Semantical search term clustering for performance prediction

Coenders, R.

Award date:
2019

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Data Mining Group

Semantical Search Term Clustering for Performance Prediction

Master Thesis

Rik Coenders

Supervisors:
Dr. Ekaterini Ioannou
Dr. Nikolay Yakovets
Mark van Werven, MSc

Final version

Eindhoven, August 2019

Abstract

Search engine advertisers are responsible for creating and maintaining advertisements that will be shown in a search engine. They do so with a specific marketing goal, such as maximizing the return of investment on these advertisements. The 'slots' in a search engine that are available for advertisement are sold in an auction. As such, one of the main challanges in search engine advertising is to optimize bids for each specific search term. Many search terms only have a few impressions, and, because of that, performance metrics for a single search cannot be determined very reliable. This makes it hard to determine the optimal bid, which causes the advertisement goals not to be reached. In this thesis, an approach is introduced to get a set of historical search terms that are semantically similar to a specific search term, such that the expected values of the performance metrics of the search term can be predicted by aggregating over the metrics of the similar search terms. The approach is based on clustering search terms together that have a high similarity. The similarity between two search terms will be calculated using a pre-trained word vector model.

Preface

This research was done within the Data Mining Group of the Department of Mathematics and Computer Science at the Eindhoven University of Technology. It was supervised by Dr. Ekaterini Ioannou, who is an assistant professor in this group.

The project was done in co-operation with ADchieve. ADchieve is founded in 2005 to fill in the need for automation in the, until then, very labor intensive market of search engine advertising. The company focusses on automation of advertisement campaign creation and management. This is done mostly for, but not limited to, search advertisements in Google. ADchieve is a Premium Google Partner and a Google Ads API partner.

ADchieve has offices in Den Bosch and Rotterdam, both in The Netherlands. Both locations together host around 30 employees (as of August 2019). At ADchieve, people are working with a background in software engineering, data science, econometrics and search engine advertising.

In recent year ADchieve shifted it's focus from pure creation of advertisements to a more data-driven decision making approach, which includes coming up with smart marketing and econometric models for customers, satisfying their specific needs.

The project is done as master graduation project. The project started at February 4th, 2019 and was finished on August 28th, 2019. A total of 116 8 hour work days were spend on the project, giving 928 hours in total.

I would like to thank my supervisor Dr. Ekaterini Ioannou for guiding me during the project. She helped me by picking the problem statement and guided me in the right direction during the project. Also, I would like to thank Mark van Werven, CEO of ADchieve, for making it possible to do the project at ADchieve and making the resources available to do so. In addition, I would like to thank my colleages at ADchieve for providing the necessary information about search enginge advertising and marketing automation in general and for providing feedback on the project and this report.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Nowdays, search engines are the main way of finding content on the web. A user enters a search term into the search engine and the search engine shows the most relevant web pages. The most popular search engine at the moment of writing is Google, with a global market share of 92.81% in April 2019 [2]. A significant portion of searches are made for finding products to buy. This thesis focusses on such searches.

Website owners typically run advertisement campaigns in a search engine like Google. For this service, they pay the search engine per click on an advertisement. An auction model is used where advertisers can bid on specific keywords to determine for which searches their ad will be shown. The placement of advertisements in a search engine is called search engine advertising (SEA).

A whole industry exists around search engine advertising and internet markering in general. Companies in the SEA industry try to optimize for their clients which advertisements should be shown and for what searches. This is done by analysing performance and running experiments to test hypotheses. There can be different goals within search engine advertising, such as increasing revenue, profit, brand awareness or market share.

Profit margins in the online retail business are often small, so it is important not to bid too much on keywords that do not generate a lot of revenue. But bidding too low will cause an ad not to be shown, giving a drop in visitors to the advertisers website.

Companies such as ADchieve use smart algorithms and econometric models to determine the optimal bids. These algorithms are fed with historical performance data that can be extracted from the search engine. To optimize the algorithms, it is desirable to have as fine-grained data as possible by splitting up the keywords, so different search terms will match different keywords.

For example, for the search terms “buy television” and “compare televisions”, it is possible to create one keyword matching both or to create one keyword for each search term. In the latter case it is possible to set a different bid for both keywords, whereas in the fist case only one bid can be set. The different search terms can have different optimal bids, so it is benificial to split up the keywords and to choose a different bid for each one. But, by doing this, another problem arises. On these keywords that match very specific search terms, there is hardly any search traffic. Many search terms only have a few impressions (the number of times an ad is shown for this search term). Because of that, performance metrics for a single search term cannot be determined very reliable. This makes it impossible to depend on these performance metrics for determining the optimal bid.

The problem that will be addressed in this thesis is this problem of predicting the performance metrics for a single search term, by using data from related search terms. The challenges that arise for doing so are how to determine which search terms are similar and how many similar search terms should be taken into account for predicting the performance metrics. Also, since a lot of search terms are available, the calculation should be done in an efficient manner, because otherwise the calculation costs in terms of time and resources would become to high.

Contribution

In this thesis, an approach is proposed to predict the performance metrics of a single search term, by taking similar search terms into account. This is done by transforming each word into an embedding vector. These word embeddings capture the semantical properties of a word and can be used to calculate the similarity between two words. The semantical similarity between two search terms is calculated by using the soft cosine similarity between the search terms. This similarity measure is used to to create clusters of semantically similar search terms, so similar search terms can be queried quickly. These clusters might overlap.

Data from the related search terms is used to get a more reliable predication of the performance metrics of a single search term, than would be the case when using only data of the single search term itself.

This thesis is focused on English search terms, although the same approach might be used for other languages as well with some minor modifications.

Organization

In Chapter 2 the advertisement model of a search engine like Google will be discussed, as well as the data that can be extracted from the search engine. The chapter also contains a formal definition of the problem of predicting performance metrics based on similar search terms.

Chapter 3 will cover work related to search engine advertising, similarity measures and scalable clustering techniques. For these works, a brief description will be given of their contents. Also, it will be stated how they relate to the problem of performance metric prediction and why these works can be used in solving this problem or why not.

A method for predicting performance metrics is described in Chapter 4. This chapter is split up into several subproblems that will be discussed one by one. These subproblems are the preprocessing of the data, picking a similarity measure, using this similarity measure for creating a clustering, and using this clustering for the prediction of performance metrics.

The performance of this solution is evaluated in Chapter 5 on data from a customer of ADchieve. A description of the data set will be given as well as implementation choices and their consequences.

The final chapter, Chapter 6 contains a conclusion of the findings from the evaluation of the method and what needs to be done to bring the theory into practice. The strong points and the possible areas for improvement will also be discussed in that chapter.

Chapter 2

Problem Definition

As stated in Chapter 1, there exists a big industry around search engine advertisement. This chapter explains how search engine advertising works (Section 2.1) and goes into detail on what bid management is (Section 2.2), an important area of search engine advertisement. It explains why bidding on keyword level is desirable and the problem that arises when doing so. In Section 2.3 the available performance metrics will be discussed. These metrics are important for bid management. Section 2.4 will describe the data that is available about the products that are being sold, a data source that can be a very useful addition to the performance metrics.

In Table 2.5 at the end of the chapter, all notation that will be introduced in this chapter will be summarized.

2.1 Search Engine Advertising

Search engines are being used to find webpages about some topic. To find the desired web pages, a user enters a search term (or query) into the search engine and the search engine will respond with a set of relevant web pages. Search engines can make money by showing paid results next to their normal search results.

The description of the search engine in this thesis is mostly generic, but globally follows the principles of how the Google search engine works.

There are several kinds of results returned by the search engine, which are indicated in Figure 2.1. The search term that is entered by the user is "buy tv". This is indicated with `Q` in the figure. There are normal, unpaid search results, that are also called organic results. These results are marked with an `O` in the figure. Besides organic search results, the search engine also shows paid results, called advertisements. Google has multiple types of advertisements in their search results, including text advertisements (marked with a `T` in the figure) and shopping advertisements (marked with an `S` in the figure). This thesis only focusses on text advertisements, but the introduced approach can be extended to other advertisement types as well. When, from now on, advertisements are mentioned, text advertisements are meant.

An advertisement in Google consists of a heading text and a description. Some more fields can be shown in the advertisement such as the store location and additional links, but they will not be discussed here for brevity. Each advertisement has a destination URL. This is where a user will be directed to when clicking on the advertisement.

The advertisement positions can be bought by website owners. The people buying these advertisements are called advertisers. For advertising in Google Search, Google has the Google Ads platform, formerly known as AdWords. Advertisers can decide what the advertisment should look like in this platform.

