Linköping University | Department of Computer and Information Science
Master thesis, 30 ECTS | Datateknik
202019 | LIU-IDA/LITH-EX-A--2019/003--SE

Weight Estimation and Evaluation of User Suggestions in Mobile Browsing

Estimering och utvärdering av vikter i en mobilwebbläsares användarförslagssystem

Oscar Johansson

Supervisor : Jody Foo
Examiner : Marco Kuhlmann

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

This study investigates the suggestion system of a mobile browser. The goal of a suggestion system is to assist the user by presenting relevant suggestions in an ordered list. By weighting the different types of suggestions presented to the user, such as history, bookmarks etc., it is investigated how this affects the performance of the suggestion system. The performance is measured using the position, error and Mean Reciprocal Rank of the chosen suggestion as well as the number of written characters. It is also measured if the user chose to not use the suggestion system, by searching or entering the entire URL. The weights were estimated using a Genetic Algorithm. The evaluation was done by performing an A/B test, were the control group used an unweighted system and the test group used the weights estimated by the genetic algorithm. The results from the A/B test were statistically analyzed using BEST and Bootstrap. The results showed an improvement of position, number of written characters, MMR and the error. There was no change in how much the user used the suggestion system. The thesis concluded that there is a correlation between the position of the desired suggestion and when the user stops typing, and that weighting types is a way to improve said position. The thesis also concludes that there is a need for future work in regards to evaluation of the optimization algorithm and error measurement.

Acknowledgments

I want to give a special thanks to my supervisor Jody, examiner Marco and external supervisor Stefan for you help, support and feedback on this thesis. Thank you Rasmus for the help regarding optimization algorithms. Thank you Tomas for your overall input and help. Thank you Joel for the help and reviewing of the implementations. Thank you Niklas for the help with A/B testing. Thank you Magnus for the help and feedback regarding statistical evaluation.

Contents

List of Figures

List of Tables

1 Introduction

As of 2017, more than half of the world's population had access to the internet[1]. Around half of all internet traffic is mobile data traffic[2], a traffic type that has been growing tremendously in the last decade. Typing on a mobile device is considered inconvenient by many users. In a study from 2012, 243 people were monitored while reading and typing on a smartphone and a desktop computer [3]. The average typing speed on a smartphone was 21.79 words per minute, compared to 59.27 words per minute on a desktop computer. The test subjects expressed that typing on a smartphone is a problem. Another study showed similar results, where the average typing speed on a virtual QWERTY-keyboard was 17.23 words per minute [46].

When navigating the web using a mobile browser, a user typically starts typing in an address bar. As the user types, the browser suggests sites that are relevant for the user's input. The suggestions are represented as an ordered list, see Figure 1.1, where a higher order corresponds to a higher position. Several studies have shown that a user will value suggestions with a higher position more than those with a lower position [35, 32]. A study using eye-tracking showed that, when presented with a list of suggestions, the user only looks at the top of the list [9]. Therefore, it is important for the browser that the correct or desired suggestion is placed high in the list.

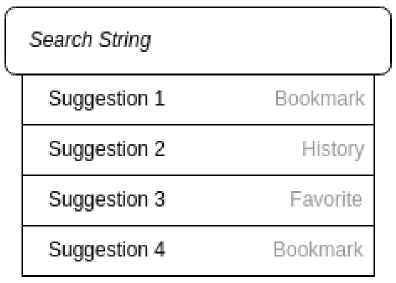


Figure 1.1: Visualization of how the suggestions are presented to the user

[1]https://en.wikipedia.org/wiki/Global_Internet_usage, accessed: 24-09-2018
[2]https://www.statista.com/topics/779/mobile-internet/, accessed: 24-09-2018

To be able to suggest sites, the browser needs to store and utilize data about the user. It can be data that the user actively saved, such as favorites and bookmarks. It can also be data that the browser saved based on the user's behavior, such as browser history. When suggesting sites, the browser needs to combine these different types of sites.

To achieve the goal of providing relevant suggestions high in a suggestion list, thus decreasing the typing required, there are several things the browser can utilize. As in all systems, it is of interest to evaluate how well the system performs, and if other approaches could perform better. One of most the popular methods for comparing two alternatives is A/B testing. The testing method is widely used by large companies, e.g. Facebook [2] and Netflix [25] use the method and both companies claim that A/B testing is the foundation for experimentation and testing of new features or improvements of existing features. When performing an A/B test, what to test and how to evaluate the results are two key aspects.

Instead of using the suggestions presented, a user has the option to either search using the input string or enter a complete URL. It is not always a user's intention to use the suggestions, examples could be when a URL is not present in the user's previous data or when the intention is to search. However, an increase or a decrease in the usage of the suggestions could measure the suggestions usefulness to the user.

1.1 Research Problem

Since typing on a mobile phone can be seen as inconvenient, reducing the required typing could lead to better user experience. The suggestion system aims to minimize the required typing, by collecting suggestions of different types. By weighting the suggestion based on type, the position of the correct or chosen suggestion can be adjusted. The aim is to place the correct suggestion as high in the list as possible. Changing input to minimize or maximize an output is an optimization problem. By using logged data from the system, new weights could be estimated to optimize the position.

To investigate how the weighting of types affect both position and typing, a test has to be performed on real users.

1.1.1 Research Questions

This thesis aims to investigate if different types of suggestions are equally important for a user. This will be done by answering the following questions:

1. How can the suggestion error rates be decreased by the introduction of weights in the system?
2. How will users' typing be affected by improving the position of the correct suggestion?
3. How can the position improvement of the correct suggestion increase the usage of suggestions, as measured by the percentage of requests being an entered address or a search?

1.2 Delimitations

This thesis will investigate how different suggestion types (such as bookmarks, history, etc.) are weighted against each other. It will not evaluate the string-matching that occurs before weighting.

The weighting will be the same for all users, meaning that the values of the different weights can be seen as constants that get sent to all the users. There might exist approaches where a weight model is trained on the device, optimizing for the specific user, but due to performance and privacy limitations, no individual weighting for the users will be considered.

2 Background

2.1 Browser Suggestions

The conducted study of this thesis will be performed on a mobile browser. To understand the rest of the thesis, it is of importance to understand how the suggestion system works. Therefore, this section aims to provide a full overview of the suggestion system of the browser.

As the user starts to type in the address bar, the input string is sent to three different suggestion providers. The different suggestion providers are:

- History suggestion provider that will provide suggestions from the user's history.
- Bookmark suggestion provider that will provide suggestions from the user's bookmarks.
- Favorite suggestion provider that will provide suggestions from the user's favorites.

Every website in the suggestion providers are represented as a suggestion item. A suggestion item consists of a title and a URL (e.g. a suggestion item corresponding to Facebook would have a title *Facebook* and URL *http://facebook.com*). Each suggestion provider will perform a string-matching on the input string against each possible suggestion item. If a match occurs a string-matching score will be calculated, the string-matching algorithm calculates the score based on how many characters that match and at what position the matching occurs. The string-matching score will be used to calculate the suggestion score, the suggestion score is calculated using:

$$suggestionScore_{type} = weight_{type} \cdot stringMatchingScore + base_{type}$$

As can be seen in the formula above each type of suggestion has its own *weight* and *base*. A suggestion type corresponds to where string-matching occurred (e.g. a match in the title of a history suggestion would be of type *history_title*). The different types can be seen in Table 2.1. $weight_{type}$ is the weight, for a certain type, that controls how much the string-matching score contributes to the suggestion score. $base_{type}$ is the base weight that controls how much the type contributes to the matching score overall.

Once all providers have returned a list of suggestions, they are sorted by their suggestion score, with a higher score being placed higher in the suggestion list. The suggestions are graphically presented to the user, see Figure 1.1.

There are several ways the user may proceed from the address bar and the suggestion system. The user can select a suggestion, search on the input string or enter an address and navigate to it. When the user has performed one of the actions, a request is sent to the internet, see Figure 2.1. Some of the request data is also sent to the browser's server to be logged, the following data gets logged:

