Intelligent Online Marketing

Predicting Conversion Rate Of New Keywords

Tommy Engström
Abstract

Intelligent Online Marketing

Tommy Engström

This thesis looks at the problem of predicting conversion rate of keywords in Google Adwords where little or no data for the keyword is available. Several methods are investigated and tested on data belonging to three different real world clients. The methods try to predict the conversion rate only given the keyword text. All methods are compared, using two different evaluation methods, with results showing good potential. Finally further improvements are suggested that could have a big impact on the results.
## Contents

1 Introduction ........................................... 4
   1.1 Abbreviations And Definitions ....................... 4
   1.2 Background ........................................... 4
   1.3 Problem Description ................................... 5

2 Methods ................................................ 6
   2.1 Introduction to the used tools ....................... 6
       2.1.1 Python ........................................... 6
       2.1.2 NumPy ........................................... 6
       2.1.3 Pandas .......................................... 6
       2.1.4 Matplotlib ....................................... 7
       2.1.5 Scikit-learn ..................................... 7
       2.1.6 Orange .......................................... 7
   2.2 Preprocessing ........................................ 7
       2.2.1 Data Formats ..................................... 7
   2.3 Inferring CR from other statistics .................. 8
       2.3.1 Motivation ....................................... 8
       2.3.2 Data Exploration ................................ 8
       2.3.3 Algorithm Used .................................. 9
   2.4 Term Model .......................................... 12
       2.4.1 Motivation ....................................... 12
       2.4.2 Preprocessing ................................... 12
       2.4.3 Data Exploration ................................ 13
       2.4.4 Weighted Linear Combination Model ............ 15
       2.4.5 Binary Word Model ................................ 15
       2.4.6 K-Nearest Neighbor ............................... 16
       2.4.7 Distance Weighted K-Nearest Neighbor ........ 18
       2.4.8 Random Forest .................................... 18

3 Results ................................................. 19
   3.1 The Problem In Evaluating Results .................. 19
   3.2 Evaluation Method 1 ................................ 19
       3.2.1 Result Presentation Method ...................... 20
       3.2.2 Mean Prediction Results ......................... 20
Chapter 1

Introduction

1.1 Abbreviations And Definitions

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR</td>
<td>Bounce Rate, $\frac{Visits - NonVisits}{Visits}$</td>
</tr>
<tr>
<td>Broad match</td>
<td>Matching any search query that Google consider similar</td>
</tr>
<tr>
<td>Clicks</td>
<td>The number of times someone clicked on the ad</td>
</tr>
<tr>
<td>Conversions</td>
<td>The number of times the ad led to a purchase or some other desirable action</td>
</tr>
<tr>
<td>CR</td>
<td>Conversions Rate, $\frac{Conversions}{Clicks}$</td>
</tr>
<tr>
<td>CTR</td>
<td>Click Through Rate, $\frac{Clicks}{Impressions}$</td>
</tr>
<tr>
<td>Exact match</td>
<td>Matching only the exact phrase</td>
</tr>
<tr>
<td>Impressions</td>
<td>The number of times an ad have been shown</td>
</tr>
<tr>
<td>Long tail keyword</td>
<td>Keyword with very low traffic</td>
</tr>
<tr>
<td>NonVisit</td>
<td>When a user visits a site but performed no further action</td>
</tr>
<tr>
<td>Match type</td>
<td>How keywords match with a search phrase</td>
</tr>
<tr>
<td>Phrase match</td>
<td>Matching search queries containing the phrase</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Squared Error</td>
</tr>
<tr>
<td>Visit</td>
<td>When a user visits a site</td>
</tr>
<tr>
<td>VPC</td>
<td>Value Per Click</td>
</tr>
</tbody>
</table>

1.2 Background

The work was done at Campanja in their office on Kungsgatan 27, Stockholm. Campanja provides a bidding platform for online advertising that strives to deliver high performance through intelligent real time analysis of data.