Advertisers can indicate on what search queries they want to be shown with so called keywords. The search engine will show the advertisement when people search for this keyword. The entered search term does not have to match the keyword exactly. It is possible to indicate how broad the

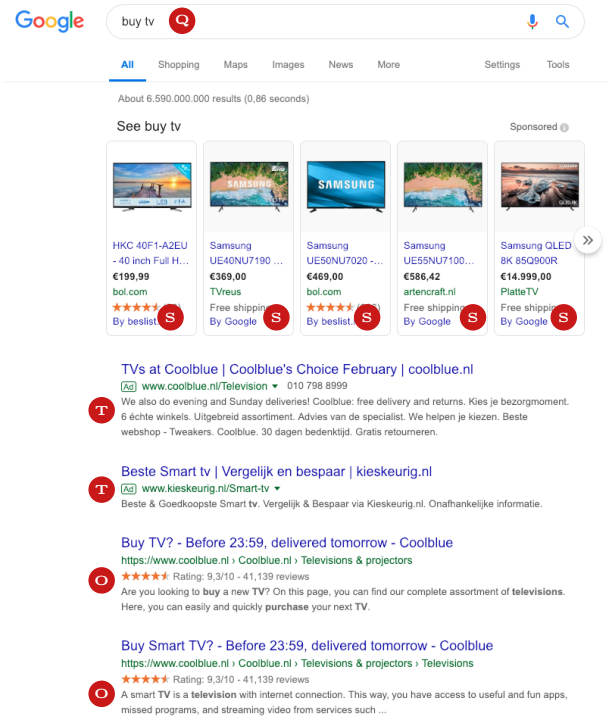


Figure 2.1: Example of the Google Search page, returned when searching with search term "buy tv"

matching of a keyword to a search term should be by using so-called match types. The match types used by Google are shown in Table 2.1[1] and a description and an example are given for each match type. Every keyword consists of a keyword text and a match type. The keyword text and match type together determine what search terms the keyword will match.

From now on, consider a single advertiser that wants his advertisements shown in the search engine. The advertiser pays the search engine per click on an advertisement. Let K denote the set of all keywords the avertiser wants to be found on. When the advertiser wants to advertise on a keyword $k \in K$, he can set a bid for this keyword. This is the maximum cost per click the advertiser is willing to pay. This bid is also referred to as the `maxCPC`. This bid for the keyword is the maximum cost that the advertiser needs to be pay for a single click on an advertisement that is shown because the search term matches the keyword k.

There can be multiple advertisers wanting to advertise for the same search term. To determine which advertisements are shown and at what position (the top position is considered the most valuable), a blind second price auction [8] is used. This is a form of a Vickrey auction [30]. In this kind of auction, the competitor advertisers will not see each other's bids. Every advertiser pays the price from the next bid lower than theirs. So, the highest bidder pays the second highest bid, hence the name second price auction. From every advertiser, at most one advertisement is shown, so for simplicity it is assumed that from every advertiser exactly one keyword matches the search term.

Let $X = \{x_1, x_2, \ldots\}$ be the set of all advertisers that want to advertise on search term s and let $b_1, b_2, \ldots$ be the bids of the advertisers $x_1, x_2, \ldots$ on the keywords corresponding to s.

There is a predefined number of n slots available for showing advertisements in the search results. This number of slots is determined by the search engine. The position the advertisement of advertiser x_i will get is denoted as r_i, with $1 \leq r_i \leq |X|$. If $r_i > n$, the advertisement will not

[1] Source: https://support.google.com/google-ads/answer/7478529?hl=en

Table 2.1: Google match types and examples matching the keyword 'women's hats'

Match type	**Description**	**Example search terms**
Broad match	Matches misspellings, synonyms, related searches, and other relevant variations.	buy ladies hats women's clothing
Broad match modifier	All the words (or close variations of those words) in any order. Additional words may appear before, after, or between these words.	women's scarves and hats winter hats for women
Phrase match	Matches of the phrase (or close variations of the phrase) with additional words before or after	blue women's hats buy women's hats women's hats on sale
Exact match	Exact matches of the term or close variations of that exact term with the same meaning.	women's hats ladies hats hats for women hats women

be shown.

Advertisements with the higher bids will get a smaller position, that is,

$$\forall_{x_i,x_j \in X} b_i > b_j \Rightarrow r_i < r_j$$

It is assumed that a smaller position is better for an advertiser.

The cost advertiser x_i has to pay if his advertisement is shown ($r_i \leq n$) is defined as follows:

$$\text{cost}_i = \begin{cases} b_j & \text{if } r_i < |X| \\ \text{minimum bid} & \text{if } r_i = |X| \end{cases}$$

where x_j is the advertiser with $r_j = r_i + 1$.

If $r_i > n$, then the advertisement of x_i is not shown and x_i does not have to pay at all. The costs only need to be paid in case of a click on the advertisement, not for every time the advertisement is shown.

Note. Google uses a slightly different model, where the ranking is not only based on the bid, but also on a quality score (how relevant Google considers the advertisement to be). These differences will be ingored for simplicity.

Example 1. Consider a scenario with four advertisers and with three available advertisement slots. The advertisers and their corresponding bids are shown in Figure 2.2. The advertiser x_3 had the highest bid so it will get the first slot. x_3 has to pay the bid of the advertiser ranked second, which is x_1. So the cost for x_3 will be the bid of x_1 which is 3. x_1, who was ranked second pays the bid of x_2 who was ranked third, and so on. x_4 does not end up in the top 3 advertisers and since there were only 3 slots availble the advertisement of x_4 will not be shown and their will also be no costs for x_4. Note again that the costs given in this example, only need to be paid when someone actually clicks on the advertisement.

The auction will be held again for every search, but the bid needs to be set beforehand by the advertiser, and not per search.

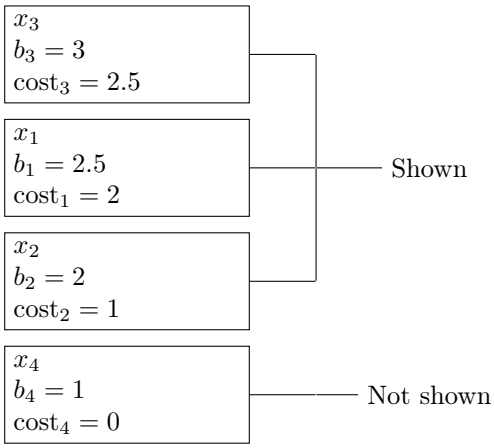


Figure 2.2: Example of the blind second price auction

2.2 Bid management

The process of determining the desired bid for a keyword is called bid management. Bid management can be done manually, but for a large number of keywords that is infeasable. A lot of factors can be taken into account when determining the desired bid, such as the product that is advertised for, the time of day and information about the user (e.g. whether this is a returning visitor or not).

What bid is desired depends on the goals of the advertiser. An advertister publishes an ad to get some value out of it, such as increased store purchases. The value an advertiser gets can be modelled as a function of random variables with the bid as parameter. The optimal bid is the bid where this function is maximized. The distibution of the random variables in the function depends on the context, such as the keyword, the advertisement, the competitors, the product being sold, et cetera. The historical performance measures, such as what percentage of people order when they have clicked on an advertisement, can be an important indicator for estimating the outcome of the function.

Since every keyword has its own (unknown) optimal bid, it is desirable to calculate the optimal bid for every keyword specific. Because data on a single keyword is often sparse and some keywords do not have historical data yet, calculating the optimal bid based on only historical data from the single keyword is hard. The goal of this thesis is to create a set of search terms similar to a specific search term and calculate aggregated metrics, such that the aggregate metrics of the similar set are as close as possible to the real (unknown) metrics of the search term. In this way, it is possible to estimate optimal keyword specific bids for keywords with no or little historical data.

2.3 Performance metrics

In this section, the available performance metrics will be explained. Performance metrics are an important factor for determining the optimal bid, because the performance metrics can give an estimation of how much money an advertiser will get out of the advertisement and how much the advertisement will cost him. With this information the advertiser can determine what bid is most benifical.

Definition 1. S is the set of historical search terms for all adverisements of the advertiser.

For every search term, there are historical performance metrics that can be extracted from the search engine. Only aggregdated data per search term can be extracted. Click stream data, that contains a record for every single impression including user data, is not available. The retrieved data contains several performance metrics, of which the ones shown in Table 2.2 are the most

Table 2.2: Available performance metrics

Metric	Short name	Description
Impressions	impr	Number of times the ad was shown
Clicks	clicks	Number of clicks on the ad
Click through rate	ctr	Percentage of impressions that got a click
Cost	cost	Total cost for the ad
Cost per Click	cpc	Average cost per click (calculated by deviding the cost through the number of clicks)
Conversions	conv	Number of sales following a click on the ad (these are measured by the website and then imported into Google Ads)
Conversion Rate	cr	Number of conversions per click (calculated by deviding the number of conversions through the number of clicks)
Conversion Value	cv	Total revenue coming from the conversions
Value per Conversion	vpc	Average value per conversion
Cost per Conversion	cpa	Average cost per conversion
Return on Ad Spend	roas	Conversion value divided by the cost

Table 2.3: Example search term performance data

Search Term	impr	clicks	ctr	cost	cpc	conv	cr	cv	vpc	cpa	roas
best tv	5	2	0.40	0.53	0.27	0	0.00	0.00	0.00	inf	0.00
android tv	10	1	0.10	0.17	0.17	0	0.00	0.00	0.00	inf	0.00
soundbar	25	7	0.28	2.73	0.39	1	0.33	199.95	199.95	2.73	73.24
soundsbar	8	1	0.13	0.13	0.00	0	0.00	0.00	0.00	inf	0.00

important ones. The table gives the name of the metric in combination with a "short name" that will be used to reference to the metric in this thesis. Also a description of the metric is given. One of the performance metrics described are the conversions. Conversions are some form of action the user performed after clicking on an advertisement, such as a purchase. The advertiser tracks this conversion and sends it to the search engine. The search engine connects the conversion to the corresponding advertisement. The advertiser can also specify a value for the conversion, which will be the conversion value. The conversion value typically is the amount of money the purchase is worth.