- Title of the correct suggestion
- URL of the correct suggestion

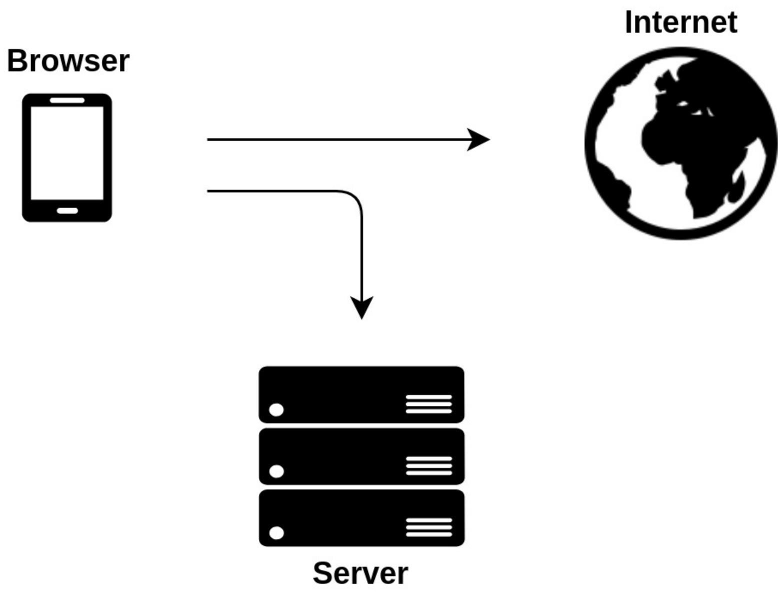


Figure 2.1: Visualization of how a request is sent from the browser

Table 2.1: The different types used when calculating the suggestion score

Name	Description
BOOKMARK_TITLE_BASE	Base for a title match in a bookmark item
BOOKMARK_TITLE_WEIGHT	Weight for a title match in a bookmark item
BOOKMARK_URL_BASE	Base for a URL match in a bookmark item
BOOKMARK_URL_WEIGHT	Weight for a URL match in a bookmark item
FAVORITE_TITLE_BASE	Base for a title match in a favorite item
FAVORITE_TITLE_WEIGHT	Weight for a title match in a favorite item
FAVORITE_URL_BASE	Base for a URL match in a favorite item
FAVORITE_URL_WEIGHT	Weight for a URL match in favorite item
HISTORY_TITLE_BASE	Base for a title match in a history item
HISTORY_TITLE_WEIGHT	Weight for a title match in a history item
HISTORY_URL_BASE	Base for a URL match in a history item
HISTORY_URL_WEIGHT	Weight for a URL match in a history item
HISTORY_TYPED_TITLE_BASE	Base for a title match in a history item that the user has typed
HISTORY_TYPED_TITLE_WEIGHT	Weight for a title match in a history item that the user has typed
HISTORY_TYPED_URL_BASE	Base for a URL match in a history item that the user has typed
HISTORY_TYPED_URL_WEIGHT	Weight for a URL match in a history item that the user has typed

The different types of values used to calculate suggestions (shown in Table 2.1) are stored on the server. The server sends the values to the mobile, where it is used to weight different suggestions types against each other. The browser supports A/B testing on the values, by dividing the users into groups and sending different values of the types to each group. There are several variables to adjust when dividing users into group.

3 Theory

This chapter will present the theory required to understand the thesis and will follow the same order as Chapter 4 and 5. It will start off by presenting different approaches to measure suggestion performance. It will then introduce weight optimization, followed by the theoretical foundation for the genetic algorithm. To evaluate an A/B test, several approaches exists. This chapter will present the frequentist, the Bayesian and the bootstrap approach. Lastly, work related to weighting and evaluation of different types will be presented.

3.1 Measuring Suggestion Performance

This section will present two different performance metrics: Mean Average Precision and Mean Reciprocal Rank.

3.1.1 Mean Average Precision

Mean Average Precision (MAP) is a metric to evaluate an ordered list of items, where several items could be relevant. A higher order corresponds to a higher position in the list, the highest position is 1. MAP is defined as:

$$MAP(L) = \frac{1}{|L|} \sum_{i=1}^{|L|} AveragePrecision(L_i)$$

Where L is a set of ordered lists, L_i is the i^{th} ordered list containing items and $AveragePrecision(L_i)$ calculates the average position of the relevant items in L_i. [36]

3.1.2 Mean Reciprocal Rank

Mean Reciprocal Rank (MRR) is metric to evaluate a ordered list of items. MRR only considers the highest positioned relevant item. The highest position on the list corresponds to a position of 1. It is a equivalent to MAP when there only exist one relevant item. A change from position 1 to 2 (0.5) is much larger than a change from position 9 to 10 (0.011). [13]

MRR is defined as:

$$MRR(L) = \frac{1}{|L|} \sum_{i=1}^{|L|} \frac{1}{position_i}$$

Where L is an ordered list of items and $position_i$ is the first relevant item.

3.2 Weight Optimization

Optimization is the task of minimizing or maximizing a function, by changing the inputs to the function. In the suggestion system such a function measures the performance of the suggestions. The output of the function varies depending on what weights the suggestion system receives. The weight can therefore be considered to be input to the performance function. There exist many algorithms to solve an optimization problem. This section will present the theory to the *Genetic Algorithm*.

3.2.1 Genetic Algorithm

A genetic algorithm is a subclass of evolutionary algorithms. It is a search-based metaheuristic used for solving optimization problems. A genetic algorithm has a set of steps to be performed. The idea behind the genetic algorithm is loosely based on natural selection, from which most of the terminology is fetched. The main objective is to have a population of candidates that, through evolution, improves each generation. The performance of a solution candidate, called *chromosome*, is measure by a *fitness score*. [37]

A flowchart displaying the required steps in a genetic algorithm can be seen in Figure 3.1. This section will describe each step in the flowchart.

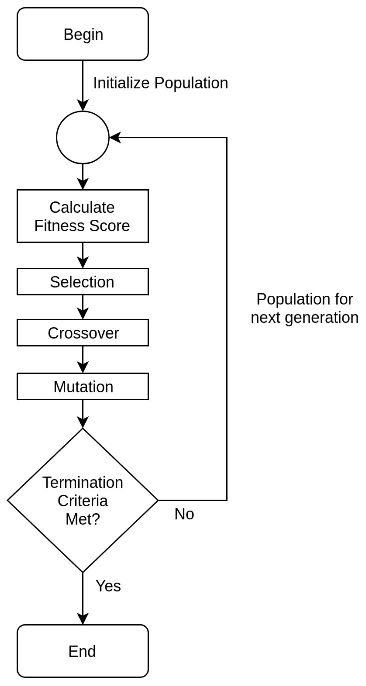


Figure 3.1: Flowchart showing the procedure of a genetic algorithm

Terminology

The terminology is, as stated before, borrowed from natural selection and biology. A candidate for a solution, for example a possible set of values/combinations for the given problem, is called a *chromosome*. If the given problem would be to estimate five numerical values for maximizing a function, a chromosome would be a set of five numerical values. A chromosome consists of *genes*, in the given example a gene would be one of the five numerical values. All chromosomes together are called *population*. For a graphical representation of genes, chromosomes and a population see Figure 3.2. All chromosomes in the population are evaluated using a *fitness score*, a score measuring the performance of a chromosome. A number of chromosomes from a population are selected, using a *selection method*, to breed the next generation. The procedure to generate new chromosomes from the selected chromosome is called *crossover*. To avoid local optimas, a randomness is introduced, called *mutation*. This procedure is repeated generation after generation until a *termination criterion* is met. The chromosome with the highest fitness score from the last generation will be used as the solution to the problem.

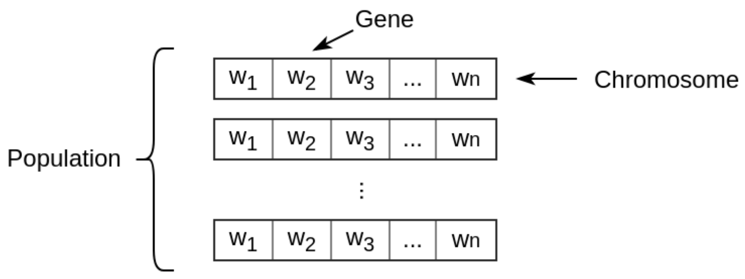


Figure 3.2: Figure showing how the weights could be represented in the genetic algorithm

Representation

A chromosome is often represented as an array of genes. How the genes are represented is problem and implementation specific. The chromosome could be represented as an array of bits, permutations, real numbers or problem specific objects. [37]

Fitness Score

The *fitness score* or the *fitness function*, is the performance measurement of the algorithm. Depending on the problem, the aim is to either maximize or minimize the score. The fitness function is the only part of the algorithm where it can receive directions on how to optimize the problem. [37]

Selection

The selection of which chromosomes that breeds the next generation can be done in a number of ways. This section will present roulette wheel selection, stochastic universal sampling and tournament selection.

The *roulette wheel selection* is a selection method that uses the following formula to calculate the probability that a chromosome is chosen for breeding [37]:

$$P(x_i) = \frac{f(x_i)}{\sum_{j=0}^{n} f(x_j)}$$

Where x_i is a chromosome, $f(x)$ returns the fitness score of a chromosome and n is the number of chromosomes in the population.

A chromosome is then chosen using the probability given from the formula above. This is repeated until the wanted number of chromosomes are picked. The name originates in the method being represented as a spinning roulette wheel, where a higher probability corresponds to a larger area on the spinning wheel. [24]

Stochastic Universal Sampling (SUS) is another selection method, similar to roulette wheel selection. Following the roulette wheel analogy, where a better fitness corresponds to a larger area on the wheel, SUS can be seen as a spinning roulette wheel with several pointers that are equally spaced, meaning that SUS only needs to be run once to select several chromosomes. [1]

Tournament selection is a selection method that randomly selects a fixed number of chromosomes, the number of chromosome to select is called tournament size. From those chromosomes, the one with the best fitness score is chosen for crossover. This is repeated until the desired number of chromosomes are selected for crossover. [4]

In [18], a comparison between tournament, roulette wheel and Stochastic Universal Sampling selection is presented. The comparison is done by running three different populations in parallel on the same problem, each using one of the three selection methods. The results were obtained by comparing fitness score on the last generation. The result showed that the roulette wheel had the best fitness score and showed the most stable results.

To ensure that the best chromosomes remains in the population it can be directly transferred to the next generation. This method is called elitism.

Crossover

Crossover is when two or more chromosomes (called parents) from the current population creates one or more chromosomes (called children) for the next generation. This is done by, using some method, combining the genes in the parents. There are several methods and techniques to perform a crossover.

