment Analysis.

In Machine Learning, a feature is a (numeric) characteristic of an observation that is used by an ML model to make decisions [47]. Most machine learning based models depend upon numerical features as inputs. There are several ways to process text into numerical features, especially in the field of Sentiment Analysis. Most applications use unigram features of the input text as the feature set, where the feature vector indicates the presence or absence of a word in the input. A slight modification of the same is the term frequency feature set, where the frequency of the corresponding terms is recorded. Some approaches use only sentiment words, taken from a pre-defined lexicon, as the terms of interest, whereas others may choose to add other features, such as Part-Of-Speech (POS) tags, negations and domain-specific features [48]. Pre-trained word embeddings, which are real-valued vector representations of words, are also very popular in converting textual data into numerical vectors [49, 50, 51]. Notable word embeddings that are freely available are word2vec [52] and GloVe [53].

ML-based techniques are much more robust than Lexicon-based methods and help alleviate the low recall problem of lexicon-based approaches. However, some approaches like Deep Learning need a lot of data to converge. Besides, in order to expand the vocabulary of most ML-based models, the entire model has to be retrained. Nevertheless, we train an SVM model on the UCI Sentiment Analysis dataset [54] and explore how this model fares against the Lexicon-based approach. In Machine Learning, Support Vector Machines (SVMs) are a class of supervised Machine Learning Models that, when employed to classification problems, try to learn a hyperplane dividing the observations into different classes, while maintaining a high margin of sep-

aration [55]. Note that this is a cross-domain application of the SVM model as the UCI dataset consists of user reviews from Amazon, Yelp, and the Internet Movie DataBase (IMDB), whereas we will be evaluating the model on news articles. We chose the UCI dataset, owing to its ease of availability and dataset simplicity.

Sarcastic sentences are always tricky to handle, even with more complex machine learning models. Implicit sentences may not contain any sentiment words but may carry some sentiment or opinion. For example, sentences such as *"What a great car! It stopped working in two days."* or *"This washer uses a lot of water."* need a much more comprehensive understanding of the real world to determine the sentiment that they carry [6]. Research on detecting sarcasm is a hot topic in Sentiment Analysis [56, 57].

In this project, we explore both lexicon-based and ML-based approaches to Sentiment Analysis. Section 3.2.4 covers the various Lexicons and Machine Learning models that we incorporate into the APNEA system.

2.3 Sentiment Analysis in Advertising

To the best of our knowledge, only two works have tried to incorporate Sentiment Analysis into the online advertising setting. In this section, we briefly introduce the key ideas behind these two works.

2.3.1 Dissatisfaction-oriented Advertising, based on Sentiment Analysis

Dissatisfaction-oriented Advertising, based on Sentiment Analysis (DASA) [58, 36], aims to use Sentiment Analysis to extract user sentiment on a particular product or aspect of a product, and intelligently place a rival company that is better at the said aspect, not only correlating with the sentiment of the document, but also increasing revenue for the publishers and providing positive publicity to the advertisers. To the best of our knowledge, DASA is the first work that applies sentiment to the domain of contextual advertising.

For example, if document D is complaining about the *safety* aspect of

an automobile brand H; in the traditional scenario, since the document D's topic closely matches that of the advertisements from brand H, it is highly likely that this document will get matched with the advertisements from H, however, similar to the case of *unfortunate placements*, this may be undesirable for both the publisher and the advertiser. DASA aims to avoid such a scenario by instead matching D with another automobile brand which is known to have a good record with respect to the aspect *safety*.

The authors of DASA, [58] divide this task into two sub-tasks, namely: topic extraction and advertising keyword extraction. This advertising keyword would then be used to match the document with an appropriate advertiser. For topic extraction, DASA uses a rule-based keyword extraction approach on the input document to extract triggering keywords. First, the authors of [58] extract opinionated sentences from the document by using the Harvard General Inquirer lexicon. They devise a set of 7 rules, that can be applied to these opinionated sentences. Then, the rules are ordered by priority, in order to handle sentences that fall under multiple rules. The topic words can be extracted from the templates of each of the rules, where each topic word is associated with the same sentiment as the opinionated sentence from which it is extracted. From the extracted topic words, only those words that have a negative sentiment associated with them are considered as advertising keywords.

Experimental setup for DASA [58] consists of evaluating both sub-tasks of the problem. Topic word extraction is evaluated against baselines such as Nearest-Noun, Random-Noun, and Random-Word, and it outperforms all baselines, with an accuracy of 55%, on ground truth data manually annotated by humans. The advertising keywords are also manually annotated by annotators, and the ad keyword extraction is compared against top-10 and top-3 *tf-idf* keywords. The rule-based approach proposed by DASA has the best commercial value of 26.5%, which was determined based on whether the extracted words triggered advertisements on a search engine. It also had the best Recall (37.5%) and F1-Score (15.9%), and Precision (10.1%) second only to the top-3 *tf-idf* (10.4%). For more details, the readers are directed towards

[58, 36].

There are a few advantages to the approach that DASA takes. DASA's rule-based approach makes execution faster, ideal for real-time systems. It tackles only negative cases, thereby simplifying the problem. However, the input documents might carry a negative connotation on the entire subject rather than an aspect (or *feature*) of the subject, which makes even a rival choice a 'bad' advertisement. Finally, the underlying motive of DASA stipulates that advertising data should be in a format that facilitates querying for rival brands by features, which makes it impractical to integrate into existing systems. Because of these reasons, along with the fact that DASA evaluates exclusively on negative advertising keywords, we decided not to compare our method against DASA.

2.3.2 Sentiment Oriented Contextual Advertising

Fan and Chang [35] propose Sentiment Oriented Contextual Advertising. The SOCA system is closely related to the current work because the authors of [35] also tackle the problem of *unfortunate placements*, primarily in relation to the blogosphere. Furthermore, to the best of our knowledge, Fan and Chang are the first ones to address this problem without alterations to the advertisement data, as opposed to DASA. For this work, the authors consider only textual advertisements which have a title, a description, and a landing page.

SOCA works at a sentence-level, identifying the sentiment of each sentence and removing those sentences that have a negative sentiment, from the page-ad matching step. The authors, for the sake of 'efficiency', divide the task of sentiment extraction into two sub-tasks: Sentiment identification and Sentiment extraction. The first sub-task, sentiment identification deals with identifying the sentences which express some sentiment, a binary classification problem. The second sub-task, sentiment extraction deals with identifying the sentiment expressed by these opinionated sentences. The authors propose to use SVMs and Decision Trees to detect the opinionated sentences, with unigram and opinion-words as features. For the sentiment extraction subtask, they additionally consider, two linear models, which consider the sign and the

strength of the opinion words respectively. As mentioned earlier, once the sentiments of all the sentences in the document are extracted, only the positive (neutral) sentences are considered for page-ad matching.

The authors explain that considering only the terms in the advertisements or the triggering document, might not be sufficient to get a page-ad match, therefore they propose the use of term expansion on both the document and the advertisements to increase the chance of intersection between relevant pages. For the term expansion, the authors first extract a set of seed terms, based on a number of criteria like position in the document (in the title, or anchor text), frequency, POS tag, etc. These seed terms go through three stages of term expansion. The first stage is to use an online lexical database for English, like WordNet[5] to extract the synonyms of the seed terms and include them in the extended terms. For example, the synonyms for *car* include *automobile*, which might be a good triggering word for automobile companies or insurance companies. All seed terms may not have synonyms in the thesaurus, therefore such terms move on to the second stage of term expansion where the Wikipedia article corresponding to these terms is fetched and the top-5 most frequently occurring nouns in the article are added to the extended terms. For example, if the triggering document is talking about the company *Nokia*, *Nokia* will not have any synonyms in WordNet, but the Wikipedia page of Nokia can be used to extract the frequent terms like *mobile*, *communication*, *network*, etc., which help to match it with the appropriate context. Finally, a web-based term expansion technique is used on the seed terms that originate from the title and most frequent words in the document. A search engine is used to query for these 'topic' words, and the top-3 documents are fetched and the LODR metric [59] is used to determine the most frequently co-occurring words with these topic words, which are then included in the expanded terms if the LODR is greater than a threshold δ.

The last part of the SOCA framework is page-ad matching. This part is formulated as an information-retrieval problem, where given a query, which is the triggering document, a set of documents (which are the advertisements, in

[5]https://wordnet.princeton.edu/

this case) are retrieved, ranked by their relevance to the said query. The ads and the document are both represented by *tf-idf* vectors, over the extended vocabulary. A relevance score is calculated for every page-ad pair, by combining two forms of similarity: cosine similarity and ontological similarity. Cosine similarity is calculated between an ad-vector and document vector. Ontological similarity is calculated from the WordNet relations between synsets, *homonymy*, and *hypernymy*, between any two nouns in the document and the advertisement under consideration [35]. The relevance score is then calculated as the linear combination of the two, weighted by a hyper-parameter β:

$$Score(a, p) = \beta * Sim_{cos}(a, p) + (1 - \beta) * Sim_{Onto}(a, p)$$

Finally, the authors evaluate the framework by evaluating the sentiment extraction sub-task and the page-ad matching sub-task. For the sentiment extraction evaluation, the authors use *epinions.com* to train their classifiers. They report in their findings that the SVM model with unigram features has the best Precision, Recall, and F-Score. For the page-ad matching subtask, the authors use human annotators to annotate 150 web-pages and 30 advertisements and test their system against the simpler Contextual Advertising framework (without sentiment) and Google Adwords. Their reports suggest that their system outperforms the other two with 68.2% accuracy on the dataset. Finally, they use Precision-Recall curves, Precision@K and Mean Average Precision to demonstrate that the use of both Ontological and Cosine similarity is superior to using either one independently. For more details about the framework and the evaluation, we direct the readers to [35].