Definition 2. The random variables $\phi_{\text{impr}}(s), \phi_{\text{clicks}}(s), \ldots$ indicate performance of search term $s \in S$. The observerd historical values for these metrics are denoted $\bar{\phi}_{\text{impr}}(s), \bar{\phi}_{\text{clicks}}(s), \ldots$, with impr and clicks the metrics corresponding to the short name in Table 2.2.

To refer to a performance metric in general, the notation $\phi_{\ldots}(s)$ is used.

Table 2.3 contains example values for some search terms, with numbers that are comparable with a typical customer of ADchieve. Lots of rows have few impressions, and mostly no conversions.

A separation of the performance measures shown in Table 2.2 can be made into two categories: counter performance metrics and derived performance metrics. Counter performance metrics are only increasing over time. Derived performance metrics can be calculated based on the counter performance metrics. In Table 2.2, the metrics 'impressions', 'clicks', 'cost', 'conversions' and 'conversion value' are counter metrics. The other metrics in this table are derived metrics that can be calculated from the mentioned counter metrics.

The derived performance metrics say something about the result of a single search, and therefore are the one that are most relevant for optimal bid calculation. This paper will therefore only focus on predicting derived performance metrics. Derived performance metric prediction can be done by applying the calculation of the derived metric to predictors of the counter metrics. It is

not necessary that these counter performance metrics are predicted accurately to get a prediction for the derived performance metrics, as long as the counter metrics have the same scalar derivation from the real expected value, because the scalar derivation will cancel out in the calculation of the derived performance metric.

2.4 Product data

Often advertisements are based on a specific product or a group of products, for example an advertisement for a brand. Data about the products can be an enrichment to the available search term data. For example brands are often occuring in search terms and therefore being able to identify these words as a brand can be benificial for determining similarity between search terms.

An example product with the product properties is shown in Table 2.4[2].

The set of products that will be offered by the advertiser will be denoted P, with $P = \{p_1, p_2, \ldots\}$. For these products, a number of properties are available. $Z = \{z_1, z_2, \ldots\}$ is the set of product properties. $\pi_z(p)$ will denote the value of property $z \in Z$ for product $p \in P$.

[2]Source: https://support.google.com/merchants/answer/7052112?hl=en

Table 2.4: Example product values

Field	**Example Value**
Id	A2B4
Content Language	en
Target Country	US
Title	Mens Pique Polo Shirt
Description	Made from 100% organic cotton, this classic red men's polo has a slim fit and signature logo embroidered on the left chest. Machine wash cold; imported.
Product Category	Apparel & Accessories > Clothing > Outerwear > Coats & Jackets
Link	http://www.example.com/asp/sp.asp?cat=12&id=1030
Image Link	http://www.example.com/image1.jpg
Condition	new
Availability	in stock
Price	15.00 USD
Brand	Google
GTIN	3234567890126
MPN	GO12345OOGLE
Color	Black

Table 2.5: Used notation in problem definition

Notation	**Description**
K	Set of keywords from the advertiser
k	Keyword
s	Search term
X	Set of advertisers that want to advertise on a specific search term
x	Advertiser
b_i	Bid of advertiser x_i
n	Number of advertisement slots availble in the auction
r_i	Advertisement position in the search results for advertiser x_i
cost_i	The price advertiser x_i has to pay per click on his advertisement
S	Set of historical search terms
$\phi_{...}(s)$	Performance of search term $s \in S$
$\phi_{...}(s)$	Historical performance search term $s \in S$
$\hat{\phi}_{...}(s)$	Estimated performance search term $s \in S$
P	Set of products from the advertiser
p	Product
Z	Set of product properties
z	Product property
$\pi_z(p)$	Value of property z for product p

Chapter 3

Related work

A lot of research has already been done to bid management in search engine advertising. In Section 3.1 some works that are relevant to this area will be discussed. More specific to the approach chosen in this thesis are the sections following thereafter. Section 3.2, is about works related to search term similarity. For the clustering approach in this thesis, it is necessary to have a similarity measure between two search terms. At last, processing at scale will be discussed (Section 3.3). With the amount of search terms available, simple clustering approaches would be to computationally expensive, so a clustering approach optimized for large scale data is required.

For all related work advantages and disadvantages and their applicability to the problem definition will be given.

3.1 Search Engine Advertising

There are multiple types of online advertising auction systems. Perlich et al. [25] describe a method for determining an optimal bid in a real time bidding (RTB) auction. In an RTB auction, a bid is placed at the moment a user visits a webpage. In their paper, Perlich et al. describe that for an individual user, data is gathered and analysed. When the user is considered relevant, it will be put in a pool of users on which advertising is desired. An opportunity score is calculated for every targeted user for a specific advertisement. The opportunity could be the conversion rate or something else. In the opportunity score, the site on which the ad will be shown also is taken into account. On this opportunity score, the bid is based. A base bid is modified by a factor depending how the opportunity score differs from some base score. Logistic regression is used to predict the opportunity score.

The extra user information that is taken into account can be very useful, but it is not possible to do this in the auction model used for the Google search engine, because the bid is not placed real time for a single user, but beforehand.

Kitts and Leblanc [17] present a mechanism that calculates a bid for each keyword. A different bid is placed on different times of the day. It estimates the number of clicks per time unit based on the keyword, time and position. Data from surrounding time slots is taken into account by applying a weight. An estimation is made of which position would be received with what bid. Then the revenue is estimated. The bid is determined by exploring the bidding landscape, instead of taking the optimum directly. A tradeoff between exploration and exploitation is made when choosing the bid. Semantics of search terms are not taken into account. So, this does not work well for long tail keywords, because they have only little amount of data. However, not including long tail keywords would result in missing an import part of the conversions [4]. Empirical analysis on the data of some of the customers of ADchieve shows this same importance.

Ghose and Yang [9] descibe a hierarchical Bayesian modeling framework to estimate the click trough rate and the conversion rate of an advertisement, based on the rank, brand, retailer,

keyword length and landing page quality. It does, however not take into account the semantical value of the search terms, so, no difference will be made between very similar terms and dissimilar terms.

Graepel et al. [11] propose a probit regression model for predicting the click trough rate of an advertisement. The model predicts wheter there will be a click for an impression based on the features for this impression. The input is a sparse binary vector. Features like campaign, structure, info from the search term, and ad, product and landing page features are being used. These vectors are fed into a factor graph on which the regression is done. Using the campaign structure as feature depends on the knowledge of the marketeer instead of the model, this can be solved by including the marketeers decisions as features instead of the campaign structure. How the features should be engineered is not described by the authors.

Rutz and Bucklin [26] propose a logistic model that does take into account keyword features. These manually chosen features try to represent the textual properties of the keyword. In this way they try to solve the problem of wanting to know data on keyword level, but with sparse info on keyword level. They transform the keyword into a feature vector. These features have to be determined manually. Then the keyword performance is estimated by applying logistic regression. Having to choose the vectors manually could be a problem because latent factors can be missed. Also it can take a lot of time to engineer the features.

Abhishek and Hosanagar [3] propose a method in which a a set of suggested keywords is generated with maximal aggregate profit, not exceeding a given advertising budget. A corpus is created by crawling the website of the merchant, and crawling additional websites containing these keywords recursively. Words are then filtered to keep the ones with the highest td-idf scores of the document. A dictionary is created from these terms. All terms are submitted to a search engine to retrieve relevant documents. The returned documents create a context vector and the context vectors of two keywords are compared to get their similarity. Generated keywords are kewords that are similar to an existing keyword. The problem with taking data from semantically similar search terms can be solved in this way, but the test from the paper is on a really small scale, and would not scale well to the size that is needed.

Shariat et al. [28] propose a method to evaluate bid prediction models. They suggest methods to forming groups for A/B testing. They try to measure the difference in return of investment (ROI) and show methods to do so. Having some measure to test different strategies can help in picking the best strategy not on paper but in real-world.

The approach im this thesis will not determine the bids itself, but only provide data that can be used in bidding algorithms, so a difference in ROI can come the proposed model or from how the model is used to determine bids. Thus, an A/B-test is not useful on itself to test the performance of the proposed method, but can be useful testing the overall impact of the method in combination with a bidding algorithm.