One of the first crossover method suggested was the 1-point crossover [30]. 1-point crossover will use two parents to breed two children. A point or index in the chromosome is selected, most often randomly. All genes before the point in the first parent will be copied to the first child, and all genes after the point in the first parent will be copied to the second child. The same method is then applied to the second parent. This will result in two children who both have different genes from each parent. See Figure 3.3 for a graphical representation of this example.

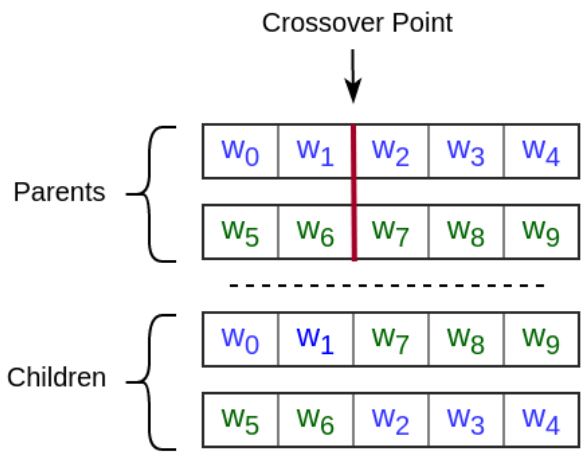

Crossover Point
Parents
Children
w_0 w_1 w_2 w_3 w_4
w_5 w_6 w_7 w_8 w_9
w_0 w_1 w_7 w_8 w_9
w_5 w_6 w_2 w_3 w_4

Figure 3.3: An example of a 1-point crossover procedure

The 1-point crossover can be generalized to a k-point crossover, where k points are selected and divided to the children in the same way. Traditionally, the chosen values are: $k = [1..2]$ [29]. See Figure 3.4 for an example. However, [20] highlights that a crossover with few points will lead to a positional bias, where genes who are closer located in the chromosome will have a higher chance of being passed on to the same child. In [29], an investigation of different values of k was performed. While there was no clear correlation between a higher k and better performance, the result showed that a low value of k (k = [1..2]) led to a worse result than higher values of k.

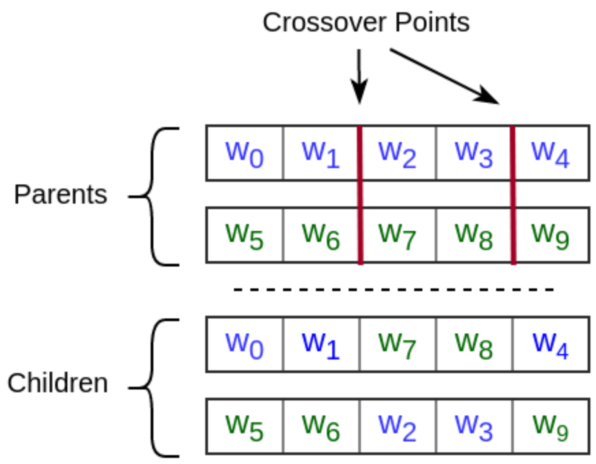

Crossover Points
Parents
w_0 w_1 w_2 w_3 w_4
w_5 w_6 w_7 w_8 w_9
Children
w_0 w_1 w_7 w_8 w_4
w_5 w_6 w_2 w_3 w_9

Figure 3.4: An example of a k-point crossover procedure, where n = 2

If each gene must be unique then a k-point crossover would be problematic, since there is a risk of a duplicate being transferred from the parents. This requirement is the case when a genetic algorithm is used to solve The Travelling Sales Problem. A *Cycle Crossover* is suitable [37]. The cycle crossover cycles the genes in each parent to create two children with the unique genes from each parent.

A diagonal crossover uses more than two parents to create children. See Figure 3.5 for a diagonal crossover where three parents produce three children. Practical evaluation of the diagonal crossover has shown that using more crossover points and more than two parents leads to better performance [19].

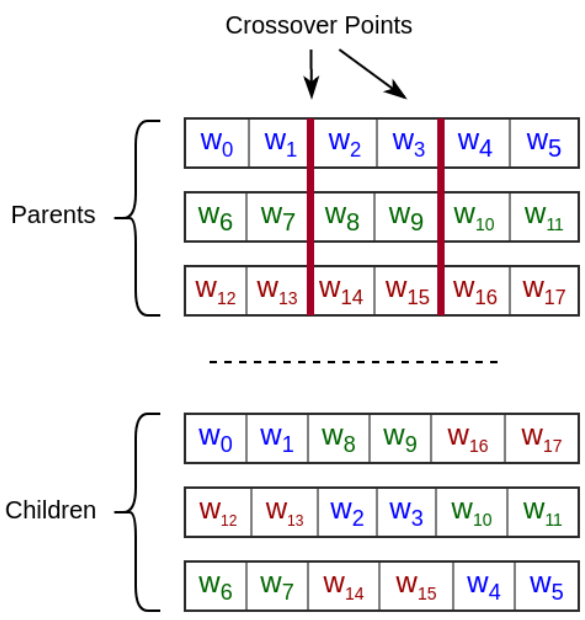


Figure 3.5: An example of a diagonal crossover with three parents and three children

A uniform crossover will, for each position in the parent, randomly selects which parent pass the gene to which children. This method eliminates the positional bias, since genes who are positioned closer in the chromosome do not have a higher probability of being transferred to the same children compared to genes who are further away.

Mutation

The mutation introduces randomness in the algorithm and is used to increase exploration by changing the value of a gene based on randomness. Even though the mutation rate is low, the mutation will make the genetic algorithm perform notably better compared to no mutation [38].

There are several mutation operators that can be used in a genetic algorithm. This section will present the following mutation types: bit-flip, uniform, polynomial and Gaussian.

A bit-flip operator can be used on chromosomes with binary representation. The bit-flip operator will consider each gene and using a given probability flip the bit (where 0 becomes 1 and vice versa). The probability for a flip can vary, [37] suggests a low value of the probability (around 0.001) and [11] uses a probability p based on the length of the chromosome n:

$$p = \frac{1}{n}$$

A uniform mutation uses the uniform distribution between the legal values of a gene to calculate the new value. It does not consider the current value of the gene.

A Gaussian mutation is a mutation type that performs the mutation of a gene using a Gaussian distribution, also called normal distribution. Once a gene has been selected for mutation, the value mutates using the following formula:

$$Gaussian(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where:

- x is the numerical value of the gene
- σ is the standard deviation
- μ is the mean of the distribution

A polynomial mutation uses the polynomial distribution to mutate the value of the gene. [15]

In early works of the Gaussian operator [28], it was compared to the bit-flip mutation. The comparison showed that the Gaussian mutation produced better results for most cases. Using a Gaussian mutator can not only enhance the quality but also the robustness of the genetic algorithm [21].

In [8], a comparison between uniform, polynomial and Gaussian operator was performed. The comparison was evaluated on an optimization problem using the criterias: efficiency, reliability and accuracy. The study showed that in terms of efficiency, uniform and Gaussian was the fastest to converge, with polynomial requiring longer time. In terms of reliability, the operators performed almost the same. In terms of accuracy, polynomial performed best, with uniform being second and Gaussian third. Another interesting finding the authors note is that using a mutation range that narrowed down through the generations was preferable for better results.

Termination Criteria

Termination criteria refers to the criteria that makes the genetic algorithm stop and return the chromosome with the highest fitness score. There are several ways to determine when to terminate a genetic algorithm. [37] claims that when the chromosome with the highest fitness score is stable and do not change for a number of generations, the algorithm should terminate. [27, 8] use a fixed number of generations before termination. In [43] the authors note that when using a fixed number of generations the number is often chosen by doing test runs to determine a suitable value.

3.3 Evaluation of Results

A/B testing is a test approach where two or more different variations of a system is tested on users. This is done by dividing the users into groups and then assign a variation of the system to each group. The aim for A/B testing is to evaluate what variation of the system is the most valuable, measured by some chosen metric. [45]

However, even though the groups A and B in A/B testing may be the same size, the individuals in the groups can never be the same. By exhibiting individual behavior, this will introduce randomness or noise in the data. When measuring a specific metric or parameter using A/B testing it is crucial to evaluate if the result measured is, by some probability, due to the variation between the groups or due to noise in the data. [45]

There are several statistical approaches to interpret data from an A/B test. Two common approaches are frequentist and Bayesian [16, 42]. Both frequentist and Bayesian approach is a parametric test, since it makes assumptions regarding the distribution of the data. *Bootstrap* is a statistical test that does not necessarily need to make any assumptions about the data [41].

A common usage of an A/B test is to evaluate the mean of a parameter. When evaluating the mean of a parameter the frequentist approach would be to use either a *t test* or a *z test*, depending on sample size [34]. The Bayesian approach would be to use the *Bayesian Estimation Supersedes the t test* (BEST) [33].