As mentioned earlier, Fan and Chang are the first to address the problem of unfortunate placements — as it applies to our work. However, there are two main issues with their approach. Firstly, removing negative sentences might not solve the problem completely, as the remaining sentences might still trigger an unfortunate placement. Secondly, Fan and Chang fail to do a run-time analysis of the system, which is crucial for real-time systems. Nevertheless, this is the closest work to our current thesis and therefore we compare our approach against SOCA, both in terms of speed and accuracy.

Chapter 3

Methodology

In this chapter, we elaborate on the architecture behind our APNEA system and provide details regarding the Sentiment Analysis models that we incorporate into the proposed system. We also describe how the architecture enables APNEA to meet the low latency standards of the modern advertising mechanisms. The APNEA pipeline consists of three significant stages, i.e. Advertisement Pre-processing, Sentiment Extraction and Page-Ad matching. Before we delve into the details of each stage, we would like to redefine the Thesis Statement presented in Section 1.2:

Given a web page P and a set of advertisements ADS, where each advertisement $adv_i \in ADS$ has a list of bidding phrases, $\{p_i^1, p_i^2, ...p_i^j\}$ with auxiliary information, select relevant advertisements $adv_{win} \subseteq ADS$ such that the sentiment associated with each advertisement $adv_k \in adv_{win}$ is non-negative, with respect to P.

Note that we have added auxiliary information to the advertisement data in the above definition. We will shortly discuss what the auxiliary information comprises of, and its usefulness in an intelligent advertising system like APNEA.

3.1 Advertisement Pre-processing

Unlike Fan and Chang, [35], the textual advertisements considered for our system are just the bid phrases for each advertiser, with optional auxiliary information to provide flexibility to the advertisers. We believe that keyword-

based advertising is much more generic than textual advertisements with a title and description, enabling the easy extension of our system to other display advertising domains. Besides, we retrieve our bidding keywords data from Google Adwords [60] dataset, which is geared towards Sponsored Search. Therefore, we curated the list of companies in the dataset and their associated keywords to best suit the task of Contextual Advertising. We make the code for this work, along with the data used, available at our repository[1]. We should note at this point that the advertisement data used throughout this work has only one advertisement per advertiser, whereas in the real world, the same advertiser may have different advertisements corresponding to different campaign agendas. Therefore, we use the terms *advertiser* and *advertisement* interchangeably throughout the rest of the work.

The auxiliary information gives advertisers additional control on whether they care about the sentiment context of the bidding phrases, and the importance of the phrases to the advertisers' campaign. More precisely, each advertiser adv_i has a list of 3-tuples, (p_i^j, s_i^j, w_i^j), where p_i^j is the j^{th} bidding phrase, s_i^j is a Boolean flag, denoting whether the advertiser is sentiment-agnostic with respect to the current bidding phrase and w_i^j is a real number denoting the (commercial or semantic) importance of the bidding phrase to the advertiser. By default, all bidding phrases are sentiment-sensitive and carry a weight of 1.0.

Traditional systems only care about the bidding phrases, but we found the need to incorporate this additional information to give advertisers more control over their ad placement. The s flag in a bidding 3-tuple enables advertisers to disregard the sentiment and bid for a phrase regardless. This can be important for advertisers who bid for phrases that are generally associated with a negative sentiment, or phrases that are important regardless of the context in which they appear. One such example in Listing 3.1, is a law firm, called *A-Lawyers* who specialize in automobile accident cases. Since the firm is interested in keywords that relate to accidents, they would not benefit from a system that avoids placing their advertisements, since the sentiment associated with these

[1] `https://github.com/blumonkey/apnea`

keywords is usually negative.

On a similar note, we also incorporate a term-expansion mechanism, to ensure that even if the bidding phrases are not present verbatim, the expanded terms help capture the semantics of the keywords and relevant advertisers get matched with relevant content. However, we believe that there may be significant noise in the expanded terms, which might disrupt the page-ad matching mechanism, therefore we would like to discount the effect the expanded terms have on the final relevance score. This is achieved by reducing the weight associated with the expanded terms, by a reduction factor r, which is a real number in the range $[1, \infty)$ and is a hyper-parameter of the system.

A sample of the advertisers and their bidding phrases along with the auxiliary information is presented below:

```
{
...
        "hp": [
                ...
                ("printers",  false, 1),
                ("toner␣and␣cartridges",  false, 1),
                ("servers␣for␣businesses",  false, 1),
                ...
        ],
        "Spain␣Tourism": [
                ...
                ("spain",  false, 1),
                ("tourism",  false, 1),
                ("vacation",  false, 1),
                ...
        ],
        "A-Lawyers": [
        ...
        ("crash",  true, 1),
        ("accident",  true, 1)
        ...
        ],
...
}
```

Listing 3.1: Sample of Advertiser Data

More formally,

$$ADS = \{(k, v) | k \in K \wedge v \subseteq V\}$$

where

$$K = \{adv_i \, \forall \, i = 1, 2, 3...N\}$$

and

$$V = \{(p_i^j,\, s_i^j,\, w_i^j) \,|\, p_i^j \in \Sigma^* \wedge s_i^j \in \{\top, \bot\} \wedge w_i^j \in \mathbb{R}\}$$

Here, Σ^* denotes the set of all strings and N is the total number of advertisers in the database. For the sake of simplicity, we assume that all bid phrases are equally important for each advertiser and have a weight of 1.0, but also note that it is easy to extend our model to accommodate this additional information when available.

3.1.1 Constructing the Ad-Vectors and Bid Keyword Set

In order to determine the ad-relevance with respect to a given web page, we represent each advertiser as an $n-$dimensional ad-vector. We also use a dictionary that maps from a bidding keyword to an advertiser, to quickly identify advertisers who are bidding on a token.

In computational linguistics, a vector space model is an algebraic model where a textual entity is represented as a real-valued vector in an n-dimensional space. The number of dimensions depends on how the space is modeled. Each dimension may correspond to a unique token in the vocabulary, or a token from a set of desired tokens (such as sentiment words from a lexicon), or an arbitrary feature corresponding to latent characteristics of the entity. The vectors can be used as inputs to other systems such as Machine Learning models etc., or can be used to query quantities such as the similarity between two entities etc. In this work, we represent both the advertisements and the triggering document as unigram vectors, under this model.

The ad-vectors are constructed as follows: Each of the bidding phrases of an advertiser adv_i is pre-processed, with steps including lemmatization,

converting to lowercase, and stop-word removal. The weights of the resulting tokens are then cumulatively added to the corresponding dimension in the vector space of the ad-vector, with a token that is derived from a bidding phrase p_i^j, contributing a weight of w_i^j. If term-expansion is enabled, the bidding phrase p_i^j is also queried on Wikipedia and if the corresponding article is not a disambiguation page, then the top 15 most frequent words (excluding stop words) in the article are extracted from which, up to 5 nouns are added as expanded terms to the ad-vector, each contributing a weight corresponding to w_i^j/r where r is the reduction factor hyper-parameter.

The `keyword_to_advertiser` mapping is also constructed in a similar manner: the pre-processed bidding phrases are broken down into bidding keywords which are then inserted as keys into a map. The corresponding value for a given key in the map is a set of advertisers that are interested in that bidding keyword. If term expansion is enabled, the expanded terms that are extracted for the construction of the ad-vectors are also included in this dictionary.

Finally, for each advertiser, we keep track of the s-flag for the different bidding keywords. There are a few assumptions that we make at this step: Firstly, an advertiser is always sentiment-sensitive towards their own brand. Secondly, any expanded terms that are derived from a bidding phrase p_i^j will share the same s-flag, s_i^j. Thirdly, if a bidding keyword or expanded keyword has two different s-flags in different 3-tuples, then the `true` flag will have priority over the `false` flag. This is to ensure that the advertiser's preference for sentiment-insensitivity prevails. Algorithm 1 below presents a brief outline of how the individual data structures are constructed.

Note that the loop defined in Line 9 iterates over the keywords. This is a simplification over using the key-phrases, as the phrases can be of arbitrary length, making the task of finding bidding key-phrases in a chunk of text overly complicated. The function `pre_process` lemmatizes the text, converts to lowercase and removes the stop words. The resulting mapping simplifies the process of finding bidding keywords while only compromising on the order of the words. Targeting individual words also allows us to fuzzy-match interesting keywords.

Input : An Advertiser-Keywords mapping, ADS
Output: Keyword-Advertisers mapping, keyword_to_advertiser
Output: Ad-Vectors, ad_vectors
Output: Sentiment-Sensitivity Mapping, sent_sensitivity

Let keyword_to_advertiser be a mapping from keywords to advertisers.
Let ad_vectors be an empty collection
for *every advertiser* a_i *in* ADS **do**
    Let ad-vector $\vec{a_i} \leftarrow \vec{0}$
    for *every bidding triplet* $(p_i^j,\ s_i^j,\ w_i^j)$ **do**
        e $\leftarrow$ `expand` (p_i^j)
        $t \leftarrow$ `pre_process` (p_i^j)
        keywords $\leftarrow$ e $\cup\ t$
        for *token in* keywords **do**
            If *token* is from t, add w_i^j to the dimension corresponding to *token* in $\vec{a_i}$; else add w_i^j/r
            $\text{sent\_sensitivity}_{a_i}(token) = s_i^j$
            keyword_to_advertiser $(token) \cup= a_i$
        end
    end
    Add $\vec{a_i}$ to ad_vectors
end
return keyword_to_advertiser, sent_sensitivity, ad_vectors

Algorithm 1: Advertisement Pre-Processing Step, with term expansion.

3.2 Sentiment Extraction from Triggering Page

In this section, we go into the details of the second component of the APNEA system – extracting the sentiments expressed in the triggering page.