3.2 Semantical similarity between search terms

There exist various metrics for measuring similarity between terms. [10], [22] and [14] provide an overview of different types of approaches for measuring term similarity. Text similarity measures can be divided into several categories.

Character based similarity metrics use the actual characters within a word and compare them in some way. These methods work best for comparing relative short strings such as words or names. Examples of character based similarity metrics are the Longest Common Subsequence (LCS) which bases the similarity on the length of the longest substring that occurs in both strings and the Levenshtein distance [19] that calculates the number of operations (add, remove or replace characters) that need to be done to convert one string into the other. Another method is splitting the string into n-grams and counting the amount of n-grams that are similar between the strings.

Another category of string similarity metrics are the term based similarity metrics that compare a whole term, where the term is replaced by a vector of word counts and using a distance metric on these vectors. Examples are the manhattan distance, euclidean distance, cosine distance, Jaccard similarity [15] and dice coefficient [7]. TF-IDF [27] can be used instead of simple word counts to reduce the impact of unimportant words that occur a lot such as 'the' and 'I'.

Character and term based similarity metrics do not take into account semantical similarity between words, only textual similarity. With corpus-based similarity metrics, semantic similarity is trained first on a large corpus. Examples are Hyperspace Analogue to Language [20] and Latent Semantic Analysis [18]. The latter one assumes there is a relation between the words that occur close to eachother. By creating vectors of counts of words close by a relation can be found between words that have similar words close by.

Knowledge based similarity metrics use information from semantic networks like WordNet (`https://wordnet.princeton.edu/`). Knowledge based similarity metrics have as problem that a semantic network needs to be available with the relevant words in the right language.

Hybrid methods combine other types of similarity methods together to obtain a better similarity measure.

Another technique is focussing on groups of data, so multiple strings that are about the same entity, for example a person has a first name, a last name, a nationality, a gender, etcetera. These techniques are not suitable for the problem adressed in this thesis beceause only a single string search term is given.

Collective similarity techniques do not only look at two terms, but take inner relations between terms into account. [5] proposes a distance measure for use in clustering of linked data. In [6] this distance measure is used to propose a strategy of finding links in combination with another similarity measure such as string matching, that still has a factor of uncertainty. For example, the common last name Smith could relate to the same person or to a different one. This problem is not solvable with string matching techniques alone. A clustering is updated in an iterative fashion. Every iteration the duplicates are updated. Duplicate detection is done by checking if the similarity is above a given threshold.

3.3 Scalable processing

Clustering on large data sets can give performance issues and even may be infeasable. McCallum et al. [21] introduce a technique for clustering large data sets with lots of features and lots of clusters, by first dividing into overlapping subsets, called canopies. They do so by using an approximate distance measure that is relatively cheap. Clustering is then done with the exact similarity metric, but only calculates similarity of points that share at least one canopy. In this way, the exact, but computationally more expensive similarity measure only needs to be calculated on a limited amount of pairs.

Chapter 4

Method

In this chapter a method is proposed to estimate the performance metrics of a single search term by using data from semantically similar search terms. The method makes use of historical search terms that are retrieved from the search engine, including the historical performance measures. Before processing, the search terms are first replaced with a normalized version. This normalization steps makes sure that variations in search terms, that a human would consider equal, are also seen as one search term.

To create the estimation, possibly overlapping clusters of search terms are created. This clustering is done based on a similarity measure between two search terms, capturing the semantical similarity of the two. For the clustering, first centroids are determined such that the centroids are not to close to eachother but every search term is close to at least one centroid. Search terms within a specified range from a centroid are added to a cluster with this centroid. The clusters might overlap.

The performance metrics $\phi_{\dots}(s)$ for a search term s can then be estimated by aggregating data from the clusters that a search term belongs to and clusters that are close by. An algorithm is proposed in this chapter to travel from one cluster into other clusters with high similarity until enough data is gathered.

In the following sections first the normalization process will be discussed (Section 4.1). Then, the similarity measure will be explained (Section 4.2). Following, it will be explained how this similarity measure is used to create clusters (Section 4.4). Finally, it will be explained how these clusters can help to get a prediction for the performance metrics (Section 4.4).

In Table 4.6 at the end of the chapter, all notation that will be introduced in this chapter will be summarized.

4.1 Search term normalization

Normalization starts with the 'raw' search terms that were entered into the search engine. The result of the normalization should be a sequence of words for each search term, which will be called the normalized search term. The normalization of s will be denoted $\bar{s}$ and $\omega(\bar{s})$ will denote the sequence of words of $\bar{s}$. These words should only contain characters that are considered relevant. Normalization is done to make further processing easier, because there will be less variations of the same word.

The following normalization steps are proposed:

- Remove control characters (such as `U+0000 NULL` and `U+0007 BELL`). These are not human readable and probably not entered on purpose by the user.
- Replace characters by their simplified (NFKC) unicode form, this makes sure that characters that look exactly the same, are the same. An example would be the characters `U+0031 DIGIT ONE` and `U+2081 SUBSCRIPT ONE`.

- Remove diacritcs and replace characters by their lowercase variants, because some people will type characters while others might not, while the same thing is meant.
- Strip whitespace characters from the beginning and the end of the search term and split to words on one or more whitespace characters.

4.2 Search term similarity

To determine how similar two search terms $\bar{s}_1, \bar{s}_2$ are, a similarity measure is proposed in this section.

Definition 3. The function $\text{sim}(\bar{s}_1, \bar{s}_2)$ gives the similarity between the search terms $\bar{s}_1$ and $\bar{s}_2$, with $\text{sim}(\bar{s}_1, \bar{s}_2) \in [0, 1]$

$\text{sim}(\bar{s}_1, \bar{s}_2) = 1$ means the search terms are completely similar and $\text{sim}(\bar{s}_1, \bar{s}_2) = 0$ means the search terms are completly dissimilar. The function sim is commutative. That is, $\text{sim}(\bar{s}_1, \bar{s}_2) = \text{sim}(\bar{s}_2, \bar{s}_1)$.

4.2.1 Word similarity

To simplify the problem of finding a similarity measure, the problem is split into creating a word similarity measure and using this word similarity measure to create a term similarity measure. In this section, the word similarity measure will be proposed. Words should be seen as very similar when they have similar meaning (they are synonyms, e.g., 'bike' and 'cycle') or when one is a misspelling of the other. Also, two words could be another form, like the verbs 'eats' and 'eating'. Or, the words can have some relation, such as 'bike' and 'car' (both means of transportation). Also words with similar patterns should be considered similar (e.g., serial numbers).

Definition 4. The function $\text{wordsim}(w_1, w_2)$ gives the similarity between words $w_1, w_2 \in W$

A popular approach is to map words to a so called word embedding [1]. This is a vector that represents the word.

Definition 5. The word embedding of a word $w \in W$ is represented by the vector v_w.

One of the most popular methods nowdays for word embedding is word2vec [23]. Word2vec consists of two approaches to train a word embedding using a neural network, named continous bag of words (CBOW) and skip-gram. The input and output of these models are one-hot encoded vectors of words. A one-hot encoded vector is a vector in which one of the values is 1 and the other values are 0. For the word w_i, the i-th value is 1.

The CBOW model takes surrounding words in a sentence as input and tries to predict the missing word. All input vectors go through the same matrix M. The sum of these vectors gives a hidden vector. This hidden vector goes throug a second matrix M'. On the resulting vector a softmax layer is applied to give a output vector containing word probabilities.

The skip-gram model works in the opposite way. It takes a single word as input and tries to predict the surrounding words. All surrounding words get the same predicted probabilities, keeping the amount of parameters of the neural network limited.

The neural network is only used for training, the rows of the matrix M represent the word vectors.

Figure 4.1 and Figure 4.2 give a schematic overview of the neural networks for the CBOW and skip-gram model that are described above.

The weight matrix of the skip-gram model contains the number of words times the embedding length parameters. Updating all paramaters during training every time can be very slow. To optimize training of the skip-gram model, negative sampling [24] can be used. With negative sampling, only part of the words not in the context are taken into account, instead of all. This drastically reduces the number of parameters that are modified.