One of the most important tasks that Campanja’s software performs is to submit bids in online ad auctions. For search campaigns, search queries are matched to advertisers requests for advertising, in the form of bids for keywords. E.g. a user searches for white widgets (this is called the search query). There
are three advertisers willing to be shown for the word widgets (called a keyword, with an associated match type), and two willing to be shown for white widgets. They will all be ordered by their respective bids and shown on the search page. When a user clicks an ad, the advertiser is charged the price the advertiser with the second highest bid was willing to pay. In reality this process is more complex and also uses quality score, which is how relevant Google think the ad is, and other factors. The true formula is subject to change and not public.

These auctions take place millions of times per day. Auctions are run on a per search basis and affected by bid, relevancy and many more factors. However, from an advertisers point of view, an auction and the subsequent click is just the beginning of a process, that, ideally, leads to the user purchasing something or making another desirable action on the advertisers website. Such a desirable action is usually called a conversion, i.e. the user converts from being just a visitor to becoming a customer. The fraction of users that make a conversion is called the conversion rate, CR. The CR must be big enough for the advertiser to make a profit. It is thus essential to try to figure out the conversion rate in advance, to be able to set a correct bid.

1.3 Problem Description

Ads on Google Adwords are paid for by click, if there is no click there is no cost. The key metric when determining how much one should be willing to pay for an ad is the Value Per Click, VPC, which is defined as $VPC = \text{Conversion Rate} \times \text{Conversion Value}$.

The objective of this thesis is to predict the Conversion Rate, CR, for keywords with little or no data. This is very important since a single customer can have several million active keywords. Many keywords have far to little data to calculate any significant statistics, but combined they make up for a big chunk of the total conversions.
Chapter 2

Methods

All code was written in Python using appropriate libraries. It was a clear choice since it is very suitable for the task and what is currently used for such tasks at Campanja.

2.1 Introduction to the used tools

2.1.1 Python

Python [1] is a dynamically typed, interpreted high level language aiming for high readability and productivity. There are several good machine learning libraries and data analysis available for it.

2.1.2 NumPy

NumPy [2], short for Numerical Python, is a math library for Python. The central part of NumPy is the array for which there are very well optimized C and Fortran functions for linear algebra and manipulation. Many libraries are build upon NumPy.

2.1.3 Pandas

Pandas [3] is a library providing high performance data structures for Python based on NumPy arrays. It provides R-style data frames and very powerful methods for data manipulation. Table 2.1 show a short example of how Pandas is used.
2.1.4 Matplotlib


2.1.5 Scikit-learn

Scikit-learn[5] is one of the most used machine learning libraries for Python.

It contains algorithms for regression, classification, clustering, and dimensionality reduction all working with NumPy arrays.

2.1.6 Orange

Orange[6] is a machine library for Python. Compared with Scikit-learn it provides more high level data structures. It is the main machine learning library used on Campanja as of now.

2.2 Preprocessing

2.2.1 Data Formats

Campanja provided the data in CSV-files. The first data set explored contained 292 columns and $7 \times 10^5$ rows. The data was aggregated by day with the key being Ad group, Campaign, Criterion text. Data exploration showed that the data set was very sparse, a lot of zero values and also missing values. The features that were available, relevant and shared by most clients are described below. These are the only features used in the models. More features were
available but was omitted because of unreliable values, which became obvious after plotting the data.

- Clicks
- Conversions
- Impressions
- Ad group Name
- Campaign Name
- Criterion Text

### 2.3 Inferring CR from other statistics

#### 2.3.1 Motivation

A problem with CR is that a fair amount of data is needed to get a good estimation of the true CR. Conversions are much more scarce than clicks, often more than 50 times so. The initial idea is to predict CR using statistics for which we gather data more rapidly. Two statistics that could be potentially be used is Click Through Rate, CTR, and Bounce Rate, BR.

#### 2.3.2 Data Exploration

To see whether or not a model based on this would be likely to help in predicting CR I started off by examining to see how it’s distributed and to find correlations.

**Conversion Rate Distributions**

Figure 2.1 shows the distribution of CR for different keywords on different clients for keywords with at least 100 clicks. The figures indicate that different clients can have very different properties and could potentially need to be treated differently.

**CR-CTR Correlation**

Figure 2.2 shows scatter plots of CR and CTR and Table 2.2 shows the Pearson Correlation Coefficients, PCC. PCC is the co-variance divided by the product of the standard deviations, defined as $\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$ [7]. It results in a value between $-1$ and $1$ with $0$ representing no correlation and the sign representing the direction of the correlation, e.g a PCC value of 1 would imply that if the CTR were to double so would the CR. PCC assumes both CR and CTR are normally distributed, which is not exactly true but a reasonable approximation.