3.3.1 Frequentist Approach

In a frequentist approach the following steps can be performed to execute and evaluate an A/B test [44]:

- Choose what parameter to measure during the A/B test
- Formulate the *null hypothesis*
- Formulate the *alternative hypothesis*
- Choose value interval to reach *statistical significance*
- Calculate *p-value* to determine statistical significance

Null Hypothesis

A null hypothesis H_0 is the hypothesis that the test wants to disapprove. If the null hypothesis can be disapproved, the test has reached statistical significance and the data have a probability of being significant of some value p.[44]

If two groups are comparing a parameter μ using an A/B test, the null hypothesis could be:

$$H_0 : \mu_a = \mu_b$$

Alternative Hypothesis

An alternative hypothesis H_A is the hypothesis that the test aims to fulfill [44]. Using the same example as above, the alternative hypothesis would be:

$$H_A : \mu_a \neq \mu_b$$

Statistical Significance

If the null hypothesis can be disproven, the results are said to be statistically significant. The null hypothesis can only be disproven at a certain probability, called the $p - value$. How the p-value is calculated is dependent on the user group and what parameters to measure. E.g. if a mean of a parameter is calculated, depending on the size of the user group either a t test or a z test should be performed [34]. These tests will result in a p-value. What p-value interval that is acceptable is problem specific, although p < 0.05 is commonly used. However, the tests assumes that the data is normally distributed. [44]

There is controversy and much debate around statistical significance and its p-value. There is a misconception that the p-value is the probability of the null hypothesis being true with the given data [26], while it is the probability of the data given that the null hypothesis is true [12].

3.3.2 Bayesian Approach Using BEST

To use Bayesian statistics to evaluate an A/B test, the *Bayesian estimation supersedes the t test* (BEST) can be used. This section will present BEST, as described in the original paper [33]. The Bayesian approach is based on the Bayes's theorem [17]:

$$p(H|D) = \frac{p(H)p(D|H)}{p(D)}$$

Which can also be written with A and B, but since H corresponds to *hypothesis* and D to *data* it is more appropriate with this naming convention. Just like the frequentist approach, the Bayesian approach makes assumptions about the distribution of the data.

BEST uses as a prior, a distribution of allocation of credibility before the data is applied. The prior should either be a distribution from an earlier test or a distribution based on the characteristics of the observed data. Making this prior vague and wide will let the data control the Bayesian inference.

The observed data also needs to be represented using an appropriate distribution. The implementation referred to in the paper [33], as well as other implementations [14], uses a noncentral t-distribution to represent the observed data. However, [33] notes that it is important to use the distribution that is most appropriate to describe the observed data.

Using *Markov Chain Monte Carlo* (MCMC) the prior and observed data is applied, resulting in the posterior distribution. The result of BEST is two posterior distributions, one for group A and one for group B. To measure the *lift*, how much relatively better group B performed compared to group A, the following formula can be applied [14]:

$$lift = Mean(\frac{posterior_A - posterior_B}{posterior_B})$$

3.3.3 Bootstrap

Bootstrap does not make assumptions about the data, rather it creates a distribution by drawing samples from the data. By resampling the data with replacement, each sampling will create a mean value of the data. The means from each sample will generate a distribution over the means in the data. [41]

The approach is performed by the following steps [41]:

1. Repeat r times:
   a) Draw n samples from the data
   b) Calculate the mean of the n samples
2. Use the means to:
   - Draw histogram
   - Find confidence interval
   - Calculate standard deviation

3.4 Related Work

This section will present work relevant to the problem and approach for this study.

A meta-search engine combines search results from multiple search engines, aiming to provide better search results from different sources. This could be compared to the suggestion system of a browser, when combining suggestions from different sources. A challenge is to efficiently merge the search results from the different engines. [10] propose a model to merge the results using 5 different metrics: position in original list, number of duplications between search engines, the match between the search term and the content of the result, the capacity of the search members and if the result is in the users' interest area. The study outperformed regular search engines in the experiment, and placed similar to other meta-search engines. The paper concluded that when a user had their interests defined, the meta-search engine outperforms its competitors in terms of personalization and satisfaction.

In [39], the constants for a PID-controller is estimated using a genetic algorithm. The PID-controller is used to control the steering of a missile system. The constants have a binary representation. To select which chromosomes to proceed to the next generation a roulette wheel selection is used in combination with elitism. To breed new chromosomes, a multi-point crossover method is used. Finally, to mutate the new candidates a flip mutation is performed.

It is noteworthy that no related work that investigated weighting using constants on a type level was found.

4 Method

To be able to answer the research questions presented in Chapter 1, several results need to be obtained. This chapter will present the method used to obtain the results. It can be divided into several steps. An overview of these steps can be seen in Figure 4.1. The first step is to collect the necessary user data regarding user suggestions. Once the data is collected, an offline simulation can be performed, where a genetic algorithm was applied to estimate new weights for the system. These weights would then be applied to the browser during A/B testing, where the control group used the original unweighted system and the treatment group would be using the weights estimated by the genetic algorithm. Finally, the data from the A/B test can be evaluated, using BEST, to draw conclusions regarding the effects of the weights. This chapter will follow the same chronological order as presented in Figure 4.1 with the exception of starting off by describing *Error*, a key concept for the rest of the chapter.

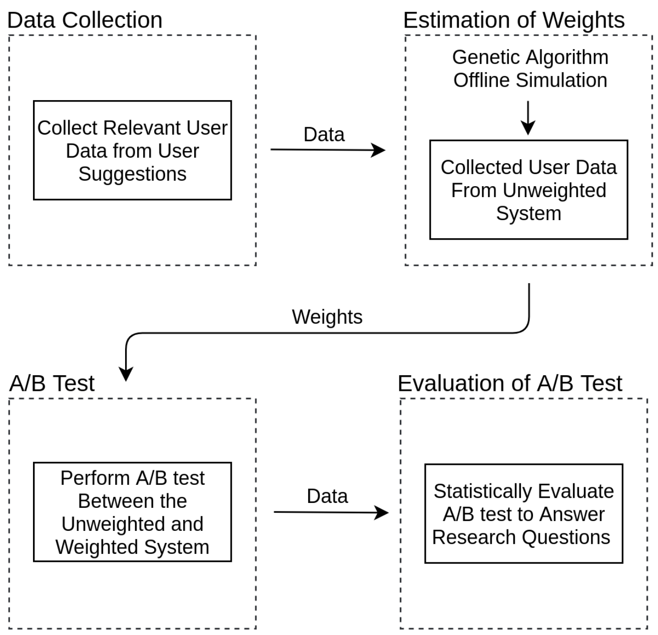
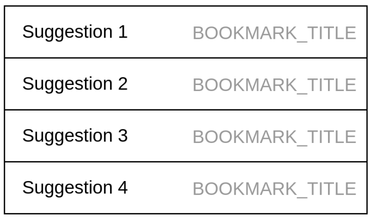

Data Collection
Collect Relevant User Data from User Suggestions
Data
Estimation of Weights
Genetic Algorithm Offline Simulation
Collected User Data From Unweighted System
Weights
A/B Test
Perform A/B test Between the Unweighted and Weighted System
Data
Evaluation of A/B Test
Statistically Evaluate A/B test to Answer Research Questions

Figure 4.1: Overview of how the method was performed

4.1 Error

The suggestion list presented to the user is sorted on suggestion score, where a higher score corresponds to a higher position in the list. When evaluating a suggestion list, it would be of interest to calculate the error of the suggestion list. Since it is known which suggestion was chosen, one could think it would simply be how many suggestions are placed above the correct suggestion. A well established measurement of error, also based on the position, such as MRR could be suitable. However, since this thesis do not consider the string-matching, and rather aims investigate different suggestion types, such a measurement could be misleading. Let's first consider an example in Figure 4.2, where all suggestions are of the same type.

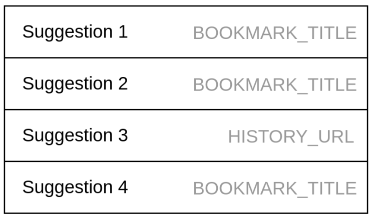


Suggestion 1	BOOKMARK_TITLE
Suggestion 2	BOOKMARK_TITLE
Suggestion 3	BOOKMARK_TITLE
Suggestion 4	BOOKMARK_TITLE

Figure 4.2: An example of a suggestion list with where all suggestions are of the same type

Using an error measurement based on position would yield a different error for each suggestion in Figure 4.2. However, the error measurement should be used when optimizing the different types of suggestions. In this example, the weights do not affect the positioning. Therefore, adjusting the weights would have no effect on the positioning. Using a error measurement such as MRR would yield a misleading error, since there is no type specific error in this example.

Let's consider another, but similar, suggestion list, presented in Figure 4.3. Examining the top two suggestions in the list, *Suggestion 1* and *Suggestion 2*, it can be seen that both are of the same type. As stated before, this means that only the string-matching score can differ and is the reason why one is placed higher than the other. Since only the types should be optimized, and not string-matching score, it would be wrong to value one of the suggestions over the other. If *Suggestion 3* was the correct suggestion, weighting the types differently could have impacted the ordering, potentially placing it above *Suggestion 2* and *Suggestion 1.* A final but important observation from the example is that the weighting could also affect the positioning of *Suggestion 4*, potentially placing it above *Suggestion 3* but never above *Suggestion 1* or *Suggestion 2.*



Suggestion 1	BOOKMARK_TITLE
Suggestion 2	BOOKMARK_TITLE
Suggestion 3	HISTORY_URL
Suggestion 4	BOOKMARK_TITLE

Figure 4.3: An example of a suggestion list with multiple suggestions of the same type

From the observations from the examples it can be concluded that suggestions below the correct suggestion are not off interest. It can also be concluded that suggestions above the correct suggestion of the same type are not of interest, since those suggestions can never change order. The suggestions of interest for the error measurement is suggestions above the correct suggestions of a different type. Using this logic, this thesis will let the error e of a suggestion with index x (where 1 is the topmost item) be the sum of all suggestions with a lower index that are of a different type. This can be described in the following formula:

$$e_x = \sum_{i=1}^{x} \begin{cases} 1, & \text{if } type_x \neq type_i \\ 0, & \text{otherwise} \end{cases}$$

where $type_x$ is the suggestion type of the suggestion with index x.