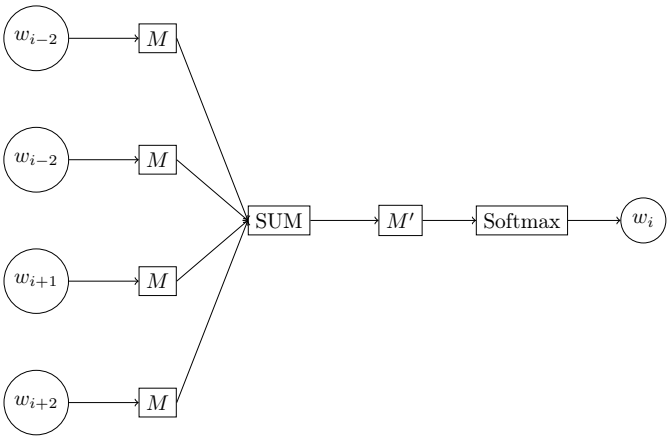


Figure 4.1: Continous Bag of Words (CBOW) model

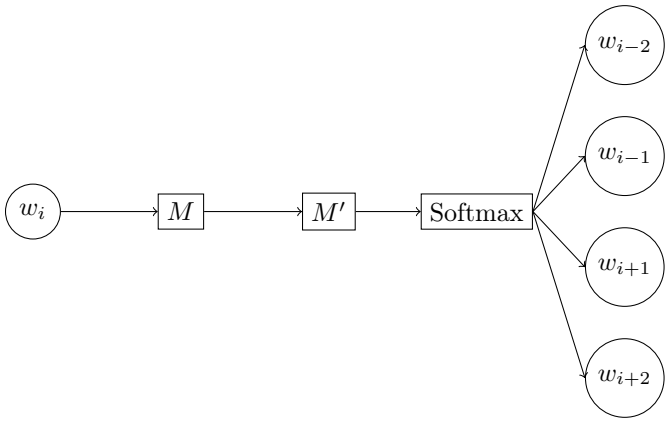


Figure 4.2: Skip-gram model

The word2vec model only uses whole words. This means it cannot handle words that it has not seen before, and handles rarely seen words poorly. Search terms often contain rare words. Joulin et al. [16] solve this problem in a model they call FastText.

FastText works in a similar way as the skip-gram model of word2vec, but instead of only taking a single word as input, n-gram substrings of the word are also taken as input. In this way, every n-gram gets it's own embedding. So, unseen words can also be vectorized by using the embeddings of it's n-grams. In case of an unseen word the average of the vectors for the seen n-grams of the word is used as the word vector.

The FastText model can be trained on a large corpus like Wikipedia. So, way more data can be used for training then only the available search term data.

In the trained FastText model, the properties of the products, such as the brand names and product categories do not occur as words. However, these words occur frequently in the search terms, so it is desirable to take them into account in some way. This can be done by adding vectors for the words in the set of properties that are considered relevant. These properties will be denoted Z_{attr}. The vectors will be based on data drom other textual properties, such as the titles and descriptions, that will be denoted Z_{txt}. The vector of a word w is calculated by taking the average of the term vectors of the products where w occurs in any of the properties of Z_{attr}. This gives the following updated word vector for word w:

$$v_i' = \begin{cases} \frac{1}{|P_{\text{rel}}(w)| \cdot |Z_{\text{txt}}|} \sum_{p \in P_{\text{rel}}(w)} \sum_{z \in Z_{\text{txt}}} \text{tv}(\pi_{p,z}(w)) & \text{if } P_{\text{rel}}(w) \neq \varnothing \\ v_i & \text{otherwise} \end{cases}$$

where $P_{\text{rel}}(w)$ are the products that have this word w as property

$$P_{\text{rel}}(w) = \{p | p \in P, \exists_{z \in Z_{\text{attr}}} \pi(p, z) = w\}$$

and the term vector $\text{tv}(\bar{s})$ is the average of the word vectors of the words in a term

$$\text{tv}(\bar{s}) = \frac{1}{|\omega(\bar{s})|} \sum_{w \in \omega(\bar{s})} v_w$$

The similarity between words can be described by a function over its embeddings. FastText uses the cosine similarity by default, this gives the following function for the word similarity:

$$\text{wordsim}(w_1, w_2) = \frac{v'_{w_1} \cdot v'_{w_2}}{\left\|v'_{w_1}\right\| \left\|v'_{w_2}\right\|}$$

4.2.2 Soft cosine similarity

The similarity between two search terms can be determined with the soft cosine similarity [29]. The soft cosine similarity works like the ordinary cosine similarity but adds a similarity score between words. Using the soft cosine similarity is benifical over just using the normal cosine similarity. With the soft cosine similarity, words that are different, but semantically similar, are also seen as similar. Whereas, with the normal cosine similarity, these would be seen as completely different.

Let W denote the set of all words occuring in any normalized search term. The soft cosine similarity is then defined as follows:

$$\text{sim}(\bar{s}_1, \bar{s}_2) = \frac{\text{sim}'(\bar{s}_1, \bar{s}_2)}{\sqrt{\text{sim}'(\bar{s}_1, \bar{s}_1)}\sqrt{\text{sim}'(\bar{s}_2, \bar{s}_2)}}$$

with

$$\text{sim}'(\bar{s}_1, \bar{s}_2) = \sum_{w_i, w_j \in W} \text{wordsim}(w_i, w_j) \cdot \gamma(\bar{s}_1, w_i) \cdot \gamma(\bar{s}_2, w_j)$$

where $\gamma(\bar{s}, w)$ is the relevance of word $w \in W$ within $\bar{s}$. For γ, the count of a word for can be used (or formally: $\gamma(\bar{s}, w) = \sum_{w' \in \omega(\bar{s})} 1[w' = w]$). Another possibility would be to use TF-IDF, but this is expected not to work well because words that are made unimportant for TF-IDF could be important for search term similarity.

4.3 Search term clustering

With the method described in the section before, the similarity of two search terms can be measured. This measure can now be used to propose a clustering approach that groups together semantically similar search terms. It consists of a method of choosing centroids that will form the base of a cluster, and a way to determine to what clusters a search term belongs. The approach is based on the clustering apporach by Ioannou and Garofalakis [13] that also solves the problem of creating canonical entities from multiple entities. So, the problem is very similar.

In this approach, clusters are created around centroids, based on the similarity of a search term with the centroids.

First the centroids need to be chosen. To reduce computation power required, only search terms with at least minimal number of impressions are considered to be taken as centroid. All search terms that fullfil this condition are randomly shuffled by weight, where the historical number of impressions are taken as weight. The higher the weight, the more likely a search term is to occur earlier in the shuffled search terms. The position of a search term s_i after shuffling is denoted q_i. For two search terms s_1 and s_2 it holds that

$$P[q_1 < q_2] = \frac{\bar{\phi}_{\text{impr}}(s_1)}{\bar{\phi}_{\text{impr}}(s_1) + \bar{\phi}_{\text{impr}}(s_2)}$$

At the start, the set of centroids is empty. Every search term is picked one by one from the shuffled search terms. If there is no centroid in the set of centroids with similarity with the search term of at least threshold τ_1, the search term is added to the set of centroids. Because of the weighted shuffling, search terms with more historical impressions are more likely to become a centroid. This is done because otherwise search terms with little impressions and with little information would be overly present in the set of centroids. The algorithm for picking the centroids is formalized in Algorithm 1. The set of centroids will be denoted C, with $C \subseteq S$.

Algorithm 1 Centroid selection

function GET_CENTROIDS(S, min_impressions) ▷ S: set of search terms
  candidates $\leftarrow \{s | s \in S, \phi_{\text{impr}(s)} > \text{min\_impressions}\}$
  $C \leftarrow \varnothing$
  while candidates $\neq \varnothing$ **do**
    Select a random c from candidates where each point $s \in$ candidates has probability $\frac{\phi_{\text{impr}}(s)}{\sum_{s' \in \text{candidates}} \phi_{\text{impr}}(s')}$ of being picked
    $C \leftarrow C \cup \{c\}$
    candidates $\leftarrow \{s | s \in \text{candidates}, \text{sim}(\bar{c}, \bar{s}) < \tau_1\}$
  end while
  return C
end function

A search term s belongs to the cluster of centroid $c \in C$ if s and c are similar enough, that is, $\text{sim}(\bar{s}, \bar{c}) > \tau_2$. The threshold τ_2 should be less than τ_1. This gives the following cluster with centoroid c:

$$\text{cluster}_c = \{s | s \in S, \text{sim}(\bar{s}, \bar{c}) < \tau_2\}$$

The overlapping area of the clusters with centroids $C' \in C$ is the set of search terms that are in all clusters of C', but not in any other cluster. This is formalized as:

$$\text{overlap}(C') = \{s | s \in S, \forall_{c \in C} c \in C' \Leftrightarrow s \in \text{cluster}_c\}$$

$O = \{C' | C' \subseteq C, \text{overlap}(C') \neq \varnothing\}$ is the set of all overlapping areas.

The shared properies of the overlapping areas will be used in Section 4.4 to predict the metrics of a single search term.