All keywords with less than 100 clicks were filtered out. The plot for Client 3 looks very suspect. It turned out that the data had been tempered with in order to make up for the problems of having very sparse data.
Table 2.2: Pearson Correlation Coefficients

<table>
<thead>
<tr>
<th>Client</th>
<th>Pearson Correlation Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client 1</td>
<td>0.44</td>
</tr>
<tr>
<td>Client 2</td>
<td>0.29</td>
</tr>
<tr>
<td>Client 3</td>
<td>0.02</td>
</tr>
</tbody>
</table>

**BR-CR Correlation**

Unfortunately it turned out that the BR data had just recently been implemented and a bug had been present early on causing the data to be unreasonable. Due to the small amount of data collected after the fix the BR-CR correlation was omitted. This leaves only CTR to use which unfortunately decreases the potential of the model.

**2.3.3 Algorithm Used**

To predict logCR given logCTR linear regression was used. Linear regression calculates the straight line with the minimum sum of errors. The error function is usually the squared error, $(x_i - y_i)^2$ where $x_i$ is the value predicted and $y_i$ is the observed value. In doing linear regression it is assumed that each point is taken from distributions with equal variance [7], which should be a reasonable approximation once low traffic keywords has been filtered out.
Figure 2.1: CR Distributions

Client 1

Client 2

Client 3

Figure 2.1: CR Distributions
Figure 2.2: CR-CTR Correlation
2.4 Term Model

2.4.1 Motivation

The goal of this model is to be able use the information about words present in a keyword in a more elaborate way. The idea is to construct a model that exploits the information of what particular words are present and uses it to predict the conversion rate. Another good thing about this is that the statistics for words reach significant levels quicker than statistics for keywords. This allows for a way to estimate the value of a keyword when we have statistics for the words, or at least some of them, but none at all for the keyword.

2.4.2 Preprocessing

For a model to be able to make use of this information the information has to be restructured in an appropriate format. Table 2.3 show the preprocessing algorithm. Since the algorithm return a Data frame and also counts the number of occurrences for each word it provides a very good way of filtering the data before using it in a predictive model.

Table 2.3: Word Model Preprocessing

1. Let $K = K_1...K_n$ represent the set of keywords, $W_i$ the set or words in $K_i$ and $T$ the set of all words, $T = W_1 \cup W_2...W_n$.

2. For each word, $T_i$, sum the Impressions, Clicks and Conversions of all keywords it is part of $S_i = \{i\mid W_i \in K_i\}$,

   $T_i^{Impressions} = \sum_{j \in S_i} K_j^{Impressions}$

   $T_i^{Clicks} = \sum_{j \in S_i} K_j^{Clicks}$

   $T_i^{Conversions} = \sum_{j \in S_i} K_j^{Conversions}$

3. Calculate

   $T_i^{CR} = \frac{T_i^{Conversions}}{T_i^{Clicks}}$

   $T_i^{CTR} = \frac{T_i^{Clicks}}{T_i^{Impressions}}$
Table 2.4: Keyword data to word data

<table>
<thead>
<tr>
<th>Keyword Text</th>
<th>Impressions</th>
<th>Clicks</th>
<th>Conversions</th>
<th>CR</th>
<th>CTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>movies online</td>
<td>1200</td>
<td>225</td>
<td>20</td>
<td>0.09</td>
<td>0.19</td>
</tr>
<tr>
<td>free movies</td>
<td>1500</td>
<td>154</td>
<td>3</td>
<td>0.02</td>
<td>0.10</td>
</tr>
<tr>
<td>action movies</td>
<td>210</td>
<td>107</td>
<td>17</td>
<td>0.16</td>
<td>0.51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word</th>
<th>Impressions</th>
<th>Clicks</th>
<th>Conversions</th>
<th>CR</th>
<th>CTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>210</td>
<td>107</td>
<td>17</td>
<td>0.16</td>
<td>0.51</td>
</tr>
<tr>
<td>free</td>
<td>1500</td>
<td>154</td>
<td>3</td>
<td>0.02</td>
<td>0.10</td>
</tr>
<tr>
<td>movies</td>
<td>2910</td>
<td>486</td>
<td>40</td>
<td>0.08</td>
<td>0.17</td>
</tr>
<tr>
<td>online</td>
<td>1200</td>
<td>225</td>
<td>20</td>
<td>0.09</td>
<td>0.19</td>
</tr>
</tbody>
</table>