3.2.1 Page Contents: Fetching and Cleaning

The first step in this phase is to fetch the contents of the web page and clear unnecessary elements in the Document Object Model (DOM) tree. To simplify this task, we opted to go for a ready-to-use API[2], which removes the clutter from the HTML page and returns the content and the title. The content may still have HTML elements therefore we use an HTML parser such as `BeautifulSoup`[3] to extract the text.

We define a *chunk* as a singular piece of text that is evaluated for sentiment. We can tokenize the input document into chunks at different levels, which we will discuss briefly.

3.2.2 Document Tokenization

Section 2.2.2 explains the different levels of Sentiment Analysis and the pros and cons of each. In order to demonstrate the effect of each level of analysis on the performance of the proposed system, we use a Document Tokenizer to divide the cleaned document into chunks. More specifically, we choose three different tokenization approaches: Document Level, Sentence Level and Sentence N-gram Level.

- **Document Level Tokenization:** In this approach, we treat the entire document as a single chunk and extract its sentiment. Therefore, all the bidding keywords that appear in this document, have the same sentiment. This approach is much more coarse-grained than sentence level analysis, as the same sentiment is associated with all the triggering keywords in the document.

[2] `https://mercury.postlight.com/web-parser/`
[3] `https://www.crummy.com/software/BeautifulSoup/`

- **Sentence Level Tokenization:** This approach treats each sentence of the document as a chunk, extracts its sentiment and associates it with the bidding keywords that appear in this chunk. Compared to Document Level Tokenization, this approach provides a much more accurate reflection of the extracted sentiment on the bidding keywords.

- **Sentence N-gram Tokenization:** In computational linguistics, an n-gram, in a text, is a list of consecutive n-tokens. When n is 1, they are called *unigrams*. For example, given the sentence: "I liked the movie *Aliens*.", the unigrams in this text would be: ['I', 'liked', 'the', 'movie', '*Aliens*.']. The bigrams (when n is 2) would be: ['I liked', 'liked the', 'the movie', 'movie *Aliens*.']. In this work, we extend the same concept to sentences.

  Sentence Level Tokenization may still run into the problem of co-reference resolution, due to the presence of pronouns. Inability to associate a pronoun with its target noun may lead to lost or inaccurate associations. Owing to the time constraints on our system, we cannot afford to use a co-reference resolution mechanism [61] on our document, therefore we opt for an intermediate approach where we treat each sentence n-gram in the document as a chunk and associate the extracted sentiment with the bidding keywords in the chunk. This is under the assumption that the pronouns are Personal pronouns[4] and reference targets that are local to the n-gram. The value of n is taken as a hyperparameter for the system.

We hypothesize that the attention of the user varies across the content of the web page. In other words, parts of a web page such as the title and the first few paragraphs grab more attention compared to an intermediate paragraph. Therefore, we wish to accommodate this information in our model by associating weights to each subpart, such as a paragraph of the web page, where larger weights correspond to greater importance. For the sake of simplicity, we assign a weight of *2.0* to the first chunk and equal weights of *1.0* to the

[4] `https://en.wikipedia.org/wiki/Pronoun#Personal`

rest of the web page. We wish to experiment with different distributions of these 'attention weights' in our future work.

3.2.3 Pre-processing

Many natural language systems require some level of pre-processing on the input text to avoid stop words, token duplication and to improve the model performance. In this section, we shall discuss the different pre-processing steps employed for the Sentiment Analyzers discussed in Section 3.2.4.

1. **Lemmatization:** First, we use a lemmatizer to convert the tokens in the text to lemmas. Lemmatization refers to reducing the inflectional forms and sometimes derivationally related forms of a word to a common base form [62]. For example, words like *'eating'* and *'eat'* are both reduced to *'eat'*. This step helps the model to treat different forms of the same word in the same manner. This is especially useful in Lexicon-based approaches where all forms of a lemma may not have an entry in the lexicon. In this project, we opted to go for Stanford CoreNLP Lemmatizer [63], which runs as a RESTful API locally and has minimal overhead.

2. **Text Tokenization:** Next, we tokenize the text into individual words. We use the Natural Language Toolkit (NLTK)[5] in Python for text processing.

3. **Uniform Case:** The tokens are converted to lower case to prevent a mis-match between tokens that represent the same word, for example: *'Phone'* and *'phone'* both correspond to the same entity and therefore need not be treated separately. However, some words like *'SAT'* and *'sat'* might not be related, the former referring to the standardized test in the United States, whereas the latter is the past-tense of *'sit'*. Therefore, by converting them to lower-case, we lose the distinction between the two tokens. For the sake of simplicity, we simply ignore such cases for now.

[5] `https://www.nltk.org/`

4. **Removing punctuation marks:** In this step, we remove any punctuation marks from the tokens, as they do not contribute to the sentiment of the text. The punctuations considered by us are: `.,;!?`.

5. **Expanding tokens:** Some tokens may be in a condensed form like *n't* for *'not'*, *'d* for *'would'* and *'s* for *'is'*. These expansions might prove to be important, as they may change the semantics of the text, especially words like *'not'*, which may invert an existing sentiment.

6. **Remove stop-words:** Finally, we remove stop words from the tokens. The stop words are retrieved from the `nltk.corpus` module and edited to retain some words like *'no'*, *'never'*, *'not'*, etc. that may modify sentiment. The exact list of stop words is available on our project repository.

7. **Joining negative tokens:** For lexicon-based approaches, we have an additional step where tokens that follow negative tokens such as *'not'*, *'never'*, *'no'*, etc. are combined to form a new token such as '*not_like*' from '*not like*'. These special tokens are handled internally by the Sentiment Analyzer and are assigned a sentiment vector that is the negation of the sentiment originally associated with the target token (*'like'*, in this example.) In this context, negation refers to interchanging the positive and negative scores in the 2-dimensional sentiment vector. This enables us to better evaluate the semantics of phrases that incorporate negation, although limited to a one-word context. Besides, there may be several cases where the negative modifier precedes a verb, such as *'not work'*, where such a modification does not add any advantage. In such cases, for the sake of simplicity, we revert back to the old technique of treating the two tokens separately.

3.2.4 Sentiment Extraction from Chunks

Lexicon-Based Sentiment Analysis

As discussed in Section 2.2.3, a sentiment lexicon is a list of words and their associated sentiments. It can also be expressed as a mapping from a token to a

2-dimensional sentiment vector, where the values in each dimension correspond to *positive* and *negative* sentiment scores respectively. We opt to go for binary classification of the sentiment space, as *neutral* sentiments are also considered favorable for advertising. However, we ensure that they are differentiated from a positive sentiment, by assigning 0.5 points to the positive score of a neutral sentiment vector. In other words, every neutral sentiment extracted, i.e. [0, 0] is transformed to [0.5, 0]. We use a sentiment lexicon to determine the sentiment of a chunk of text by summing up the sentiment vectors of the constituent words. There are several other approaches such as taking the average of these sentiment vectors, or the `argmax` of the sum, etc. Preliminary experiments have shown that the sum of the sentiment vectors, gave the best results for our application. Several text pre-processing steps discussed in the previous section, like stop word removal, lemmatization, etc. are employed to avoid common pitfalls in text analysis. The main advantage of this approach is the independence of the model from the target domain, which may not be always possible in the case of supervised learning techniques like SVM classifiers or Neural Networks, as the latter typically need domain-specific training data. Lexicon-based approaches are also simple and time-efficient, which is crucial in real-time applications like ad-bidding. In this project, we chose to explore three different sentiment lexicons:

- **Opinion-Miner [64]:** The Opinion Mining Lexicon has been put together by Bing Liu and Minqing Hu. It consists of two lists of words, corresponding to the positive and negative sentiment respectively. There is a total of 6789 words in the Opinion Mining Lexicon, with 2007 positive words and 4782 negative words. It is an easily available resource and therefore we include it as a strong baseline.

- **SocialSent Lexicon from `r/news` [65]:** The SocialSent lexicon is based on the argument that the sentiment of a word depends on the context. For example, "soft" in itself may be associated with a gentle and kind nature, a positive sentiment; but calling a hockey player "soft" may be

an insult and carries a negative sentiment [6]. SocialSent, therefore, treats each such domain independently and developed a sentiment lexicon from the comments on different subreddits on Reddit. Since we like to apply our work to news articles, we decided to use lexicon developed from the `r/news` community, a community where news is shared and discussed on Reddit. The mean sentiments are thresholded at zero and converted to a 2d sentiment vector.

- **SentiWordNet Lexicon [66]:** SentiWordNet is based on WordNet synsets and identifies each lexicon entry with Part of Speech tag and WordNet synset ID. It provides a 3d sentiment vector with positive, negative and objective scores, such that the sum of these scores adds up to 1. Since APNEA works with 2d sentiment vectors, we consider only the positive and negative scores. Because of its representation, SentiWordNet needs the POS tag and the synset ID of each token to access its sentiment, which is an additional overhead and may contribute to the latency of the entire system .

Machine Learning Based Sentiment Analysis

Machine Learning based approach to Sentiment Analysis is more complicated than Lexicon-based approach but it is quite robust. In general, ML-based approaches have a higher recall than Lexicon-based approaches, whereas the latter have high precision. The downside of ML-based techniques is that they require a lot of training data, usually from the same domain. Also, it is not easy to modify the model once trained. In this project we choose to include two ML-based approaches in our Sentiment Extraction subtask:

- **SVM:** Support Vector Machines (SVMs) have long since been used for Sentiment Classification, as discussed in Section 2.2.4. Since we do not have any domain-specific training data to train our SVM model, we chose to use the Sentiment Labelled Sentence Dataset from UCI [54] for cross-domain training. The dataset consists of reviews from 3 major websites

[6] https://nlp.stanford.edu/projects/socialsent/

on the World Wide Web: Yelp, IMDB, and Amazon. We combine all the reviews and shuffle them randomly before training. We used the 50-dimensional GloVe [53] word embeddings and converted each sample to a sentence vector by averaging the constituent vectors. Several pre-processing steps like lemmatization and stop word removal were used to exclude irrelevant tokens. Our model reported a mean Accuracy of `76.4%` across 5 random train-test splits, with the best penalty parameter found to be `16.0`, using internal cross-validation.