Going back to the suggestion list in Figure 4.3 and using the formula above, the error for the different suggestions can now be calculated. The error for the different suggestions can be seen in Table 4.1. This error will be the metric to evaluate the offline simulation.

Suggestion	Position/Index	Error
Suggestion 1	1	0
Suggestion 2	2	0
Suggestion 3	3	2
Suggestion 4	4	1

Table 4.1: Table with the error for the suggestions from Figure 4.3

4.2 Data Collection

In Chapter 2, it was explained that some data was logged when a request was made. The following data was sent and logged:

- Title of the correct suggestion
- URL of the correct suggestion

To be able to apply a genetic algorithm and to evaluate the results using A/B testing, additional data had to be sent and logged. The extended data collection was implemented so that the browser also sent the following data:

- Number of written characters
- List position of the correct suggestion
- Title of the correct suggestion
- URL of the correct suggestion
- A list of all suggestions from the top of the list down to a maximum of five suggestions below the correct suggestion, including the following information:
  - Type of match (see Table 2.1 for the different types)
  - Position in suggestion list
  - Suggestion score
  - Value of *base_<type>*
  - Value of *weight_<type>*

The data collection was made in two steps: one for the training data and one for the data from the A/B test. The collection was made on the same user group for both collections.

4.2.1 Training Data

To estimate new weights, training data was needed. The training data only needed to contain requests where the user picked a suggestion. Around 100,000 such requests could be collected each day. Due to time constraints it was decided that 500,000 requests were needed for the genetic algorithm. 500,000 also seemed like a reasonable amount of data to use for the genetic algorithm, to not make the execution time of it too long.

4.2.2 A/B Test Data

The A/B test data required to include all types of requests:

- Requests where the user has selected a suggestion
- Requests where the user has performed a search of the input string
- Requests where the user has entered a complete address

It was decided that as much data as possible were to be collected. There was a time constraint for the collection and it was decided that 8 days was sufficient to collect data for the A/B test.

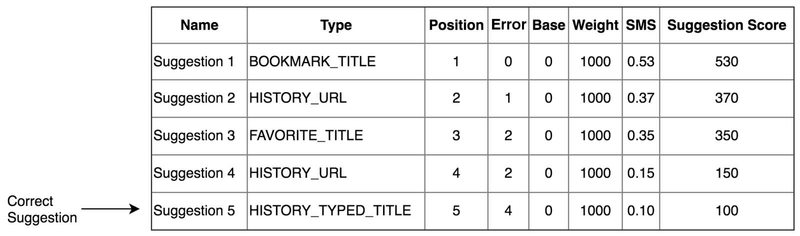
4.3 Offline Simulation

To facilitate the estimation of new weights using a genetic algorithm, a method to measure improvements when the weights are adjusted must exist. This section will describe how new weights could be tested on the user data to see how the error is impacted.

The user data contains the suggestion list presented to the user. It contains *base*, *weight*, *string-matching score (SMS)* and *suggestion score*. It also contains the position of the correct suggestion and the position of the other suggestions. With this information the error for each suggestion can be calculated, see Section 4.1.

4.3.1 Example of an Offline Simulation

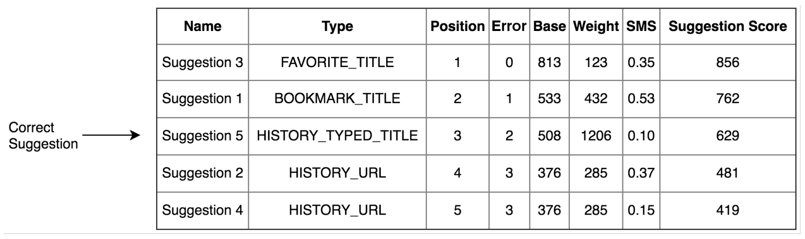
To describe how the offline simulation is performed more clearly, an example will be presented. In Figure 4.4 an instance from the user data is shown. It should be noted that base and weight are the same for all suggestion items, which corresponds to an unweighted system. This means that the position is only based on the string-matching score (SMS in the figure). The correct suggestion, the suggestion the user clicked, is *Suggestion 5*.


Correct Suggestion

Name	Type	Position	Error	Base	Weight	SMS	Suggestion Score
Suggestion 1	BOOKMARK_TITLE	1	0	0	1000	0.53	530
Suggestion 2	HISTORY_URL	2	1	0	1000	0.37	370
Suggestion 3	FAVORITE_TITLE	3	2	0	1000	0.35	350
Suggestion 4	HISTORY_URL	4	2	0	1000	0.15	150
Suggestion 5	HISTORY_TYPED_TITLE	5	4	0	1000	0.10	100

Figure 4.4: Example of an unweighted instance of user data used in the offline simulation

When the weights are changed by an optimization approach, the order of the suggestions needs to be changed. An example of new weights has been added to the example. How the new weights affect the user data can be seen in Figure 4.5. The string-matching score is unchanged, but the rest of the values has been affected. The position, and more importantly the error has changed for the suggestion items. The error of the correct suggestion has changed from 4 to 2, which is an improvement.


Correct Suggestion

Name	Type	Position	Error	Base	Weight	SMS	Suggestion Score
Suggestion 3	FAVORITE_TITLE	1	0	813	123	0.35	856
Suggestion 1	BOOKMARK_TITLE	2	1	533	432	0.53	762
Suggestion 5	HISTORY_TYPED_TITLE	3	2	508	1206	0.10	629
Suggestion 2	HISTORY_URL	4	3	376	285	0.37	481
Suggestion 4	HISTORY_URL	5	3	376	285	0.15	419

Figure 4.5: Example of an instance how the suggestions from Figure 4.4 are adjusted with new weights

This procedure can now be repeated for each user data entry and for all chromosomes in the genetic algorithm.

4.4 Estimation of Weights Using a Genetic Algorithm

This section will present how the genetic algorithm was used to optimize the weights. As the theory stated, the configuration of the genetic algorithm is problem specific. To evaluate what configuration was suitable an evaluation was done. This evaluation is presented in Appendix A. Configuration 2 had the best fitness score after the execution. Therefore, the configuration from Configuration 2 will be used.

4.4.1 Framework for Genetic Algorithm

In [22], a Python-based framework for evolutionary algorithms is presented. The framework, named Distributed Evolutionary Algorithm in Python (DEAP), is open source and aims to provide a toolbox for performing evolutionary algorithms, including genetic algorithms. The software is written to provide a toolbox where it is easy to extend or provide tools. Adding a problem specific representation or operator is therefore trivial. DEAP has been used in many other research projects [40, 47, 23].

4.4.2 Fitness Score

The fitness score used by the algorithm will be average error for a chromosome. The average error will be calculated on the training data set.

4.4.3 Representation

The weights are represented as numerical values. Each chromosome will consist of 16 numerical values, representing the different weights.

4.4.4 Selection

In the evaluation in Appendix A, tournament selection exhibits the best results. Therefore, tournament selection with a tournament size of 5 was used. Elitism with the best chromosome being transferred to the next generation without crossover or mutation was applied.

4.4.5 Crossover

Many crossover methods presented in the theory suffered from some degree of positional bias. To avoid the positional bias, a uniform crossover was chosen.

4.4.6 Mutation

The Gaussian mutation operator will be used. By using the Gaussian Mutation Operator the new value will be randomly chosen using a normal distribution (also known as gaussian distribution). The mean of the distribution will be the current value of the gene. The standard deviation was, through the evaluation in Appendix A, chosen to be 400.

4.4.7 Summary

The final configuration used in for the genetic algorithm can be seen in Table 4.4.7.

Type	**Value**
Population Size	20
Number of Generations	200
Chromosome Representation	Numerical
Fitness Score	Average Error
Selection	Tournament Selection, Size = 5
Crossover	Uniform Crossover
Mutation	Gaussian Mutation, $\sigma = 400$
Mutation Probability	0.01
Elitism	1

Table 4.2: Table with the final configuration for the genetic algorithm

4.5 A/B Testing

An A/B test are performed to measure and evaluate the effects of the using a weighted system, compared to an unweighted system. The groups consisted of a test group (weighted system) and a control group (unweighted system). The population of the test was the user base of the browser. New or existing users were not considered different. The users were randomly and evenly split. There existed countries where an A/B test was already in progress. Since A/B testing is about changing one feature and measure its effects, those countries was excluded from the A/B test performed in this thesis. No other exclusions or filtering of the user base was made. The following countries were excluded:

- USA
- Russia
- Ukraine
- Bangladesh
- Indonesia
- India

The parameters to be evaluated were:

- Number of characters written for a request
- The error of the correct suggestion for a request
- The position of the correct suggestion for a request
- The percentage of the requests being searches
- The percentage of the requests being addresses entered

From these two groups the average for each of the metrics can be calculated and compared. However, it is important to evaluate the results with a statistical approach, to evaluate if the results are significant or not. The evaluation was done using both Bootstrap and BEST. BEST was chosen since it is a recognized method to evaluate A/B tests and the posterior distributions are effective to compare. But BEST does require an assumption regarding the prior data and the distribution of the observed data. As will be described more detailed below, the assumptions were made to minimize the effect on the posterior distribution. To validate the results further, evaluation using Bootstrap was also done, a method not requiring any assumptions.

4.5.1 BEST

The test was performed once for each parameter to evaluate, with identical implementations. The implementation was fetched from [14] and modified to suit the data.