2.4.3 Data Exploration

Figure 2.3 show the distribution of CR on word level for two customers the red line marks the average CR. The figures show that the CR per word is about as spread as the CR per keyword, making it seem very plausible that there is information to be extracted this representation.
Figure 2.3: Distributions of word CR
2.4.4 Weighted Linear Combination Model

Motivation

The idea behind this method is to use the CR of the individual words of the keyword to predict the CR of the keyword. By calculating statistics for individual words we get statistics with high confidence sooner which allows the model to take advantage of data for the client even when there is no data available for the given keyword. Although it seems very likely this information could be used to make an estimate of the CR it is not obvious how to combine the word estimates in a good way. The following sections describes the approaches tested.

Algorithms

To find a good way to combine the statistics of individual words into a prediction of the CR for a keyword several different formulas were tried. Table 2.5 shows the formulas initially tested.

The formulas in Table 2.5 represent different ways to combine the words yet they fail to take into account the importance of individual words. With that in mind a weighted average formula as shown in table 2.6 was proposed. It’s built upon the idea that words with CR that deviate more from the average CR will be more important. In the case where statistic was not available for a word the average CR was used, giving that word zero weight.

Table 2.5: CR Combining Formulas

<table>
<thead>
<tr>
<th>CR Type</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average CR</td>
<td>( K = \frac{1}{n} \sum_{i=1}^{n} W_i )</td>
</tr>
<tr>
<td>Max CR</td>
<td>( K = \max(W_i) )</td>
</tr>
<tr>
<td>Min CR</td>
<td>( K = \min(W_i) )</td>
</tr>
</tbody>
</table>

\( K = \)Keyword CR, \( W = \)Word CR, \( A = \)Average CR, \( n = \)Number of words in Keyword

Table 2.6: Weighted Average Formula

1. Calculate the weight constants, \( C_{W_i}' = W_i - K_{\text{mean}}, i = 0...n \)
2. Normalize, \( C_{W_i} = \frac{C_{W_i}'}{\sum C_{W_i}' } \)
3. Calculate the new CR as, \( K = \sum_{i=0}^{n} C_{W_i}W_i \)

2.4.5 Binary Word Model

To be able to use common machine learning algorithms predicting the CR the data had to be restructured in a suitable format. The format proposed is based on binary features representing the occurrence of a word along with natural
number for features representing the number of words and the number of unknown words. Unknown words show up when words are filtered out or when completely new words show up in the test set. Table 2.7 shows the algorithm and 2.8 shows the format of the data for a made-up data set.

Table 2.7: Binary word model creation algorithm

1. Calculate word statistics as described in 2.3.
2. \( M_{ij} = \begin{cases} 1, & W_i \in K_i \\ 0, & W_i \notin K_i \end{cases} \), where \( K \) and \( W \) are the sets of keywords and words.
3. Save \( M_{ij} \) together with \( K_i \) and \( CR_i \)

Table 2.8: Example of training data

<table>
<thead>
<tr>
<th>( W_{count} )</th>
<th>( U_{count} )</th>
<th>buy</th>
<th>food</th>
<th>pizza</th>
<th>sushi</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.04</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.02</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.09</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.11</td>
</tr>
</tbody>
</table>

\( W_{count} \) represent word count and \( U_{count} \) represent the count of unknown words.

2.4.6 K-Nearest Neighbor

KNN is very suitable for this data format. The data is structured in such a way that keywords that share words and length will be considered similar. It is also a good base approximation to compare other methods to [8]. The algorithm works as described in Table 2.9.