- **Stanford Sentiment Analyzer [67]:** We have also explored the Recursive Neural Tensor Network (RNTN) trained on the Stanford Sentiment Treebank [67]. RNTN is a state-of-the-art deep learning model that works on "tensor-based composition of intermediate results at individual nodes of the parse tree structure of an input sentence". The model achieves an accuracy of around 85% at sentence level on the Treebank test set. Besides its strong performance, the model is easily available through the Stanford CoreNLP Suite, making it easy to integrate into our experiments. The RNTN model that comes with the CoreNLP Suite returns the sentiment of the input text as one of 5 categories, which we map to 2d sentiment vectors: Very Positive ([2, 0]), Positive ([1, 0]), Neutral ([0, 0]), Negative ([0, 1]) and Very Negative ([0, 2]). For more details on the model architecture, we direct the reader to [67].

3.3 Page-Ad Matching

In this section, we describe in detail the last stage of the APNEA pipeline – page-ad matching. This stage comprises of computing the document vector from the sentiments extracted from the chunks and calculating the relevance scores between all advertisers and the document as the cosine similarity of the ad-vectors computed in Section 3.1 and the computed document vector.

Before we go into the details of the page-ad matching process, we introduce two more functionalities of the APNEA system that further help it in solving the problem of unfortunate placements.

3.3.1 Blacklists and Targeted Sentiment

Blacklists

Understanding the sentiment associated with a web page is sometimes not enough to avoid an unfortunate placement. For example, a news article that talks about rising obesity rates in the United States may also mention triggering keywords like *food* and *restaurants*. An advertiser for fast food, say McBurgers, may place an advertisement on this web page and hurt their brand image. In other cases, a brand may have a history of controversies on a particular topic that they would like to avoid. In such cases, a blacklist would come in handy to help an advertiser avoid an unfortunate placement in contexts that they do not wish to appear in.

In our work, we use a simple list of keywords that an advertiser provides as a blacklist. During page-ad matching, whenever a blacklisted keyword has been encountered in the document, the advertisers that blacklisted the said keyword are marked and removed from the bidding process.

Blacklist entries are entirely optional for each advertiser. For the sake of simplicity, we examined a validation set and added relevant blacklist entries to appropriate advertisers. More details regarding the validation set will be covered in Chapter 4.

Targeted Sentiment

Often, news articles that talk negatively (or positively) about a brand/company, mention the same in the title of the article. For example, *SocialNetwork accused of striking 'secret deals over user data'*, exclusively targets the company *SocialNetwork* over shady data practices. This would be relevant to *SocialNetwork* but, due to its negative orientation towards the subject, will harm the company if an advertisement is placed on this web page.

However, we also recognize that this presents an opportunity to bump up the competitors of *SocialNetwork* and boost their chances of advertising themselves as better alternatives. In order to achieve this, we incorporate targeted sentiment into our system. The system identifies targets by parsing

the title of the current article and matching them against its database of existing advertisers. If no targets are found, the system continues to evaluate all brands according to the sentiments reflected in the document. Otherwise, the relevance scores of all non-targeted advertisers are calculated on the absolute measure of the computed document vector. This has the effect of not penalizing the non-targeted advertisers based on the sentiment context of the document.

The main assumption behind targeted sentiment is, given any targeted advertiser, we assume that its competitors have bid phrases similar to its own. Therefore, its competitors will also get triggered due to the semantics of the article. Since the competitors are not penalized, they get bumped up to a higher rank and have better chances of getting placed on the web page. If the article is positively oriented towards the targeted advertiser, the computed document vector should have very few entries that are in opposite orientation to corresponding entries in the ad-vector of the targeted advertiser, thereby increasing its chances to bid on the document.

3.3.2 Page-Ad Matching Algorithm

The algorithm works as follows, the document retrieved is divided into *chunks* as described in Section 3.2.2. For each chunk, we use the sentiment analyzer discussed in Section 3.2.4 to determine the sentiment of the chunk. The extracted sentiments are scaled according to the 'attention weights' described in Section 3.2.2.

The algorithm iterates over each token in a chunk and uses the mapping `keyword_to_advertiser` constructed in Algorithm 1, to find the triggering keywords and the corresponding advertisers who are bidding on the said words. The bidding advertisers are added to a set to make up the candidate advertisers C. For each bidding keyword in the chunk, the extracted sentiment vector is added to the corresponding dimension of the document matrix M_d. It is also at this stage that the tokens are checked for any blacklisted key-phrases and the corresponding advertisers B are removed.

In order to compute the document vector from the matrix $M_d \in \mathbb{R}^{2\times|V|}$,

where $|V|$ is the size of the vocabulary, we apply a scoring function column-wise on M_d. A scoring function takes a 2d sentiment vector and outputs a real value, reflective of the sentiment expressed by the input vector. If a candidate advertiser is sentiment-agnostic towards a set of keywords, the scores along those dimensions are taken as absolute values. Similarly, if targeted sentiment is enabled and the candidate advertiser is not a targeted advertiser, then the scores along all dimensions are taken as absolute values, as described in Section 3.3.1. Then, the relevance score for each candidate advertiser is calculated as the cosine similarity between the ad-vector constructed in Section 3.1 and the computed document vector. The formula for cosine similarity between an ad-vector and a document vector is given as:

$$cos_{sim}(\vec{a}, \vec{p}) = \frac{\vec{a} \cdot \vec{p}}{\|\vec{a}\| \|\vec{p}\|}$$

where, $\vec{a}$ represents the ad-vector and $\vec{p}$ represents the document vector.

Finally, the advertisers are sorted in decreasing order of their relevance scores. Algo. 2 describes the page-ad matching process.

3.3.3 Scoring Functions

The system uses several different scoring functions which we adopt from [68]. The work intends on "incorporating sentiment into the similarity measure between two words", which we extend to vectors in this thesis. We also propose a new scoring function based on intuitive metrics and evaluate its performance against the rest. The scoring functions take a 2d sentiment vector as input and output a real value, which is reflective of the sentiment of the input. In the following formulae, we represent the positive score in the sentiment vector sv by p and the negative score by n.

- **Sentiment Difference (SD):** This function calculates the difference between the positive and the negative scores of a sentiment vector.

$$Score_{SD}(sv) = p - n$$

.

Input : Triggering Page, P
Input : Ad-vectors from Algo. 1
Output: Relevance scores, *scores*

Let C:= {} ← set of candidate advertisers
Let B:= {} ← set of blacklisted advertisers
Let T ← set of targeted advertisers from the title of P
Let M_d be the document matrix, $M_d \in \mathbb{R}^{2\times|V|}$
for *every chunk in* P **do**
    $S \leftarrow$ `SentimentAnalyzer` (*chunk*) $*$ `attention_weights` (*chunk*)
    Let C_i be the advertisers interested in keywords in *chunk*
    Let B_i be the advertisers with blacklisted keywords in *chunk*
    Let k be keywords in *chunk*
    for k_i *in* k **do**
        Add S to column k_i of M_d
    end
end
$C = \bigcup_i C_i$
$B = \bigcup_i B_i$
$C = C \setminus B$
Let *scores* be an empty collection
for *every* a_i *in* C **do**
    $\vec{d} \leftarrow$ `vectorize` (M_d, a_i, T)
    Let $\vec{a_i}$ be the ad-vector of a_i from Algo. 1
    Add (a_i, Sim_{cos} ($\vec{a_i}$, $\vec{d}$)) to scores
end
Sort *scores* in decreasing order, by score.
return *scores*

Algorithm 2: Page-ad matching.

- **Sentiment Maximum (SM):** This function calculates the maximum of the positive and the negative scores of a sentiment vector. The scores are differentiated by the sign attached, for example, $max_{SM}([5, 2]) = +5$ whereas $max_{SM}([2, 7])$ = -7.

$$Score_{SM}(sv) = max_{SM}(p, n)$$

.

- **Threshold Difference (TD):** The SD metric does not account for sentiment vectors who have a zero score, but we may wish to include the

Input : Document Matrix, M_d
Input : Advertiser, a
Input : Targeted Advertisers, T
Output: Document Vector, $\vec{d}$

Function `vectorize`(M_d, a, T)**:**
  Let $\vec{d} \leftarrow$ Apply scoring function on M_d column-wise.
  Let L be the keywords that a is sentiment-agnostic on, from $\mathsf{sent\_sensitivity}_a$ in Algo. 1
  for *every* l *in* L **do**
    Set $\vec{d_l} \leftarrow$ `abs` $(\vec{d_l})$
  end
  if $T \neq \emptyset$ *AND* a *not in* T **then**
    Take `abs` $(\vec{d})$ on all remaining dimensions of $\vec{d}$.
  end
  return $\vec{d}$

Algorithm 3: Vectorizing the Document Matrix M_d.

contribution of those inputs in our system. The Threshold Difference (TD) metric adds a threshold of 1 to all the SD scores while preserving the original sign.

$$Score_{TD}(sv) = sign(p - n) * (1 + abs(p - n))$$

.

- **Threshold Maximum (TM):** The TM function is similar to the TD function where we use the Sentiment Max (SM) scores instead of the SD scores.

$$Score_{TM}(sv) = sign(max_{SM}(p, n)) * (1 + abs(max_{SM}(p, n)))$$

.