The prior mean distributions are represented as normal distributions, as [33, 14] suggests. The normal distribution is defined by a mean μ and τ. μ is calculated as the pooled mean for both the test group and the control group, meaning that μ will be the same for both groups. τ is calculated with the formula:

$$\tau = \frac{1}{\sqrt{100 * \text{pooled\_std}}}$$

The pooled standard deviation is multiplied with 100 to create a wide distribution.

The prior standard deviation distributions are represented as uniform distributions, as suggested by [33, 14]. Like the prior mean distributions, pooled values from both the test and control group were used. The uniform distribution is defined as:

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{for } a \geq x \geq b \\ 0, & \text{for } x < a \text{ or } x > b \end{cases}$$

The starting value a is set to:

$$a = \frac{\text{pooled\_std}}{1000}$$

The end value b is set to:

$$b = \text{pooled\_std} \cdot 1000$$

The prior distribution of the observed data is in both [33, 14] represented as noncentral t-distribution. However, a noncentral t-distribution do not represent every observed parameter in the data. A t-distribution is a continuous distribution, while the position, error, number of written characters, number of searches and number of entered addresses are discrete and non-negative. An intuitive distribution for a discrete and non-negative distribution is the Poisson distribution. However, the variance is greater than the mean, making the distribution overdispersed and a Poisson distribution is not possible. A suitable probability distribution for an overdispersed Poission distribution is a negative binomial distribution [31]. Therefore, the observed data for position, error, number of written characters, number of searches and number of entered addresses are modeled using a negative binomial distribution. MRR is modeled as a noncentral t-distribution since it is continuous.

Using the prior and the sampling algorithm *Markov Chain Monte Carlo* (MCMC) is performed to calculate the posterior distribution. The number of steps before the algorithm has reached a suitable convergence was set to 10,000, and then the algorithm went to 25,000 steps. These steps intervals were recommended in the implementation of [33, 14].

4.5.2 Bootstrap

Bootstrap was implemented as the steps described in Section 3.3.3. The number of iterations r was set to 1000, as in [41]. The entire data set was used each sample.

5 Results

5.1 Data Collection

The data collection was divided into two parts. The first part consisted of collecting data of the unweighted system to use as training data. The second part was the data collection for the A/B testing. The population of the data collection was the same for both parts.

5.1.1 Training Data

500,000 requests were collected as training data during a few days. Only requests where the user had chosen a suggestion was collected, no modification or filtering other than those stated was used when collecting the data. The distribution over the suggestion types shown and chosen can be seen in Figure 5.1. With 92.7 %, history is by far the most visible suggestion type. It is also the most chosen suggestion type, making up 90.6 % of the chosen suggestions.

Some suggestion types are visible much more frequently than they are chosen. 33.0 % of the visible suggestions are *history_title*, but is only chosen 16.5 % of all requests. Similar, 29.2 % of the visible suggestions are *history_url*, but only chosen 9.4 %. The opposite relationship exists, most notable for *history_typed_url* and *history_typed_title*, where is is visible 30.5 % and chosen 64.7 %.

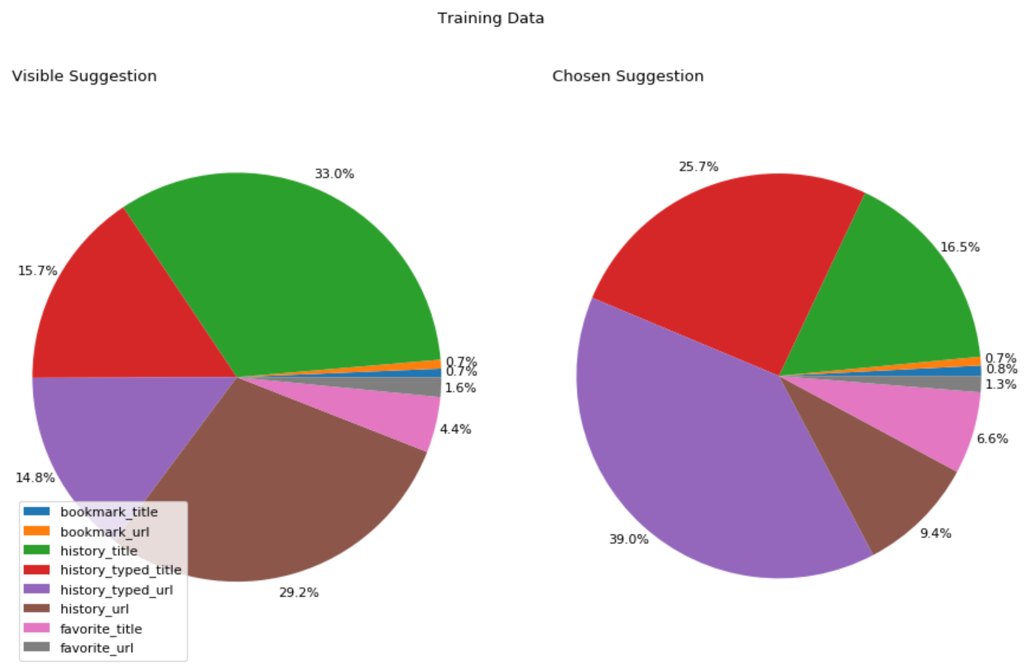


Figure 5.1: Pie chart of the distribution of the different visible suggestion types and the chosen suggestion types for the training data

5.1.2 A/B Data

The A/B data was collected during 8 days. No modification of filtering other than those stated was used when collecting the data. After the 8 days the groups consisted of:

- Group A (Control group):
  - 7,727,082 requests in total
    - 1,022,133 requests where a suggestion was chosen
    - 784,107 requests where the user entered the address
    - 5,912,135 requests where a user has searched
- Group B (Test group):
  - 7,753,232 requests in total:
    - 1,058,180 requests where a suggestion was chosen
    - 788,806 requests where the user entered the address
    - 5,897,305 requests where a user has searched

The distribution of the suggestion types shown and chosen for Group A (control group) can be seen in Figure 5.2. The distribution for both the visible and the chosen suggestion types are similar to the distribution presented in the training data and Figure 5.1.

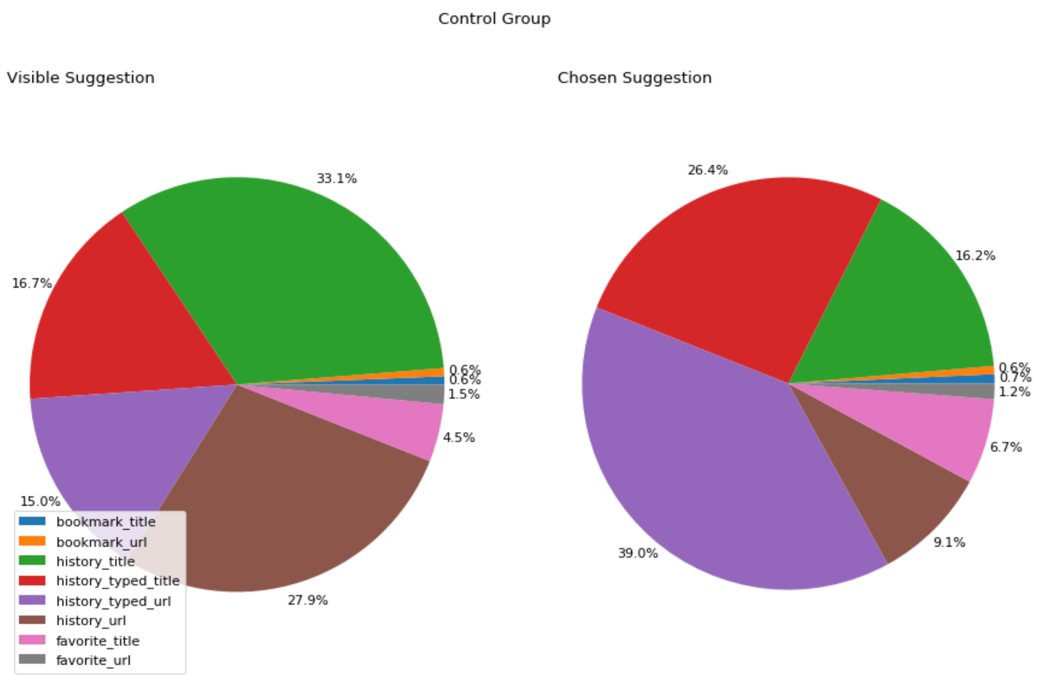


Figure 5.2: Pie chart of the distribution of the different visible suggestion types and the chosen suggestion types for the control group

The distribution of the suggestion types shown and chosen for Group B (test group) can be seen in Figure 5.3. These charts are not similar to the charts presented in Figure 5.1 and Figure 5.2. History is still the most shown and chosen suggestion type. However, *history_typed_url* is by far the most frequent suggestion type, making up 46.8 % of the visible suggestions and 67.5 % of the chosen suggestions.