Table 2.9: KNN Regression Algorithm

1. Define the distance as between a keyword, \( W \), and the target, \( T \), as \( \sum_{i=1}^{N} |W_i - T_i| \), where \( i \) is the column as shown in table 2.8 and \( N \) is the number of word.
2. Let \( N \) be the set of CRs for the \( K \) keywords closest by distance to the target keyword.
3. Calculate estimated target CR as \( \frac{1}{K} \sum_{i=1}^{K} N_i \)
Table 2.10: KNN Example

The CR prediction of “buy pizza or kebab” given the data in Table 2.9 and $K = 2$.

1. Convert to the format of Table 2.9 and call it $P$:

<table>
<thead>
<tr>
<th>$W_{count}$</th>
<th>$U_{count}$</th>
<th>buy</th>
<th>food</th>
<th>pizza</th>
<th>sushi</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The word “or” was not part of the set of words and are thus counted as an unknown word.

2. Calculate the distance to each sample in Table 2.9 as $Distance_i = \sum_{j \in \text{features}} (S_{ij} - P_j)^2$, where $S$ are the samples from the training data.

<table>
<thead>
<tr>
<th>$W_{count}$</th>
<th>$U_{count}$</th>
<th>buy</th>
<th>food</th>
<th>pizza</th>
<th>sushi</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2 – 3)$^2$</td>
<td>(0 – 1)$^2$</td>
<td>(1 – 1)$^2$</td>
<td>(1 – 0)$^2$</td>
<td>(0 – 1)$^2$</td>
<td>(0 – 0)$^2$</td>
<td>4</td>
</tr>
<tr>
<td>(3 – 3)$^2$</td>
<td>(1 – 1)$^2$</td>
<td>(1 – 0)$^2$</td>
<td>(1 – 0)$^2$</td>
<td>(0 – 1)$^2$</td>
<td>(0 – 0)$^2$</td>
<td>2</td>
</tr>
<tr>
<td>(2 – 3)$^2$</td>
<td>(0 – 1)$^2$</td>
<td>(1 – 1)$^2$</td>
<td>(0 – 0)$^2$</td>
<td>(0 – 1)$^2$</td>
<td>(1 – 0)$^2$</td>
<td>4</td>
</tr>
<tr>
<td>(3 – 3)$^2$</td>
<td>(0 – 1)$^2$</td>
<td>(0 – 1)$^2$</td>
<td>(0 – 0)$^2$</td>
<td>(1 – 1)$^2$</td>
<td>(1 – 0)$^2$</td>
<td>3</td>
</tr>
<tr>
<td>(4 – 3)$^2$</td>
<td>(2 – 1)$^2$</td>
<td>(1 – 1)$^2$</td>
<td>(0 – 0)$^2$</td>
<td>(1 – 1)$^2$</td>
<td>(0 – 0)$^2$</td>
<td>2</td>
</tr>
</tbody>
</table>

3. Average the value of the 2 samples with the shortest distance to the target, $\frac{(-2.56 + (-4.12))}{2} = -3.34$
2.4.7 Distance Weighted K-Nearest Neighbor

This approach comes from the intuition with a CR that diverges a lot from the average will likely influence the keyword’s CR more. Research have shown that using distance weighting can allow the algorithm to use a higher K, which makes it more robust [9]. The algorithm used is the same but instead of using binary features for each word they are represented with a scalar, $d_i = |W_i - A|$, that represent the words distances from the average CR.

Given that the average CR is 6% and the word “pizza” has a CR of 14% a the weight of the “pizza” feature would be $|0.14 - 0.06| = 0.08$. If the keyword also contains the word “salad” with a CR of 7% that word will influence the distance to other keywords less.

2.4.8 Random Forest

Random forest is an ensemble method consisting of a number of decision trees. A regression decision tree is a binary tree with each node containing a splitting criterion and each leaf containing a scalar. Figure 2.4 show a minimal random forest with two trees. If used to calculate the CR of “free action movie” it would predict $\frac{0.01 + 0.02}{2} = 0.015$

The downside with decision trees is that the tend to over fit the data unless there ability to split is restricted. In a random forest the high variance problem is countered by using an ensemble of decision trees trained on different subsets of the data. Since each tree is trained on a different sample they will all have different sample errors. It can be shown that given a sufficient number of trees a random forest will always converge [10].