- **Logarithmic Ratio (RL):** In addition to the scoring functions above, which we adopted from [68], we formulated our own metric which we intend to evaluate against the others. The function evaluates a sentiment vector sv as follows:

$$Score_{RL}(sv) = \begin{cases} max(p, \epsilon), & \text{if } n = 0 \\ p/n * \log_{10}(abs(p - n) + 1) * sign(p - n), & \text{otherwise} \end{cases}$$

The reasoning behind the choice of function is as follows:

- For the base case when the negative sentiment score is zero, the function returns the max of the positive score or a small ϵ (default value 0.01) as a threshold.
- For all other cases, we included 3 sub-parts in the formula:
  1. p/n: The ratio of the positive score to the negative score gives a general idea of the polarity of the input vector, depending on whether it is greater than or less than 1. However, the ratio can be misleading at times, when different scores can have the same ratio but may imply different strengths of the sentiment vector.
  2. $\log_{10}(abs(p-n)+1)$: The second sub-part takes this into account by taking the difference between the p and n, which will increase as the common factor between the scores also increases. The logarithm suppresses this difference from increasing linearly, while the 1 avoids invalid inputs to the logarithm function.
  3. $sign(p-n)$: The third sub-part is the sign of the difference between p and n and it assigns the polarity of the sentiment to the final score.

3.4 Real-time Performance

In computing, the run-time of a program is defined as the time during the execution of a program. Latency, in this context, is the delay in a bidding system between a user visit and an ad being displayed. In Chapter 2, we presented the steps behind Real-Time Bidding (RTB) and why it is essential for any augmenting functionality to work in real-time. APNEA attempts to achieve this in several ways:

- Firstly, the system pre-processes the advertisements offline, therefore eliminating the need to expand terms and construct the ad-vector for

each advertiser at run-time. The computed ad-vectors are stored in memory and retrieved upon request.

- Secondly, unlike SOCA [35], APNEA eliminates term expansion at the document level. In other words, the terms in the document are analyzed verbatim, without any expansion at run-time. This greatly reduces the processing time of each document, while not sacrificing much on accuracy since the ad-vectors are already term-expanded.

- Thirdly, the proposed system primarily utilizes lexicon-based sentiment analyzers, which are faster than ML-based approaches due to the absence of additional steps such as the conversion of input text to an embedding. Besides, while the SOCA framework only supports the SentiWordNet lexicon, we support other lexicons such as the SocialSent Lexicon and Opinion Mining Lexicon as the latter do not require the input tokens to be in Wordnet Synset Format, further speeding up the ad-matching process.

- Finally, APNEA is parallelizable. The document matrix construction can be mapped to multiple processes that handle different parts of the input document. The final matrix can be constructed by simple summation of the individual results. In a similar fashion, the relevance scores of each advertiser can also be computed parallelly, making APNEA practical to port to existing Demand-Side-Platforms (DSP). Figure 3.1 shows an outline of how APNEA can be parallelized.

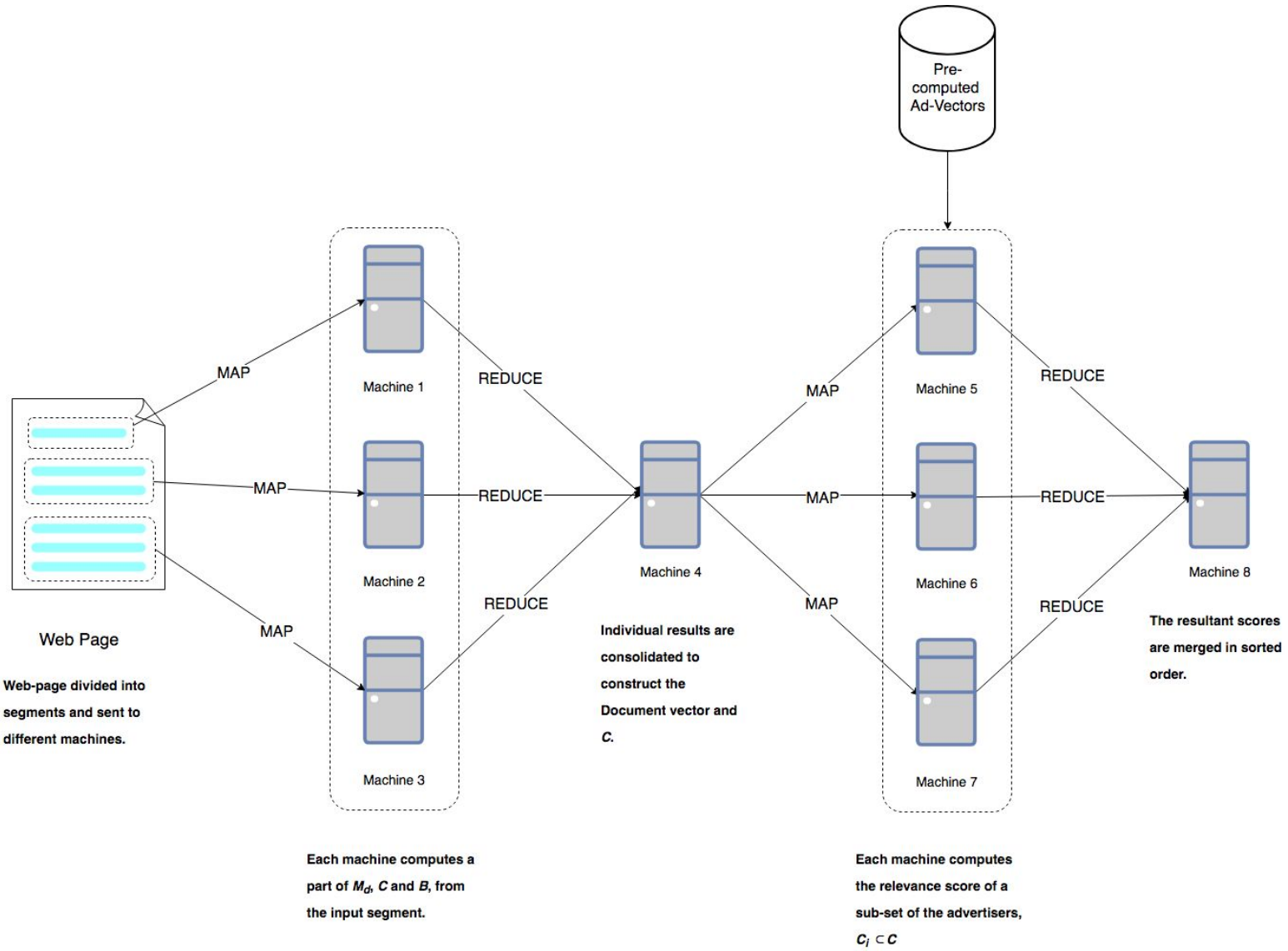

Figure 3.1: Parallel working of the APNEA system. Here, C and B denote the candidate advertisers and blacklisted advertisers respectively, while M_d is the document matrix.

Chapter 4

Experimental Results

In this chapter, we go into the details of the evaluation of our system against baselines and explore how the different parameters affect the performance of the system on test data. We also discuss how the test set was collected and other experimental details. The code and data used for the evaluation will be available at our online repository[1].

4.1 Data

All contextual advertising systems rely on an input advertiser database to represent the set of advertisements. We use a modified version of the Open Advertising Dataset [60] from Google, which consists of data related to Sponsored Search, with advertisers, landing page URLs, click-through-rates etc. More precisely, we use the web pages dataset, and edit the advertisers and the keywords, such that relevant articles could be found on Google News and the `r/news` subreddit. The final version of the advertisements dataset consisted of 68 advertisers, each advertising a single advertisement.

Since APNEA does not have any learning component, we do not have any training data in our experimental setup. However, we use a small additional dataset of 25 news articles as a validation set to explore and tweak the different parameters of the system. The details of the validation set are found in Table 4.1:

[1] `https://github.com/blumonkey/apnea`

Articles	Sentences	Tokens	Avg. No. of Sentences	Avg. No. of Tokens
25	685	13940	27.4	557.6

Table 4.1: Validation Set: Statistics

The validation set was used to develop the strategy of targeted advertising and reduction factor. The blacklist used for all experiments in this chapter was also constructed using the validation set.

The test set consists of 177 articles that were selected manually using Google News Search and Reddit `r/news` Search. The dataset was annotated by 3 annotators based on the criterion: 'Is this a good place to advertise my brand?'. The annotators worked independently and the results are consolidated by taking the majority vote. The average pair-wise Kappa agreement coefficient is 0.87. Table 4.2 shows the statistics of the test set.

Articles	Sentences	Tokens	Avg. No. of Sentences	Avg. No. of Tokens
177	4381	93119	24.75	526.10

Table 4.2: Test Set: Statistics

Each instance in either set consists of a URL which corresponds to the triggering document, and a list of advertisers, each of which is marked with one of three different labels: `UNSAFE`, `SAFE` and `DONT_CARE`. Advertisers marked `UNSAFE` are those that the annotators believe will get hurt if their brand is placed on the triggering document. Advertisers marked `SAFE` are those that are not only safe to be placed on the triggering document but are also relevant to the context of the document. Advertisers marked `DONT_CARE` are those who are not relevant to the context of the triggering document but may not be harmed when advertised there.

Bateman [69] describes how 3-4 advertisements per page is the most ideal number of ads for the best user experience. We use this as a rule of thumb and consider only those ads that are in the top-5 ranks by relevance, to be placed on the triggering page. A ranking is desired if none of the `UNSAFE` advertisers are in the top-5 while at least one of the `SAFE` advertisers is in the top-5. Note

that there may be samples where no advertiser is `UNSAFE` (or `SAFE`).

This leads to two different types of errors that we come across in our evaluation.

- **Type 1 Errors** ($type_1$): These are the samples where at least one `UNSAFE` advertiser is in the top-5 ranks. In other words, this is a false-positive ranking where an `UNSAFE` advertiser is marked safe to advertise.
- **Type 2 Errors** ($type_2$): These are the samples where none of the `SAFE` advertisers are in the top-5 ranks. In other words, this is a false-negative ranking where a `SAFE` advertiser is falsely marked as `UNSAFE` or `DONT_CARE`.