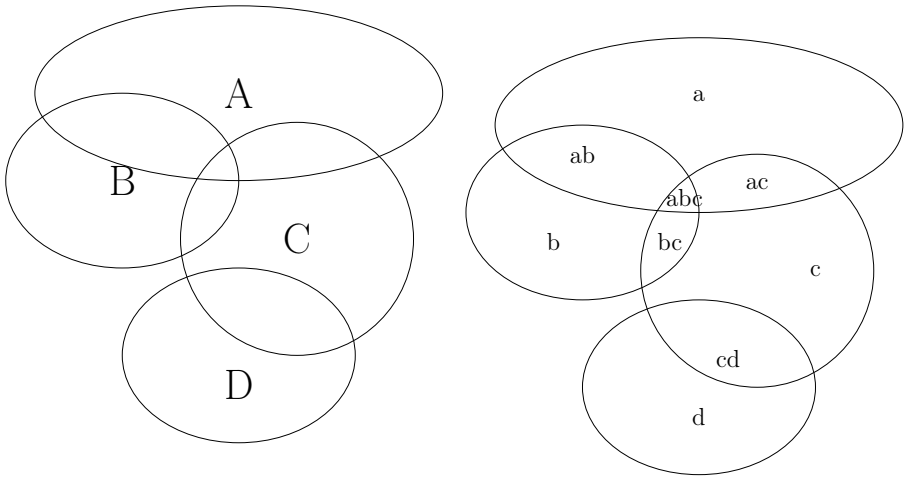
Table 4.1: Example centroids

Cluster	Centroid
A	blue bike
B	bike
C	mountain bike
D	mountain boots

Table 4.2: Example search terms and corresponding areas

Search term	In Cluster				Area
	A	B	C	D	
bike	No	Yes	Yes	No	bc
bicycle	No	Yes	No	No	b
blue bike	Yes	No	No	No	a
bike blue	Yes	Yes	No	No	ab
blue mountain bike	Yes	Yes	Yes	No	abc
blue mountain bikes	Yes	No	Yes	No	ac
mountain bike	No	Yes	Yes	No	bc
mountain bike helmet	No	No	Yes	No	c
mountain boots	No	No	No	Yes	d
mountain gear	No	No	Yes	Yes	cd

Example 2. Consider a scenatio with the 4 centroids shown in Table 4.1. All the centroids have a corresponding cluster. In Table 4.2 some example search terms are given and the clusters the search terms belong to based on their similarity with the centroids of the clusters. The search term 'blue mountain bike' is similar to the centroids 'blue bike' (cluster `A`), 'bike' (cluster `B`) and 'mountain bike' (cluster `C`). Because of this, the search term belongs to the area overlapping these three clusters, area `abc`. Figure 4.3a and Figure 4.3b show the overlap of the clusters corresponding to this example and the corresponding overlapping areas.



(a) Clusters (b) Overlapping areas

Figure 4.3: Example clustering and corresponding overlapping areas

4.4 Metric prediction

In this section, an alogrithm is introduced that can be used to predict the performance metrics of a single search term, based on the clusters from the previous section.

Algorithm 2 Similar search terms retrieval

function GET_SIMILAR_TERMS(s, goal)
  $C' \leftarrow \{c | c \in C, \text{sim}(\bar{s}, \bar{c}) > \tau_2\}$
  queue $\leftarrow \{o | o \in O, \exists_{c' \in o} c' \in C'\}$
  total $\leftarrow 0$
  visited $\leftarrow \varnothing$
  result $\leftarrow \varnothing$
  while queue $\neq \varnothing$ **do**
    $o \leftarrow \text{argmax}_{o' \in \text{queue}} \text{areasim}(s, o')$
    if $\frac{\text{total}}{\text{goal}} \leq \text{areasim}(s, o)$ **then**
      result $\leftarrow$ result $\cup \{(s', \text{areasim}(s, o)) | s' \in \text{overlap}(o)\}$
      total $\leftarrow$ total $+ \sum_{s \in \text{overlap}(o)} \bar{\phi}_{\text{impr}}(s)$
      visited $\leftarrow$ visited $\cup \{o\}$
      queue $\leftarrow$ queue $\setminus \{o\}$
      queue $\leftarrow$ queue $\cup (\{o' | o' \in O, \exists_{c' \in o'} c' \in o\} \setminus$ visited$)$
    else
      queue $\leftarrow \varnothing$
    end if
  end while
  return result
end function

Algorithm 2 gives search terms that are considered similar to a search term s. It starts with the areas of the clusters that s belongs to, and keeps expanding outwards until enough data is available to give a reliable prediction for the performance metrics. When to stop is determined by a goal value. The algorithm returns a set of similar search terms and corrseponding similarity. The algorithm makes use of a similarity measure between two overlapping areas, because calculating the similarity of a search term with all the search terms inside the area would be to computationally expensive. The area similarity takes the average of the similarities of the search term with all

Table 4.3: Example centroid similarity

Centroid	Similarity
A	0.95
B	0.75
C	0.5
D	0.3

centroids of the overlapping area and is formalized in the following equation:

$$\text{areasim}(s, C') = \frac{1}{|C'|} \sum_{c \in C'} \text{sim}(\bar{s}, \bar{c})$$

with $s \in S$ and $C' \subseteq C$

The working of the algorithm will be explained with the following example:

Example 3. Take the example clustering of Figure 4.3. Lets say, there is a search term 'red bike'. The similarity between 'red bike' and the clusters is given in Table 4.3. To get the similar terms of s with goal 100, first the clusters to which s belongs are determined. When τ_2 is assumed to be .75, this would be cluster `A` and `B`. The queue of areas to be considered is then filled with all areas overlapping cluster `A` or `B`, namely `a`, `b`, `ab`, `ac`, `bc` and `abc`. For each of these areas, the area similarity is calculated and the most similar area will be picked. Table 4.4 shows the results of this calculation, In this case, the most similar area is `a`, with a similarity of 0.95. The algorithm would go through the following steps:

- Take the area with the highest similarity: `a` (with similarity 0.95)
  Check if this area should be added. $0.95 \geq \frac{0}{100}$, so it should be added.
  Add the search terms from `a` to the result: 'blue bike'
  No new areas to add.
  The total becomes: $0 + 40 = 40$

- Take the area with the highest similarity: `ab` (with similarity 0.85)
  Check if this area should be added. $0.95 \geq \frac{40}{100}$, so it should be added.
  Add the search terms from `ab` to the result: 'bike blue'
  No new areas to add. The total becomes: $40 + 10 = 50$

- Take the area with the highest similarity: `b` (with similarity 0.75)
  Check if this area should be added. $0.95 \geq \frac{50}{100}$, so it should be added.
  Add the search terms from `b` to the result: 'bicycle'
  No new areas to add.
  The total becomes: $50 + 20 = 70$

- Take the area with the highest similarity: `abc` (with similarity 0.73)
  Check if this area should be added. $0.95 \geq \frac{70}{100}$, so it should be added.
  Add the search terms from `abc` to the result: 'blue mountain bike'
  Add areas from cluster C to the queue: `c`, `cd`
  The total becomes: $70 + 20 = 90$

- Take the area with the highest similarity: `ac` (with similarity 0.73)
  Check if this area should be added. $0.73 < \frac{70}{100}$, so it should not be added.

The resulting search terms with their relevant similarity are given in Table 4.5

Now, the count performance metrics can be calculated by taking the weighted sum of the perofmance metric for the similar search terms. The predicted performance metric will become:

Table 4.4: Example area similarity

Area	**Similarity**	**# of impressions**
a	0.95	40
b	0.75	20
c	0.50	30
d	0.30	20
ab	0.85	10
ac	0.73	15
bc	0.63	12
cd	0.40	16
abc	0.73	20

Table 4.5: Search terms similar to 'red bike'

Search term	**Area similarity**
blue bike	0.95
bike blue	0.85
bicycle	0.75
blue mountain bike	0.73

$$\hat{\phi}_{...}(s) = \frac{1}{\sum_{(s',\sigma)\in \text{similar\_terms}(s)} \sigma} \sum_{(s',\sigma)\in \text{similar\_terms}(s)} \sigma \cdot \bar{\phi}_{...}(s')$$

The predicted performance metrics can now be used existing bid management algorithms to determine a bid on keyword level.

Table 4.6: Used notation in method

Notation	Description
$\bar{s}$	Normalized version of search term s
$\text{sim}(\bar{s}_1, \bar{s}_2)$	Similarity between search terms $\bar{s}_1$ and $\bar{s}_2$
$\omega(\bar{s})$	The sequence of words in $\bar{s}$
$\text{wordsim}(w_1, w_2)$	Similarity between words w_1 and w_2
w	Word
W	Set of all words occuring in any normalized search term of S
v_w	Word vector of word w
Z_{attr}	Product properties that are considered to contain relevant words
Z_{txt}	Textual product properties that are used as a corpus
v'_w	Modified word vector of word w
$P_{\text{rel}}(w)$	Set of products having word w as property
$\text{tv}(s)$	Term vector of string s
$\gamma(\bar{s}, w)$	Relevance of word w in search term $\bar{s}$
τ_1	Threshold 1
τ_2	Threshold 2
q	Position of the search term after shuffling
C	Set of centroids
c	Centroid
$\bar{c}$	Normalized version of the centroid c
min_impressions	Minimal number of impressions needed for a search term to be abe to become a centroid
$\text{get\_centroids}(S, \text{min\_impressions})$	Algorithm to get the set of centroids based on the historical search terms S
cluster_c	Cluster with centroid c
C'	Subset of C that indicates an overlapping area
$\text{overlap}(C')$	Search terms in overlapping area C'
$\text{areasim}(s, C')$	Similarity between search term s and overlapping area C'
O	Set of overlapping areas
goal	Number of impression that should at least be gathered from relevant terms
$\text{get\_similar\_terms}(s, \text{goal})$	Algorithm to get the terms similar to s
$\hat{\phi}_{...}(s)$	Estimated value of $\phi_{...}(s)$
σ	Search term relevance

Chapter 5

Evaluation

In this chapter the procedure described in Chapter 4 will be evaluated and the results will be presented and discussed. This procedure was implemented using Python 3.7. The implementation makes heavy use of the packages `numpy` (`https://www.numpy.org/`) and `pandas` (`https://pandas.pydata.org/`). All evaluations were performed on a MacBook Pro (Retina, 15-inch, Mid 2015) in a Docker container with 8GB memory available.