Random forests are also robust, the algorithm will never predict a value outside the range of values seen in the training data which make it less sensitive to outliers is the test data. Another benefit is that very little tweaking of parameters is needed since the over fitting problem is handled by the ensemble given that enough trees are used.

Figure 2.4: Random forest example
Chapter 3

Results

3.1 The Problem In Evaluating Results

3.2 Evaluation Method 1

When it comes to evaluating the results it is a challenge in itself. The goal is to predict the CR of keywords for which we do not yet have enough data, but for such keywords we have no way of knowing the true CR. All methods were tested on keywords for which we have enough data but it should be noted that these keywords can have different characteristics from keywords with little data. For example, high traffic keywords usually consist of fewer words as shown in Figure 3.14.

The algorithms were trained and tested on the keywords with at least 100 clicks. The word statistics were calculated on all the data in the training set, after that the word data where filtered in the same manner as the keyword data.

The evaluation of the results is done using 10-fold cross validation. Cross validation have two benefits. First, it minimizes the risk of selection bias when choosing training and test sets, second, it allows the algorithms to train on more data without risking the test set to be unrepresentative. K-Fold Cross Validation works as shown in table 3.1. The folds are selected at random.

To compare results the root mean squared error, RMSE, was used as prediction accuracy metric.
Table 3.1: K-Fold Cross Validation

1. Split the data into K sets
2. Take away 1 set for testing
3. Train an algorithm on $K - 1$ of the sets
4. Test the algorithm on the test set
5. Iterate using every set as test set once
6. Use all test results as total result

3.2.1 Result Presentation Method
The result for each model is visualized using a scatter plot showing the prediction compared to the observed value. The plot also contain a line diagonal representing perfect correlation. The RMSE is calculated for each plot and presented in the title and a summary of all RMSE improvements versus mean is presented in a table.

3.2.2 Mean Prediction Results
The plots in figure 3.1 show the prediction performance when assigning the average CR to every keyword. This is used as a benchmark for all other models and should also be a good estimate of what the current system would predict today.

Each plot contains 10 individual lines, one for each fold. For Client 2 and 3 this is barely visible using this scale but for Client 1 there is a big gap. It turns out that Client 1 has over 40% of its total ad-clicks coming from a single keyword. The keyword has a CR of 9.1% instead of the average 7.1%, which actually brings the average CR of all other keywords to 5.6%.

3.2.3 CTR Regression Results
Figure 3.2 show the results of the CTR Regression model. The model significantly improves the predictions for client 1, which is to be expected since Client 1 had the strongest correlation between CTR and CR.

The result for Client 3 is a great example of when linear regression goes wrong. The method has no good way of handling outliers.
3.2.4 Term Model Results

Weighted Average Formula Results
The weighted average formula turned out to be superior over other formulas proposed in 2.4.4. Figure 3.3 shows the results for the weighted average formula.

K-Nearest Neighbor
For all these plots $K = 3$ was used. This turned out to be a good fit for all data sets but it is dependent on the number of instances in the data set. The results are presented in figure 3.4.

Distance Weighted K-Nearest Neighbor
In the distance weighted KNN version a $K = 5$ was used. The distance weighing made it so that the $K$-value could be increased without dropping performance. Figure 3.5 shows the results of the predictions.

Random Forest
The random forest uses 20 decision trees, increasing the number of trees had little influence on the prediction results. The results are shown in Figure 3.6.

Model Combinations
In addition to the models described in the previous chapter a number of combinations of models were tested. The best combination was using weighted average + random forest. The model predicts the CR to be the unweighted average of the two models.

The result for the combination of models are shown in Figure 3.13.

Summary
Table 3.2 shows a shows the RMSE of every model divided by the RMSE of the mean-based model. This method was chosen to make it easy to compare performance of the algorithms for different clients, since the RMSE varies a lot between them.
<table>
<thead>
<tr>
<th>Model</th>
<th>Client 1</th>
<th>Client 2</th>
<th>Client 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean CR</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>CTR Regression Model</td>
<td>0.6599</td>
<td>0.9538</td>
<td>2.4547</td>
</tr>
<tr>
<td>Term Model: Random Forest</td>
<td>0.6301</td>
<td>0.4794</td>
<td>0.8359</td>
</tr>
<tr>
<td>Term Model: KNN</td>
<td>0.7029</td>
<td>0.5339</td>
<td>0.9188</td>
</tr>
<tr>
<td>Term Model: Distance KNN</td>
<td>0.6505</td>
<td>0.5546</td>
<td>0.9004</td>
</tr>
<tr>
<td>Term Model: Weighted Average</td>
<td>0.7673</td>
<td>0.6352</td>
<td>1.0</td>
</tr>
<tr>
<td>Combination</td>
<td>0.6524</td>
<td>0.4956</td>
<td>0.8492</td>
</tr>
</tbody>
</table>