We use accuracy as the metric of choice because the outcome of the proposed system is not the individual labels of the advertisements, but rather the ranking as a whole. The accuracy of a system can be calculated as follows:

$$Accuracy = \frac{total - type_1 - type_2}{total}$$

4.2 Experimental Results

4.2.1 Baselines

In Machine Learning, a baseline is any system that aims to solve the same problem as a target system. A baseline may be a simple solution for the problem at hand, or it may be a previous work that we try to compare against.

We compare our system against two Baselines, namely the traditional Contextual Advertising System (CA) and SOCA by Fan and Chang [35]. CA is implemented using APNEA without the sentiment component. The system computes the term-frequency (tf) vector of the document as the document vector and the cosine similarity between the document and the ad-vectors is used to calculate relevance. We also include two baselines, BL_POS and BL_NEG, where all sentiments are treated as positive and negative, respectively. All baselines based on APNEA have term expansion and s-flag enabled, using the Threshold Difference (TD) scoring function and working at Sentence

level analysis, with a reduction factor of 1.0. Enabling the *s*-flag functionality respects the individual sentiment-sensitivity settings of the advertisers.

SOCA is implemented as described in [35], with the best parameters and default values described in the work. We constructed the *tf-idf* vectors for both the advertisements and the test documents to calculate the cosine similarity component of the scoring metric used by SOCA. However, they perform poorly at the current task, therefore, we also use 300-dimensional word2vec embeddings [52] and average the embeddings of all tokens in a document to construct its representation. Furthermore, the authors of SOCA [35] do not define any default value for parameter δ in the web-based term expansion stage of the SOCA pipeline. Since the possible values for δ are in the range of $(0, \infty)$, we experimented with a few values and observed that they add noise to the system's performance. Therefore, we eliminate the web-based term expansion of SOCA. SOCA is used as a strong baseline as it tries to solve the problem of unfortunate placements. It can be considered to be the state-of-the-art in this particular field of study and therefore sets a high bar for APNEA to compare against.

Furthermore, SOCA reports its best accuracy at 68.2% using an SVM based sentiment analyzer that was trained on reviews from *epinion.com*. Since the data is no longer available at the website, we use our SVM model from Section 3.2.4, trained on the UCI dataset as the sentiment analyzer for SOCA.

In order to make the comparison between our system and the baselines fair, we use the same configuration as the BL_POS / BL_NEG baseline, with the only difference being that sentiment is now extracted using the Opinion Mining Sentiment Lexicon. We represent this configuration of APNEA as Basic Conf. in Table 4.3. In order to remove the effect of the sentiment lexicon used, we also run SOCA on the Opinion Mining Lexicon [64], denoted as SOCA w/ OM.

Metrics / Models	$type_1$ Errors	$type_2$ Errors	*Accuracy*
CA	73	8	0.54
BL_POS	75	**6**	0.54
BL_NEG	**0**	104	0.41
SOCA w/ SVM, *tf-idf* Web-expansion Enabled	71	9	0.55
SOCA w/ OM, *tf-idf* Web-expansion Enabled	71	9	0.55
SOCA w/ SVM, word2vec Web-expansion Disabled	26	57	0.53
SOCA w/ OM, word2vec Web-expansion Disabled	23	62	0.52
APNEA (Basic Conf.)	9	27	**0.80**

Table 4.3: Errors and Accuracy for the Baseline Models

Table 4.3 shows the performance of the different baselines on the test set and how APNEA out-performs all baselines with a significant margin. We can observe that the accuracies of CA and BL_POS are almost the same because under the Threshold Difference (TD) scoring function, they both behave in a similar manner. Furthermore, BL_POS and BL_NEG have high $type_1$ and $type_2$ errors respectively, which is self-explanatory.

SOCA with *tf-idf* vectors and web-expansion performs at par with the regular CA system. We suspect this is because of the noise introduced by web-expansion and the *tf-idf* vectors not capturing the document/ad representation to the fullest extent on a small dataset.

SOCA with word2vec embeddings and web-expansion disabled, manages to reduce the number of $type_1$ errors but has a large number of $type_2$ errors, which suggests that in the process of removing negative sentences from the article, it is also losing a number of important triggering keywords that may be essential for matching the right advertisers to the document. The version of SOCA with Opinion Mining Lexicon is not more effective than APNEA,

Metrics / Models	$type_1$ Errors	$type_2$ Errors	*Accuracy*
At Sentence Level Analysis			
APNEA (Basic Conf.)	**9**	27	0.80
APNEA (w/ TS)	11	26	0.79
APNEA (w/ TS & BLST)	11	26	0.79
APNEA (w/ TS & BLST, $r = 2.0$)	12	**23**	**0.80**
At Document Level Analysis, $r = 2.0$			
APNEA (DOC.)	5	30	0.80
APNEA (DOC w/ TS)	7	26	0.81
APNEA (DOC w/ TS & BLST)	7	26	0.81

Table 4.4: Errors and Accuracy with Targeted Sentiment and Blacklist

which suggests that the difference in the methodology is the primary player in these results.

4.2.2 Variations in APNEA

Our system has over 1200 possible configurations. Since it is impractical to evaluate the system over each configuration, we run experiments with increasing functionality of the system to demonstrate the effect of each feature. We also explore the effect of different sentiment analyzers, scoring functions, and analysis levels on the APNEA system.

Effect of Targeted Sentiment and Blacklist

First, we improve upon our baseline APNEA (Basic Conf.), used in Table 4.3, by including Targeted Sentiment (TS) and Blacklist (BLST). The effect of either of these is not prominent at the Sentence Level Analysis, but it is clearly evident at Document Level Analysis. We also set the reduction factor to 2.0 to see the effect of decreasing the importance of the expanded terms. The results are found in Table 4.4.

We attribute the small decrease in the accuracy at Sentence Level Analysis with Targeted Sentiment, to the samples in the dataset where enabling targeted advertising would penalize the `SAFE` advertisers to an extent that they move out of the top-5 ranks by relevance score. The blacklist constructed from the small validation set proves to be ineffective on the larger test set. However, the effect of the reduction factor is evident from the reduced number of $type_2$ errors, resulting in improved accuracy.

However, in the case of Document Level Analysis, we can see that Targeted Sentiment, boosts the accuracy of the system, albeit by a small percentage. This is because, in document level analysis, all the triggering keywords get associated with the same sentiment i.e. the sentiment of the document. By using Targeted Sentiment, the sentiment of the document reflects only on the advertisers that are targeted in the title, letting competitors to secure a position in the top-5 ranks by relevance. The effect of the blacklist on the performance of the system at this level is again non-existent.

Other Configurations

We also explore other configurations where the Sentiment Analyzer, Analysis Level and Scoring function have been changed. The base model for these variations is the APNEA with Opinion Mining Lexicon at Sentence Level Analysis, using the TD scoring function. All the results presented in Table 4.5 are at a reduction factor $r = 2.0$ with s-flag, Targeted Sentiment and Blacklist enabled.

Metrics / Models	$type_1$ Errors	$type_2$ Errors	*Accuracy*
Variations in Sentiment Analyzers			
APNEA (SentiWordNet)	29	**19**	**0.73**
APNEA (RNTN)	**2**	91	0.47
APNEA (SocialSent)	40	31	0.60
APNEA (SVM)	5	50	0.69
Variations in Scoring Functions			
APNEA (TM)	**12**	**22**	**0.81**
APNEA (RL)	20	25	0.75
APNEA (SD)	12	27	0.78
Variations in Analysis Level			
APNEA (SENT_NGRAM, $n = 3$)	6	**29**	**0.80**
APNEA (SENT_NGRAM, $n = 5$)	**5**	32	0.79

Table 4.5: Errors and Accuracy at Other Configurations

4.3 Run-time Analysis

In order to demonstrate that our APNEA is faster than the SOCA framework, we evaluate the run-times of both the systems. APNEA system uses the Opinion Mining Lexicon at Sentence and Document Level Analysis, with the *s*-flag, Targeted Sentiment and Blacklist enabled, with a reduction factor $r = 2.0$. To keep the comparison fair, the SOCA system is also evaluated on the Opinion Mining Lexicon, without web expansion enabled on the input document. Both systems are evaluated on identical machines on 25 random URLs from the test set, across 5 independent runs.

In order to reduce the implementation bias in SOCA, both systems have pre-processing of the advertisements done offline, and the time taken for fetching the Wikipedia articles on the document expansion stage for SOCA has

been excluded from the run-time computation. However, we did not index the document synsets as suggested in [35], because documents on the World Wide Web are highly volatile and dynamic and indexing information on the document side does not seem practical in a real-world scenario.

Model Name	**Time (s)**					**Mean**	**S.D**
APNEA (SENT.)	6.05	5.94	5.92	5.91	5.89	5.94	0.06
APNEA (DOC.)	2.68	2.64	2.55	2.60	2.54	**2.60**	**0.05**
SOCA (w/o Web Expansion.)	36.09	40.25	46.72	40.41	56.16	43.93	7.00

Table 4.6: Run-times across 5 runs, using SOCA and APNEA

Table 4.6 shows the run-times of each system. It is evident that our proposed system is faster than SOCA, even without web expansion enabled in the latter. Furthermore, other experiments show that without the ontological similarity component, the average time taken by the SOCA system is still around 8.70 ($\pm$0.84) seconds, which is still slower than APNEA at Document Level Analysis.

4.4 Summary

In this chapter, we experimentally evaluated the proposed system against a set of baselines. We go into the details of collection and curation of our advertisement database and the validation/test data. We explain the different configurations of our system and systematically explore the effects of each. Finally, we also evaluate APNEA against SOCA, on run-time performance across 5 independent runs. Although we were not able to explore all interesting configurations of the system, we would like to extend that to our future work. While it is evident that APNEA greatly outperforms the baselines, we can only draw conclusions constrained by the data on which the model has been built and tested.