First, the data set that will be used in the evaluations will be described (Section 5.1). A brief overview of the properties of the data will be given. The effects of applying the normalization rules from Section 4.1 will be shown. Then, the word and term similarity measures will be evaluated (Section 5.2). Some sample term similarities will be given as an indicator of how well the similarity measure performs. Futhermore, the clustering procedure will be evaluated (Section 5.3). The performance of different parameter values for the clustering will be compared and discussed. At last, the performance prediction will be discussed.

5.1 Data set

Data from of one of the customers of ADchieve is used. The selected customer is active in multiple countries. Only the advertisements of this customer targeting the USA will be used, to keep a uniform data set with only English search terms and elimination of possible issues with performance differences between countries.

A dataset containing all search terms and their corresponding performance metrics for the period May 2018 until April 2019 is extracted from Google Ads, using the `googleads` Python package (`https://github.com/googleads/googleads-python-lib`). In this dataset there are 15164 distinct search terms. A sample of the search terms in this dataset with the corresponding counter performance metrics is shown in Table 5.1. The data is sparse, as only 0.9% of the search terms has had a conversion, as can be seen in Table 5.2. This table shows the amount of zero and the amount of non-zero values for the counter performance metrics 'impressions', 'clicks' and 'conversions'. The counter metrics 'cost' and 'conversion value' are not shown because when a search has had a click, there are also costs and when the search term has had a conversion, there is also a positive conversion value. Because only search terms that were actually searched for can be retrieved, it makes sense that every search term in the data set has at least one impression.

The search terms are normalized according to the approach described in Section 4.1. After normalization, the number of distinct search terms decreases to 15142 distinct search terms, a decrease of 0.15%. So, nearly all search terms are already in the normalized form. For simplicity, search terms that after normalization have characters that are not in the Basic Latin and Latin-1 Supplement unicode blocks are removed from the data set. Doing so makes it easier to print and to debug the data set, and the search terms that are removed represent only 0.01% percent of the impressions, so removing them will not have a significant impact.

In Table 5.3 the impact of the seperate normalization rules is shown, by giving the number of

Table 5.1: Sample of search terms with historical metrics

Search term	impr	clicks	cost	conv	cv
tcx riding shoes	10	3	1.84	0.0	0.000000
forma vs tcx boots	1	1	0.60	0.0	0.000000
red helmet nexx xg 100	1	1	0.50	0.0	0.000000
scorpion exo covert helmet	211	10	4.55	0.0	0.000000
alpinestars motocross boots smx 3 size 43	2	2	0.82	0.0	0.000000
klim kodiak cerco	1	1	0.50	0.0	0.000000
airoh aviator 2.2	5748	440	200.57	8.0	3459.840088

Table 5.2: Amount of non-zero values per metric for search terms

	Zero	Zero (%)	Non zero	Non zero (%)
Impressions	0	0.0%	15164	100.0%
Clicks	122	0.8%	15042	99.2%
Conversions	15030	99.1%	134	0.9%

search terms that are modified by each specific normalization rule.

For evaluation, the data set is randomly split up into a training set and a test set. 20% of the records is put in the test set, the other values are put in the train set. This gives a train set of 13056 records and a test set of 3265 records. This percentage is chosen to keep as much records in the training set as possible while still having a test set of significant size.

Besides search term history, the current product data is also gathered. 8039 products are currently being sold. The amount of distinct values for the attributes 'brand', 'series' and 'category' are shown in Table 5.4. There are only a few brands and a few categories, so a lot of products share the same brand or category.

Table 5.3: Search terms affected by normalization rules

	#	%
Remove control characters	14	0.1%
Replace characters by simplified form	16	0.1%
Remove diacritics and change to lower case	41	0.3%
Strip whitespace	0	0.0%

Table 5.4: Amount of distinct values for product attributes

Brand	34
Series	437
Category	19

5.2 Similarity measures

In this section, word similarity will be evaluated. To embed the words, a pretrained FastText model is used [12]. By using a pre-trained model, the steps of gathering and preprocessing a corpus and of training the model can be skipped. The pre-trained model is trained on English texts from Common Crawl (https://commoncrawl.org/) and Wikipedia (https://www.wikipedia.org/). The model gives vectors of length 300 and uses n-grams of length 5, a window of size 5 and negative sampling of 10 words.

For the model to work, the words in this model should have the same normalization as the normalization used for the search terms. So these words are replaced by their normalized variants. In case two words have the same normalized variant, the first one occuring in the model is taken. The pretrained model has the most occurring words first, so in this way, the vector for the most occurring variant is used.

The words in the fields 'brand', 'series' and 'category' from the product data occur often in the search terms. Therefore, these words are updated in the model with a new vector that is obtained by calculating a term vector for the title and one for the description of each product. The title and description fields are the most relevant textual fields and their contents relate to the context of the brands, series and categories. The average of these two term vectors forms a vector for each product. The vector for one of the values of the fields is formed by taking the average of the product vectors where this value occurs. From the 490 words that occur in the brand, series or category of a product, 264 of them did not have a specified vector before. These words occur in 11786 search terms (77.8%), so they are very relevant words.

With the word vector model, the cosine similarities between words can be calculated. A sample of words with their most similar words is given in Table 5.6 and Table 5.5 contains a sample of word combinations and their similarity. In Table 5.5 it can be seen that for example words as 'roof' and 'shark' have a high similarity, which makes sense because these words are conceptually close. 'shark' is a series within the brand 'roof'. In Table 5.6 it can be seen that the words closest to for example '2019' are also years, so that also makes sense. The same holds for 'helmet' and 'visor', that have a part to whole relationship.

Now soft cosine similarity can be used to determine search term similarities. Just like with the words similarities, some tables are given to show the results (Table 5.8 and Table 5.7). In Table 5.7 it can be seen that slight variations of the search term such as 'airoh aviator 2.2' and 'airoh aviator

Table 5.5: Sample word similarity

word 1	word 2	similarity
roof	shark	0.889734
scorpion	held	0.822869
boxer	v8	0.338027
4-touring	sambia	0.779889
held	spector	0.884803
shoei	neotec	0.702001
g001.01xl	g001.04-ms	0.905213
roof	helmet	0.551451
helmet	t-shirt	0.460801

Table 5.6: Closest words (with at least 5 impressions)

value	match 1	similarity 1	match 2	similarity 2	match 3	similarity 3
roof	boxxer	0.980291	desmo	0.950107	helmets	0.940191
helmet	helment	0.830647	visor	0.737213	helments	0.725274
3	2	0.984587	4	0.982823	6	0.952771
usa	states	0.634237	america	0.610631	united	0.582304
2019	2018	0.905180	2016	0.576770	2015	0.575396
helmet	helment	0.830647	visor	0.737213	helments	0.725274
2.2	2.3	0.977052	2.1	0.956891	1.2	0.924909

Table 5.7: Sample search term similarity

term 1	term 2	similarity
cardo packtalk slim	cardo packtalk bold	0.817766
airoh aviator 2.2	airoh aviator 2.0	0.929796
nexx helmets	held helmets	0.582525
nexx helmets	held gloves	0.275382

2.0' are considered the most similar search terms. It can be seen in Table 5.7 that search terms that have little in common, such as 'nexx helmets' and 'held gloves' have a low similarity.

Table 5.8: Closest search terms (with at least 5 impressions)

value	match 1	similarity 1	match 2	similarity 2	match 3	similarity 3
klim gear	discount klim gear	0.635840	klim gear for sale	0.617685	klim clothing	0.602292
schuberth e1	schuberth e1 xl	0.834619	schuberth e1 ebay	0.819774	schuberth e1 cut	0.815966
shark modular helmet	shark modular helmets	0.980577	shark modular helmets	0.980577	shark modular	0.927996
klim kodiak	klim kodiak test	0.831905	klim kodiak jacket	0.777091	klim badlands pro vs kodiak	0.678276
tcx boots	tcx boot	0.892424	tcx boots closeout	0.869662	tcx boots closeout	0.869662

5.3 Clustering

In this section, the clustering method of Section 4.3 will be evaluated and the impact of the parameters of the clustering method on the result will be shown. Also in this section the estimation of performance metrics will be evaluated.