Table 3.2: Algorithm RMSE divided by Mean RMSE
Figure 3.1: Mean Prediction Results
Figure 3.2: CTR Regression Results

**Client 1 CTR Regression Model**
RMSE = 0.024

**Client 2 CTR Regression Model**
RMSE = 0.030

**Client 3 CTR Regression Model**
RMSE = 0.021
Figure 3.3: Formula Approach
Figure 3.4: KNN

Client 1 Term Model: KNN
RMSE=0.026

Client 2 Term Model: KNN
RMSE=0.017

Client 3 Term Model: KNN
RMSE=0.008

Figure 3.4: KNN
Figure 3.5: Distance Weighted KNN
Figure 3.6: Random Forest

Client 1 Term Model: Random Forest
RMSE=0.023

Client 2 Term Model: Random Forest
RMSE=0.015

Client 3 Term Model: Random Forest
RMSE=0.007
Figure 3.7: Combination: Weighted Average + Random Forest
3.3 Evaluation Method 2

To test the models on data from actual long tail keywords an aggregation method was constructed. The idea is to combine several keywords into one and use the predictions for each individual keyword weighed together as the group result. The algorithm works as described below.

1. Predict the number of conversions for each keyword.
2. Group keywords together so that each group has at least 500 clicks combined.
3. Divide the combined number of conversions with the combined number of clicks.

The algorithms were trained on all the keywords with at least 100 clicks. The word data was calculated from all keywords with 50 clicks or more and the testing were done using keywords with less than 50 clicks.

3.3.1 Mean Prediction Results

The plots in figure 3.8 show the prediction performance when assigning the average CR to every keyword. This is used as the benchmark for the other models.

Weighted Average Results

Figure 3.9 show the result for the weighted average formula.

K-Nearest Neighbor

Figure 3.10 show the result of the KNN model on the long tail keywords. Just as when testing on keywords with more data $K = 3$ was used.

Distance Weighted K-Nearest Neighbor

Figure 3.11 show the results of the distance weighted KNN model using $K = 5$.

Random Forest

Figure 3.12 show the performance of the Random Forest predictor using 20 trees.

Model Combinations

In addition to the models described in the previous chapter a number of combinations of models were tested. The best combination was using weighted average + random forest. The model predicts the CR to be the unweighted average of the two models.

The result for the combination of models are shown in Figure 3.13.
Summary

Table 3.3 shows the RMSE of every model divided by the RMSE of the mean-based model.

Table 3.3: Algorithm RMSE divided by Mean RMSE

<table>
<thead>
<tr>
<th>Model</th>
<th>Client 1</th>
<th>Client 2</th>
<th>Client 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean CR</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Term Model: Random Forest</td>
<td>0.312</td>
<td>0.664</td>
<td>1.013</td>
</tr>
<tr>
<td>Term Model: KNN</td>
<td>0.456</td>
<td>0.757</td>
<td>1.024</td>
</tr>
<tr>
<td>Term Model: Distance KNN</td>
<td>0.344</td>
<td>0.884</td>
<td>1.011</td>
</tr>
<tr>
<td>Term Model: Weighted Average</td>
<td>0.407</td>
<td>0.721</td>
<td>1.0</td>
</tr>
<tr>
<td>Combination</td>
<td>0.336</td>
<td>0.629</td>
<td>0.956</td>
</tr>
</tbody>
</table>
Figure 3.8: Mean Prediction Results
Figure 3.9: Weighted Average
Figure 3.10: KNN