Chapter 5

Conclusions and Future Work

In this work, we defined the problem of *unfortunate placement* and tried to address it using the APNEA system. We first introduced the problem statement and the challenges associated with it, namely avoiding an unfortunate placement and at the same time, maintaining ad-relevance with respect to other advertisers. We then discussed how Sentiment Analysis can help provide the right context to avoid an unfortunate placement, with discussions on the different approaches in Sentiment Analysis.

It is imperative to understand the online bidding mechanism to develop a practical advertisement bidding system for this problem. Therefore, we discuss the history and mechanisms behind Real-time Bidding that is prevalent in the advertising world today. Although very few have addressed the problem of unfortunate placement, we discuss the methodology, merits and demerits of two papers that deal with this problem either directly or indirectly: DASA from [58] and SOCA from [35]. DASA deviates from the current work by focusing only on the negative sentiments and aims to match competitors who are better at the aspect that is being targeted. SOCA addresses the problem of unfortunate placements in the blogosphere, but we extrapolate it to our work on news articles.

We go into the details of the proposed system, and evaluate it against set baselines, on a human-annotated dataset. Details regarding the advertisement data, the sentiment analyzers and the test data are provided. Experimental results show that our proposed system outperforms the defined baselines by

a significant margin on the annotated dataset. Run-time analysis shows that our system is also faster than the baseline SOCA system by up to 94%.

5.1 Future Work

The present work has many challenges, in the fields of Real-time bidding and Sentiment Analysis. We use Sentiment Analysis to provide the necessary context to APNEA, but we would like to extend this to Emotion Mining, which provides much more granularity and control to the advertisers. Due to the lack of annotated data, we have primarily preferred a lexicon-based approach to the Sentiment Analysis component of APNEA. With the availability of domain-specific annotated training data, we would like to test how a learning-based approach performs against the lexicon-based approach. Finally, we would also like to evaluate other interesting configurations of the proposed system to explore how the different settings complement each other.

References

[1] eMarketer, "U.S. ad spending: emarketer's updated estimates and forecast for 2017." https://www.emarketer.com/Report/US-Ad-Spending-eMarketers-Updated-Estimates-Forecast-2017/2002134, 2017. [Online; accessed 11-February-2019].

[2] G. LLC, "Alphabet announces first quarter 2018 results," 2018.

[3] A. Broder, M. Fontoura, V. Josifovski, and L. Riedel, "A semantic approach to contextual advertising," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, (New York, NY, USA), pp. 559–566, ACM, 2007.

[4] P. Chatterjee, D. L. Hoffman, and T. P. Novak, "Modeling the clickstream: Implications for web-based advertising efforts," *Marketing Science*, vol. 22, no. 4, pp. 520–541, 2003.

[5] C. Wang, P. Zhang, R. Choi, and M. D'Eredita, "Understanding consumers attitude toward advertising," *AMCIS 2002 Proceedings*, p. 158, 2002.

[6] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[7] A. Yadollahi, A. G. Shahraki, and O. R. Zaiane, "Current state of text sentiment analysis from opinion to emotion mining," *ACM Comput. Surv.*, vol. 50, pp. 25:1–25:33, May 2017.

[8] J. Wang, W. Zhang, and S. Yuan, "Display advertising with real-time bidding (RTB) and behavioural targeting," *CoRR*, vol. abs/1610.03013, 2016.

[9] Wikipedia contributors, "Display advertising — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Display_advertising&oldid=863922737, 2018. [Online; accessed 19-October-2018].

[10] Wikipedia contributors, "Advertising network — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Advertising_network&oldid=864449827, 2018. [Online; accessed 19-October-2018].

[11] Wikipedia contributors, "Online advertising — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Online_advertising&oldid=854232513, 2018. [Online; accessed 19-October-2018].

[12] Wikipedia contributors, "Click-through rate — Wikipedia, the free encyclopedia," 2019. [Online; accessed 20-February-2019].

[13] J. Wang, W. Zhang, and S. Yuan, "Display advertising with real-time bidding (rtb) and behavioural targeting," *arXiv preprint arXiv:1610.03013*, 2016.

[14] B. J. Jansen and A. Spink, "Sponsored search: is money a motivator for providing relevant results?," *Computer*, vol. 40, no. 8, 2007.

[15] S. Yuan, J. Wang, and X. Zhao, "Real-time bidding for online advertising: Measurement and analysis," in *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ADKDD '13, (New York, NY, USA), pp. 3:1–3:8, ACM, 2013.

[16] H. Zhang, W. Zhang, Y. Rong, K. Ren, W. Li, and J. Wang, "Managing risk of bidding in display advertising," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 581–590, ACM, 2017.

[17] W. Zhang, L. Chen, and J. Wang, "Implicit look-alike modelling in display ads: Transfer collaborative filtering to CTR estimation," *CoRR*, vol. abs/1601.02377, 2016.

[18] Y. Cui, R. Zhang, W. Li, and J. Mao, "Bid landscape forecasting in online ad exchange marketplace," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 265–273, ACM, 2011.

[19] K. Ren, W. Zhang, Y. Rong, H. Zhang, Y. Yu, and J. Wang, "User response learning for directly optimizing campaign performance in display advertising," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pp. 679–688, ACM, 2016.

[20] L. Kim, "New adwords ad ranking formula: What does it mean?." `https://searchengineland.com/new-adwords-ad-ranking-formula-what-does-it-mean-174946`. Accessed: 2019-02-20.

[21] M. D. Marketing, "Adwords auction; how keyword bidding works." `https://www.youtube.com/watch?v=SZV_J92fY_I`. Accessed: 2019-02-20.

[22] I. I. A. Bureau, "How an ad is served with real time bidding (rtb) - iab digital simplified." `https://www.youtube.com/watch?v=-Glgi9RRuJs`.

[23] Wikipedia contributors, "Demand-side platform — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Demand-side_platform&oldid=861149846`, 2018. [Online; accessed 31-October-2018].

[24] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: estimating the click-through rate for new ads," in *Proceedings of the 16th international conference on World Wide Web*, pp. 521–530, ACM, 2007.

[25] I. Trofimov, A. Kornetova, and V. Topinskiy, "Using boosted trees for click-through rate prediction for sponsored search," in *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, p. 2, ACM, 2012.

[26] A. Kennedy and D. Inkpen, "Sentiment classification of movie reviews using contextual valence shifters," *Computational intelligence*, vol. 22, no. 2, pp. 110–125, 2006.

[27] T. T. Thet, J.-C. Na, and C. S. Khoo, "Aspect-based sentiment analysis of movie reviews on discussion boards," *Journal of information science*, vol. 36, no. 6, pp. 823–848, 2010.

[28] H. Kang, S. J. Yoo, and D. Han, "Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews," *Expert Systems with Applications*, vol. 39, no. 5, pp. 6000–6010, 2012.

[29] D. Jurafsky, V. Chahuneau, B. R. Routledge, and N. A. Smith, "Narrative framing of consumer sentiment in online restaurant reviews," *First Monday*, vol. 19, no. 4, 2014.

[30] Y. Dang, Y. Zhang, and H. Chen, "A lexicon-enhanced method for sentiment classification: An experiment on online product reviews," *IEEE Intelligent Systems*, vol. 25, no. 4, pp. 46–53, 2010.

[31] H. Cui, V. Mittal, and M. Datar, "Comparative experiments on sentiment classification for online product reviews," in *AAAI*, vol. 6, pp. 1265–1270, 2006.

[32] N. Hossein and D. W. Miller, "Predicting motion picture box office performance using temporal tweet patterns," *International Journal of Intelligent Computing and Cybernetics*, vol. 11, no. 1, pp. 64–80, 2018.

[33] G. Coppersmith, M. Dredze, and C. Harman, "Quantifying mental health signals in twitter," in *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pp. 51–60, 2014.

[34] A. Ortigosa, J. M. Martín, and R. M. Carro, "Sentiment analysis in facebook and its application to e-learning," *Computers in Human Behavior*, vol. 31, pp. 527–541, 2014.

[35] T.-K. Fan and C.-H. Chang, "Sentiment-oriented contextual advertising," *Knowledge and information systems*, vol. 23, no. 3, pp. 321–344, 2010.

[36] G. Qiu, J. Bu, C. Chen, and F. Zhang, "Intelligent advertising for user generated content through sentiment analysis," in *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, pp. 330–333, IEEE, 2009.

[37] M. J. C. Samonte, J. M. R. Garcia, V. J. L. Lucero, and S. C. B. Santos, "Sentiment and opinion analysis on twitter about local airlines," in *Proceedings of the 3rd International Conference on Communication and Information Processing*, pp. 415–422, ACM, 2017.

[38] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[39] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 513–520, 2011.

[40] S. Vohra and J. Teraiya, "A comparative study of sentiment analysis techniques," *Journal JIKRCE*, vol. 2, no. 2, pp. 313–317, 2013.

[41] L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, and B. Liu, "Combining lexicon-based and learning-based methods for twitter sentiment analysis," *HP Laboratories, Technical Report HPL-2011*, vol. 89, 2011.

[42] T. Daouas and H. Lejmi, "Emotions recognition in an intelligent elearning environment," *Interactive Learning Environments*, pp. 1–19, 2018.

[43] B. Woolf, W. Burelson, and I. Arroyo, "Emotional intelligence for computer tutors," in *Workshop on Modeling and Scaffolding Affective Experiences to Impact Learning at 13th International Conference on Artificial Intelligence in Education*, vol. 616, 2007.