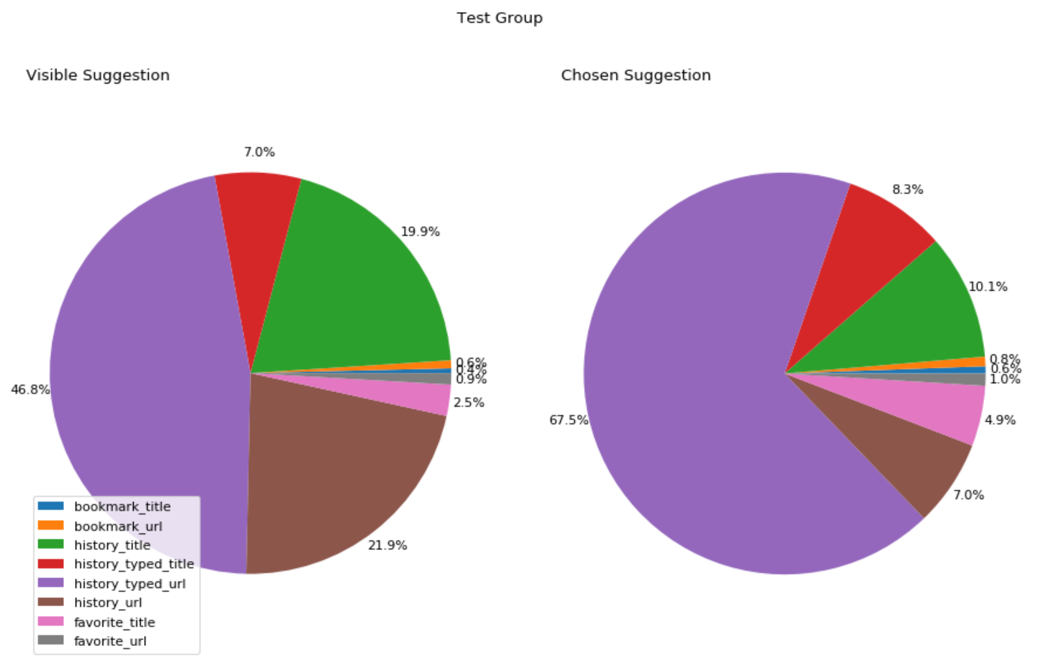


Figure 5.3: Pie chart of the distribution of the different visible suggestion types and the chosen suggestion types for the test group

5.2 Weight Estimations using Genetic Algorithm

The genetic algorithm used the configuration presented in Table 4.4.7. It was applied to the training data described in Section 5.1. The genetic algorithm optimized on the error presented in Section 4.1. In Figure 5.4 the error for each chromosome for every generation is presented. The population consisted of 20 chromosome, which means that there is a great deal of overlap present in the figure. The outliers are a result of crossover and mutation as the algorithm explores the search space.

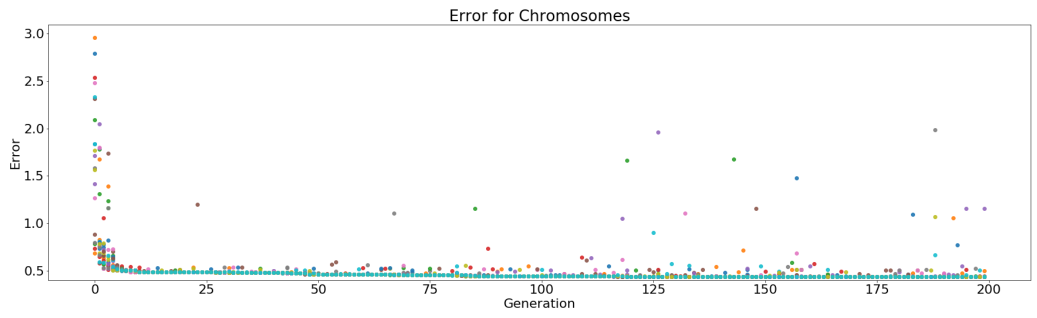


Figure 5.4: Plot of the error rate for each chromosome in each generation

In Figure 5.5 the mean error rate for each generation can be seen. As in Figure 5.4 there are generations that will have a significant increase in the error, a result of the algorithm exploring. The algorithm converges rapidly in the beginning, but keeps converging through out the generations.

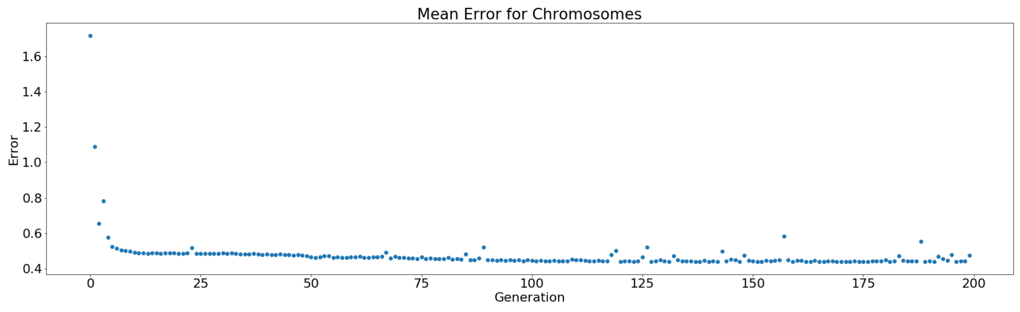


Figure 5.5: Plot of the mean error for each generation

In Figure 5.6 the best chromosome from each generation is presented. Since the algorithm uses an elitism of 1, meaning that the best chromosome always proceeds to the next generation without modification, the error rate for the best chromosome can never increase. The rapid convergence in the beginning can be seen here, as it could in Figure 5.4 and Figure 5.5. However, Figure 5.6 show how the progress slows down and almost comes to a stop. From the first generation to the 25th generation the error goes from 0.5624 to 0.4685, a decrease of 16.7 %. From the 175th generation to the 200th generation the error goes from 0.462318 to 0.46224, a decrease of 0.0169 %. While most of the progress occurs in the first generations, the progress has not entirely stagnated in the final generations.

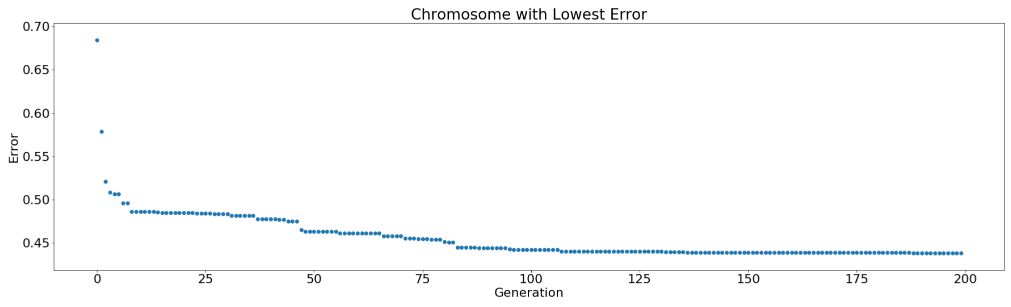


Figure 5.6: Plot of the chromosome with the lowest error for each generation

The weights of the best chromosome in the last generation can be seen in Table 5.1.

Type	**Value**
FAVORITE_TITLE_BASE	0
FAVORITE_TITLE_WEIGHT	1672
FAVORITE_URL_BASE	12
FAVORITE_URL_WEIGHT	1933
BOOKMARK_TITLE_BASE	33
BOOKMARK_TITLE_WEIGHT	1545
BOOKMARK_URL_BASE	31
BOOKMARK_URL_WEIGHT	2000
HISTORY_TITLE_BASE	144
HISTORY_TITLE_WEIGHT	444
HISTORY_URL_BASE	148
HISTORY_URL_WEIGHT	441
HISTORY_TYPED_TITLE_BASE	373
HISTORY_TYPED_TITLE_WEIGHT	278
HISTORY_TYPED_URL_BASE	590
HISTORY_TYPED_URL_WEIGHT	72

Table 5.1: Weights from the genetic algorithm, estimated using the training data

From the training data and the weights presented in Table 5.1, the performance metrics could be calculated to show how the training data was affected by the weights. The average error was calculated, this is the value that the GA optimized on. The average position was calculated, similar to the error a lower value corresponds to a higher position. Lastly, MRR was calculated, where a higher value is desired. The results can be seen in Table 5.2. Δ is the change when comparing the weighted against the unweighted. All metrics have been changed in the desired direction, as the position and error has decreased and MRR has increased. The largest improvement is the error, the metric that GA optimized on. Comparing position and MRR a large difference between the metrics are present. The difference between the metrics is that MRR values a change higher in the list more than a change further down the list.

Types	**Unweighted**	**Weighted**	**Δ**
Average Position	0.880316	0.783344	-11.02 %
Average Error	0.536162	0.439188	-18.09 %
MRR	0.794116	0.799871	0.72 %

Table 5.2: Results from the genetic algorithm, evaluated on the training data

5.3 Evaluation of A/B Test

This section will present the statistical evaluation of the A/B test. Both Bootstrap and BEST was performed on each metric, measuring and plotting if the results found in the A/B test is statistically significant.

5.3.1 Written Characters

The mean number of written characters was a parameter evaluated in the A/B test. The results were analyzed using BEST and Bootstrap. In Figure 5.7 the posterior distribution of the mean can be seen, as a result of BEST. In Figure 5.8 the mean distribution can be seen, as a result of Bootstrap. First thing to note is the similarity of distribution from the two methods. For numerical comparison see Table 5.3. Bootstrap has a wider distribution of both A and B. However, neither distributions have overlap between the two groups. Both test show that the results are statistically significant, and not a result of noise in the data.