Client 1 Term Model: KNN
RMSE=0.019

Client 2 Term Model: KNN
RMSE=0.015

Client 3 Term Model: KNN
RMSE=0.006

Figure 3.10: KNN
Figure 3.11: Distance Weighted KNN

Client 1 Term Model: Distance KNN
RMSE=0.014

Client 2 Term Model: Distance KNN
RMSE=0.017

Client 3 Term Model: Distance KNN
RMSE=0.006

Figure 3.11: Distance Weighted KNN
Figure 3.12: Random Forest

Client 1 Term Model: Random Forest
RMSE=0.013

Client 2 Term Model: Random Forest
RMSE=0.013

Client 3 Term Model: Random Forest
RMSE=0.006

Figure 3.12: Random Forest
Figure 3.13: Combination: Weighted Average + Random Forest
3.4 Result Analysis

The results look very promising for Client 1 and 2, making very significant improvements of the RMSE in both evaluation methods. The worse results of Client 3 is likely related to the smaller size of the data set and the data injections. Client 3 also has a much more concentrated distribution of CRs, making the mean predictor perform much better than for other clients.

Unfortunately the term model suggested in this thesis relies on keywords with more confident statistics to calculate the CR of keywords with no statistics. With no high traffic keywords in the data set it cannot perform well.

Considering how different the data looks for different clients it is not that surprising that certain models do better on some clients than others. The bad long tail results for Client 3 could very likely be due to that data set being significantly smaller than other data sets.

Regarding the CTR model it should be noted that the test set contained only keywords for which it is usable, but the keywords that are the primary target will not have as reliable statistics. For the term model however there is reason to believe it will add value to long tail keywords since it does not rely on anything other than the text in the keyword to make it’s predictions.

To give an idea of how the filtering skews the distribution of keywords Figure 3.14 shows the number of keywords of different length before and after filtering. As shown in the figures the keywords that have enough conversions to be used to test the model are not completely representative of the whole data set and since they have statistics they would not be predicted using the models in this paper. That being said there is still a fair amount of spread in the length of the keywords.
Figure 3.14: Keyword length distributions
Chapter 4

Future Improvements

There is quite a lot of room for improvements of the results. The results in this paper were calculated on criterion text, which is the text in the keyword. The keyword also has a match type which was not part of the data sets used here. There are three types of match types: Exact, Phrase and Broad. Exact match only matches searches equal to the criterion text, phrase match matches searches containing all the words and broad match searches that Google consider to be similar. The exact search query can be attained and used to calculate the word statistics, in which case the model would make use of the extra information gained by using broad match keywords. Calculating the statistics using search queries and including match type in the model would most likely improve the result.

As of now the term models work only with words but it would require little work to switch to a true term model, where combinations of words that appear together often will be treated as a one term. This would provide the algorithms with a way to distinguish terms such as “free trial” from other uses of the word “free”, which could prove valuable.

Another thing that should be investigated is if the performance can be improved by treating branding keywords differently. The easiest way to accomplish this would be not to include them when calculating the statistics for the term model, since branding traffic usually contains few long tail keywords anyway.

Also, if used in production this model should not be used on it’s own except when setting the first bid since it does not take the statistics for the keyword into account.
Chapter 5

Critical Analysis

During the development of this prototype I made a few time consuming mistakes. The first being a bad choice of method to import the data, leaving me with a slow implementation which I ended up rewriting using the excellent Pandas library.

The second implementation mistake was using $CR' \approx \frac{1 + \text{Conversions}}{1 + \text{Clicks}}$ as an approximation of CR to avoid division by zero. I did this before I came up with the term model and stuck with it, only to realize later on that I didn’t need it anymore and the only effect it had was making the data less correct.

I also had no good benchmark on the performance. I would have liked to compare it to Campanja’s current system but that could not be done. I ended up using the mean CR for each client as a benchmark and learned later that this is not a fair comparison of the current system’s performance.

The injected data into the data sets adds uncertainty to the results as well.

I also made the mistake of using an approximation of CR, $CR' \approx \frac{1 + \text{Conversions}}{1 + \text{Clicks}}$, to avoid division by zero. The problem with this is that it skews the values where Conversions $\approx 0$ making them useless anyway.
Bibliography

[1] python.org
[2] numpy.org
[3] pandas.pydata.org
[4] matplotlib.org
[5] scikit-learn.org
[6] orange.biolab.si