Centroids are picked from the search terms. Search terms with a higher number of impressions are more likely to be picked as centroid. Only search terms with at least 3 impressions are considered. This limits the amount of centroids that need to be taken into account. There are 4846 search terms with at least 3 impressions, which is 32.0% of the total. For different values for threshold τ_1, this gives the number of centroids shown in Figure 5.1.

It is desired to get a high similarity within a cluster, so the similarity threshold should not be too low. On the other hand, a higher threshold means less terms that are close enough to a centroid, so there will be more centroids. It is desired to not have too many centroids. These two are contradicting. To balance the two, for threshold τ_1 the value 0.8 is chosen. This gives 955 centroids. The average similarity between two centroids is 0.30 and the average similarity for the search terms with the closest centroid is 0.81.

Every centroid gets a corresponding cluster. Within these clusters lie all search terms that have a similarity of at least τ_2 with the centroid of the cluster. In Figure 5.2 the different minimum, maximum and average number of clusters a search term belongs to are given for different values of τ_2. A lower value for τ_2 means more words will belong to at least one cluster, which is positive because that means that for more words the relevant performance metrics can be calculated. On the other hand does a lower value of τ_2 means that more terms that might not be similar end up in the same cluster. So, the value of τ_2 should be balanced. The value 0.6 is chosen for τ_2, because at this value the average number of clusters has dropped to an acceptable number, and from that point, the avarage number more or less stays the same from that point. Taking a higher value of τ_2 would increase the number of search terms with no or few cluster, which would mean for these search terms nothing meaningful can be said.

A description of the sizes of the clusters and the number of impressions the search terms in a cluster have in total is shown in Table 5.9. The size of cluster varies greatly. The search term 'merlin hamlin kevlar hoody' is on its own in a cluster, whereas the centroid 'hjc helmets' has 4775 terms in the cluster.

The clusters give 9582 overlapping areas. On average, an overlapping area overlaps 33.9 clusters. The overlapping area overlapping the most clusters overlaps 315 clusters. See Table 5.10

Next, a goal value needs to be chosen. The results for different goals for 'shark race r pro carbon helmet' shown in Table 5.11. As goal the value 300 is chosen. A higher goal would lead to including data from less relevant terms and a lower goal would lead to including too little data to be significant.

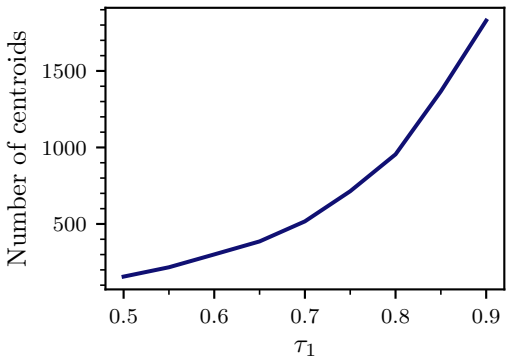


Figure 5.1: Number of centroids for different values of τ_1

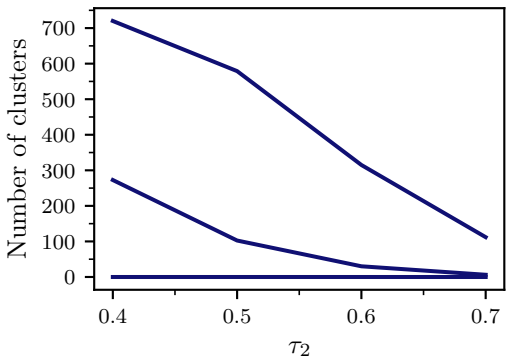


Figure 5.2: Min, max and average number of clusters a search term belongs to for different values of τ_2

Table 5.9: Cluster size info

	# Search terms	# Impressions
Minimum	1.000000	4.000000
Average	410.961257	25231.779058
Median	194.000000	7185.000000
Maximum	4775.000000	153662.000000

Table 5.10: Overlapping area info

	# Search terms	# Clusters
Minimum	1.000000	1.000000
Average	1.362555	33.867863
Median	1.000000	17.000000
Maximum	724.000000	315.000000

Table 5.11: Influence of goal on result

Goal	Number of search terms	Average similarity	Min similarity
100	46	0.783011	0.751495
200	68	0.770985	0.736153
300	79	0.765115	0.727521
500	79	0.765115	0.727521

Table 5.12: Amount of search terms from the test set for which an estimation can be made

	#	%
ctr	3031	92.8%
cpc	3031	92.8%
conversion_rate	933	28.6%
cpa	933	28.6%
roas	933	28.6%
value_per_conversion	933	28.6%

In the original data set there were lots of missing values (eg the conversion rate could not be calculated because there were no conversions). One of the goals of this project was to reduce this. By considering the test set as unseen search terms and running them through the prediction algorithm, it can be tested how many of the unseen search terms can be predicted with the subscribed method. The results of this test are shown in Table 5.12. It can be seen that for a significant portion of unseen search terms an estimation can now be made of their performance metrics.

Chapter 6

Conclusion

In Chapter 5 it was shown that word vectorization in combination with the cosine similarity measure can be an efficient method to determine the similarity between to search terms.

Besides the similarity measure, a clustering method was proposed to group the search terms into clusters based on this similiarity measure. This is done in an efficient manner that allows for large scale data sets to be used. For the search terms that end up together within a cluster it can be argued that they are similar, altough the similarity and quality of the clustering is a subjective thing.

With the proposed method for getting the estimation for a search term is possible to retrieve an estimation for a lot of terms for which this could not be done before. In that way, the proposed method solves the given problem. Whether the estimated performance metrics are actually better than the data known before and how much these values contribute into picking a better bid needs to be tested with a real life test such as an A/B-test, and is beyond the scope of this thesis.

In future work, the method proposed in this thesis can be extended with more data sources to make it more reliable. Also, the method could be combined with other, already existing methods, to get the benefits of both.

Bibliography

[1] A brief history of word embeddings (and some clarifications). https://www.linkedin.com/pulse/brief-history-word-embeddings-some-clarifications-magnus-sahlgren/. Accessed: 2019-05-24. 16

[2] Search engine market share worldwide. http://gs.statcounter.com/search-engine-market-share. Accessed: 2019-05-24. 1

[3] Vibhanshu Abhishek and Kartik Hosanagar. Keyword generation for search engine advertising using semantic similarity between terms. In *Proceedings of the ninth international conference on Electronic commerce*, pages 89–94. ACM, 2007. 12

[4] Chris Anderson. *The long tail: Why the future of business is selling less of more*. Hachette Books, 2006. 11

[5] Indrajit Bhattacharya and Lise Getoor. Deduplication and group detection using links. In *KDD workshop on link analysis and group detection*, 2004. 13

[6] Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 11–18. ACM, 2004. 13

[7] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3): 297–302, 1945. 13

[8] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American economic review*, 97(1):242–259, 2007. 4

[9] Anindya Ghose and Sha Yang. An empirical analysis of search engine advertising: Sponsored search in electronic markets. *Management science*, 55(10):1605–1622, 2009. 11

[10] Wael H Gomaa and Aly A Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18, 2013. 12

[11] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress, 2010. 12

[12] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018. 27

[13] Ekaterini Ioannou and Minos Garofalakis. Holistic query evaluation over information extraction pipelines. *Proceedings of the VLDB Endowment*, 11(2):217–229, 2017. 19

[14] Ekaterini Ioannou and Slawek Staworko. Management of inconsistencies in data integration. In *Dagstuhl Follow-Ups*, volume 5. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013. 12

[15] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901. 13

[16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016. 18

[17] Brendan Kitts and Benjamin Leblanc. Optimal bidding on keyword auctions. *Electronic markets*, 14(3):186–201, 2004. 11

[18] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997. 13

[19] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966. 12

[20] Kevin Lund. Semantic and associative priming in high-dimensional semantic space. In *Proc. of the 17th Annual conferences of the Cognitive Science Society, 1995*, 1995. 13

[21] Andrew McCallum, Kamal Nigam, and Lyle H Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. Citeseer, 2000. 13

[22] Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780, 2006. 12

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 16

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 16

[25] Claudia Perlich, Brian Dalessandro, Rod Hook, Ori Stitelman, Troy Raeder, and Foster Provost. Bid optimizing and inventory scoring in targeted online advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 804–812. ACM, 2012. 11

[26] Oliver J Rutz and Randolph E Bucklin. A model of individual keyword performance in paid search advertising. *Available at SSRN 1024765*, 2007. 12

[27] Gerard Salton and Michael J McGill. *Introduction to modern information retrieval*. mcgraw-hill, 1983. 13

[28] Shahriar Shariat, Burkay Orten, and Ali Dasdan. Online evaluation of bid prediction models in a large-scale computational advertising platform: decision making and insights. *Knowledge and Information Systems*, 51(1):37–60, 2017. 12

[29] Grigori Sidorov, Alexander Gelbukh, Helena Gómez-Adorno, and David Pinto. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3):491–504, 2014. 18

[30] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961. 4