[44] D. Morrison, R. Wang, and L. C. De Silva, "Ensemble methods for spoken emotion recognition in call-centres," *Speech communication*, vol. 49, no. 2, pp. 98–112, 2007.

[45] J. Bhaskar, K. Sruthi, and P. Nedungadi, "Hybrid approach for emotion classification of audio conversation based on text and speech mining," *Procedia Computer Science*, vol. 46, pp. 635–643, 2015.

[46] Y. Rao, Q. Li, X. Mao, and L. Wenyin, "Sentiment topic models for social emotion mining," *Information Sciences*, vol. 266, pp. 90–100, 2014.

[47] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[48] B. Pang, L. Lee, *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[49] D. Zhang, H. Xu, Z. Su, and Y. Xu, "Chinese comments sentiment classification based on word2vec and svmperf," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1857–1863, 2015.

[50] B. Xue, C. Fu, and Z. Shaobin, "A study on sentiment computing and classification of sina weibo with word2vec," in *Big Data (BigData Congress), 2014 IEEE International Congress on*, pp. 358–363, IEEE, 2014.

[51] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1422–1432, 2015.

[52] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[53] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[54] D. Kotzias, M. Denil, N. De Freitas, and P. Smyth, "From group to individual labels using deep features," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 597–606, ACM, 2015.

[55] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[56] D. Maynard and M. A. Greenwood, "Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis," in *LREC 2014 Proceedings*, ELRA, 2014.

[57] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, "Sarcasm as contrast between a positive sentiment and negative situation," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 704–714, 2013.

[58] G. Qiu, X. He, F. Zhang, Y. Shi, J. Bu, and C. Chen, "Dasa: dissatisfaction-oriented advertising based on sentiment analysis," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6182–6191, 2010.

[59] J. Fleiss, B. Levin, and M. Paik, "Statistical rates and proportions. hoboken," 2003.

[60] Google, "Open advertising dataset - google code archive." `https://code.google.com/archive/p/open-advertising-dataset/`.

[61] "Corefannotator — stanford corenlp." `https://stanfordnlp.github.io/CoreNLP/coref.html`.

[62] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[63] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60, 2014.

[64] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177, ACM, 2004.

[65] W. L. Hamilton, K. Clark, J. Leskovec, and D. Jurafsky, "Inducing domain-specific sentiment lexicons from unlabeled corpora," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, vol. 2016, p. 595, NIH Public Access, 2016.

[66] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining.,"

[67] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

[68] A. Balamurali, S. Mukherjee, A. Malu, and P. Bhattacharyya, "Leveraging sentiment to compute word similarity," Citeseer.

[69] Scott S. Bateman, "How many ads per page? answer: 3 to 153." https://www.promisemedia.com/online-advertising/many-ads-per-page-answer-3-153, 2019. [Online; accessed 31-January-2019].

Appendix A

Sample Experiments on the Validation Set

A.1 Advertisements and Blacklist Data

In this Appendix, we provide sample snippets of the data used in our experimentations and the corresponding outputs on a select number of data points. The full version of the data can be found in the project repository.


```
{
        "Facebook": [
                ["connect␣with␣friends", false, 1.0],
                ["connect", false, 1.0],
                ["reach␣people", false, 1.0],
                ["social␣media", false, 1.0],
                ["online␣friends", false, 1.0]
        ],
        "Apple": [
                ["ipad", false, 1.0],
                ["iphone", false, 1.0],
                ["smartphone", false, 1.0],
                ["icloud", false, 1.0],
                ["phone", false, 1.0]
        ],
        "Hilton": [
                ["hotels", false, 1.0],
                ["book␣hotel␣room", false, 1.0]
        ],
        "Marriott": [
                ["hotels", false, 1.0],
                ["book␣hotel␣room", false, 1.0]
        ],
        "Dominos": [
```

```
                ["restaurants", false, 1.0],
                ["places␣to␣eat", false, 1.0],
                ["pizza", false, 1.0],
                ["chicken␣wings", false, 1.0]
        ],
        "McDonalds": [
                ["restaurants", false, 1.0],
                ["places␣to␣eat", false, 1.0],
                ["burgers", false, 1.0],
                ["shakes", false, 1.0]
        ],
}
```

Listing A.1: Minified Advertiser Data

We can observe that some of the bidding phrases are similar to a user's search query. This is because these phrases have been adopted from a Sponsored Search oriented dataset. We also provide a snippet of the blacklist used, with respect to the advertisements in Listing A.1.

```
"Dominos": [
        "obesity",
        "obese",
        "cholesterol",
        "heart␣attack"
],
"McDonalds": [
        "obesity",
        "obese",
        "cholesterol",
        "heart␣attack"
]
```

Listing A.2: Minified Blacklist

A.2 Sample URLs from the Validation Set

For the sake of demonstration, we have selected 5 URLs from the validation set, based on their relevance to the minified advertisement data. However, the experiments were run on all 68 advertisements, therefore the rankings shown in these experiments would also range from `0-67`. We do not show all the rankings of all the advertisers in these samples but restrict ourselves to those presented in the minified version.

The sample URLs used in this demonstration are:

1. **URL**: `https://tinyurl.com/y9vfqcux`.
   Title: Facebook accused of striking 'secret deals over user data'

2. **URL**: `https://tinyurl.com/y3cvedny`.
   Title: Woman says Hilton employee secretly filmed her naked in hotel room and tried to extort her

3. **URL**: `https://tinyurl.com/yyn7aqux`.
   Title: Airlines cleared for mobile phone use during flights

4. **URL**: `https://tinyurl.com/y67kza4w`.
   Title: 800-Pound Man Kicked Out of Hospital for Ordering Pizza

5. **URL**: `https://tinyurl.com/y7vksvqs`.
   Title: Obesity rates now top 35 percent in 7 states

The reader is encouraged to visit the above URLs and peruse through the articles. A brief idea of the subject matter discussed in each article can be gathered from the title provided. We do not repeat these URLs in the following discussions, to ensure brevity.

A.3 Experimental Results

We first present the ground truth for the sample articles. Given an article, each advertiser is marked as `SAFE`, `UNSAFE` or `DONT_CARE` by human annotators. Figure A.1 shows the annotated ground truth for the sample articles.

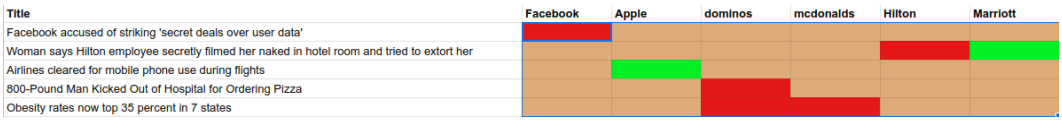

Figure A.1: Annotated Ground Truth for the Sample Articles.

Each cell in the dataset is coloured based on the annotation: a RED cell corresponds to `UNSAFE`, a GREEN cell corresponds to `SAFE` while an ORANGE

cell corresponds to `DONT_CARE`. For example, in the second row, corresponding to the second sample article, *Hilton* is marked `UNSAFE`, while *Marriott* is marked `SAFE`, while the remaining are `DONT_CARE`. The judgment behind marking *Marriott* as `SAFE` is that *Marriott* is a competitor for *Hilton* and therefore may have an advantage over the latter if advertised on this article.

We run the APNEA system on the sample articles to determine the winning bidders. The configuration used for this demonstration was APNEA using the Opinion Mining Lexicon at Sentence Level Analysis, with *s*-flag enabled, but with Targeted Sentiment and Blacklist disabled. The reduction factor used is $r = 2.0$.

Title	Facebook	Apple	dominos	mcdonalds	Hilton	Marriott
Facebook accused of striking 'secret deals over user data'	67	56	31	17	5	6
Woman says Hilton employee secretly filmed her naked in hotel room and tried to extort her	43	53	32	10	67	63
Airlines cleared for mobile phone use during flights	44	8	61	33	57	66
800-Pound Man Kicked Out of Hospital for Ordering Pizza	2	17	49	15	42	61
Obesity rates now top 35 percent in 7 states	28	19	2	0	41	55

Figure A.2: APNEA rankings on the Sample Articles, without Targeted Sentiment and Blacklist.

Figure A.2 shows the ranks as determined by the APNEA system. We can see that the rankings in rows `2`, `3` and `5` are not the desired rankings because in row `2` *Marriott* was also penalized with *Hilton*, resulting in a higher rank, making this a $type_2$ error, denoted by yellow highlights. Similarly, in row `2`, *Apple* is not in the top-5 ranks, making this a $type_2$ error as well. In row `5`, we have a case of unfortunate placement (a $type_1$ error, denoted by purple highlights), where *McDonalds* and *Dominos* are in the top-5 ranks on an article that discusses obesity.

We run the APNEA system on the sample articles again, with the Targeted Sentiment and Blacklist enabled, to demonstrate the effect of these features in our system. The remaining settings are the same as used for the experiments in Figure A.2.

The effect of Targeted Sentiment is evident in the ranking presented in row `2` of Figure A.3. *Marriott* is now in the top-5 ranks, while *Hilton* is prevented from bidding. Similarly, the ranking in row `5` of Figure A.3, demonstrates how the blacklist prevents *McDonalds* and *Dominos* from bidding on this article,

Title	Facebook	Apple	dominos	mcdonalds	Hilton	Marriott
Facebook accused of striking 'secret deals over user data'	67	12	62	48	24	25
Woman says Hilton employee secretly filmed her naked in hotel room and tried to extort her	11	16	58	39	67	3
Airlines cleared for mobile phone use during flights	44	8	61	33	57	66
800-Pound Man Kicked Out of Hospital for Ordering Pizza	2	17	49	15	42	61
Obesity rates now top 35 percent in 7 states	26	17	66	67	39	53

Figure A.3: APNEA rankings on the Sample Articles, with Targeted Sentiment and Blacklist.

thereby preventing an unfortunate placement.