Group	BEST	Bootstrap
A	3.9237	3.9241
B	3.7538	3.7536

Table 5.3: Table with the distribution means of written characters, from BEST and Bootstrap

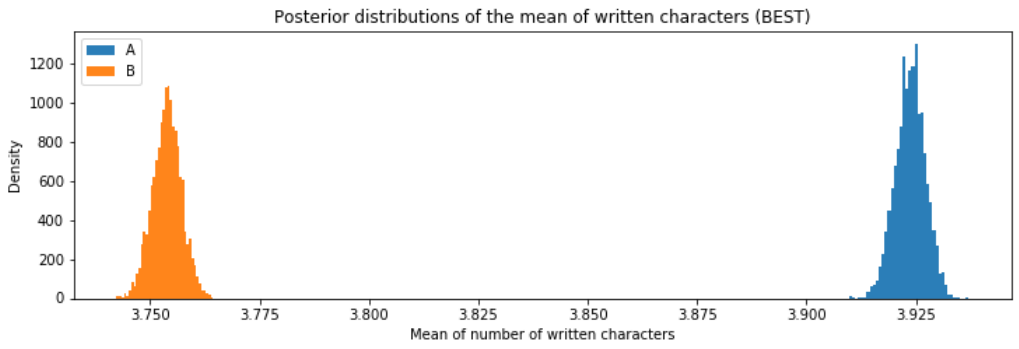


Figure 5.7: Posterior distribution of the mean of the written characters

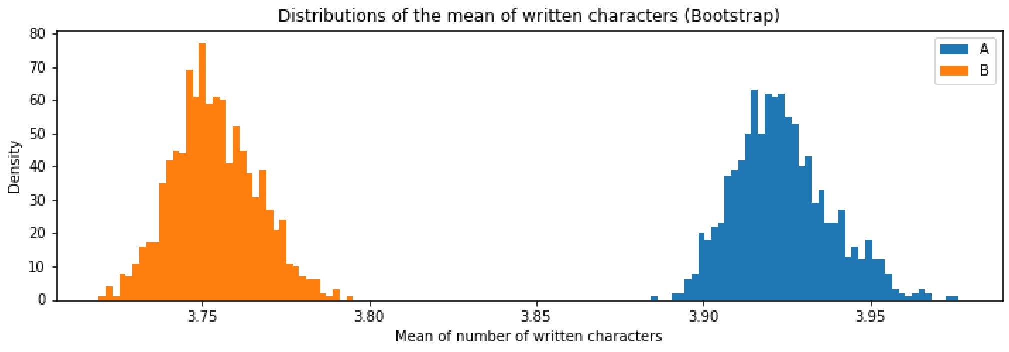


Figure 5.8: Histogram of the mean of written characters

In Figure 5.9 the posterior distribution of the relative decrease between the groups is presented. This distribution is calculated using the distributions produced by BEST, as can be seen in Figure 5.7. The mean of this distribution is 0.0433, which is the value that has the highest probability of being the increase between the groups. This corresponds to a decrease of written characters of 4.33 %.

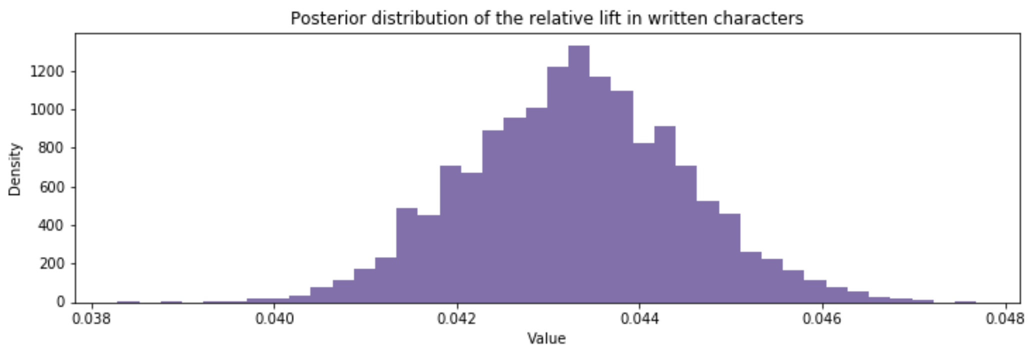


Figure 5.9: Posterior distribution of the relative decrease in written characters

5.3.2 Error Rate

The error rate, the number of suggestions above the correction suggestion of a different type, was measured during the A/B test. In Figure 5.10 the posterior distributions from BEST are presented. In Figure 5.11 the distributions from Bootstrap are presented. Comparing the means of the distributions in Table 5.4 it is clear that both approaches produce equivalent results. Neither BEST or Bootstrap show overlap of the distributions of group A and B, showing that the results are statistically significant.

Group	**BEST**	**Bootstrap**
A	0.5370	0.5371
B	0.2065	0.2065

Table 5.4: Table with the distribution means of error, from BEST and Bootstrap

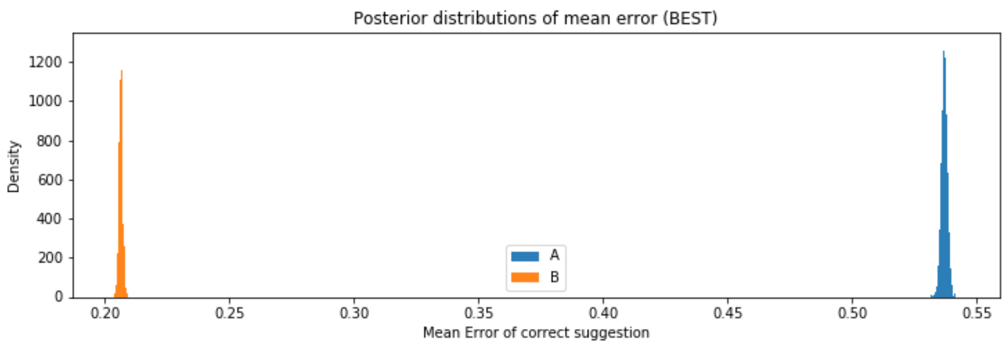


Figure 5.10: Posterior distribution of the mean error of the correct suggestion

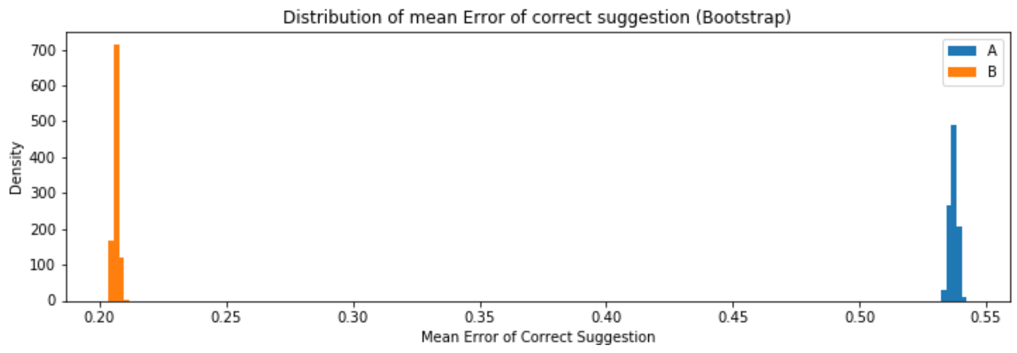


Figure 5.11: Histogram from Bootstrap of the mean of the error for the correct suggestion

The distribution of the relative decrease of the error between group A and B, as calculated using BEST, can be seen in Figure 5.12. The mean of the distributions is 0.6155, which is the value that has the highest probability of being the increase between the groups. 0.6155 corresponds to a decrease of 61.55 % in the error between group A and B.

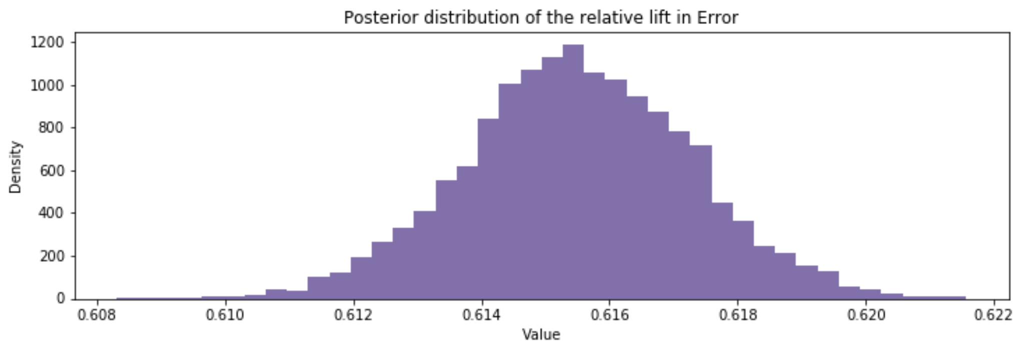


Figure 5.12: Posterior distribution of the relative decrease of the error

5.3.3 Index

Index, or position of the correct suggestion, was measured during the A/B test. The parameter was analyzed using BEST and Bootstrap. The posterior distributions for group A and B from BEST can be seen in Figure 5.13. The distributions for the same groups from Bootstrap can be seen in Figure 5.14. To compare the results of the tests, the mean of each distribution are presented in Table 5.5. With equivalent distributions mean between BEST and Bootstrap, the tests are equivalent. No overlap is present in the distributions for A and B, for both BEST and Bootstrap, resulting in the decrease being statistically significant.

Group	**BEST**	**Bootstrap**
A	0.8986	0.8986
B	0.6620	0.6619

Table 5.5: Table with the distribution means of index, from BEST and Bootstrap

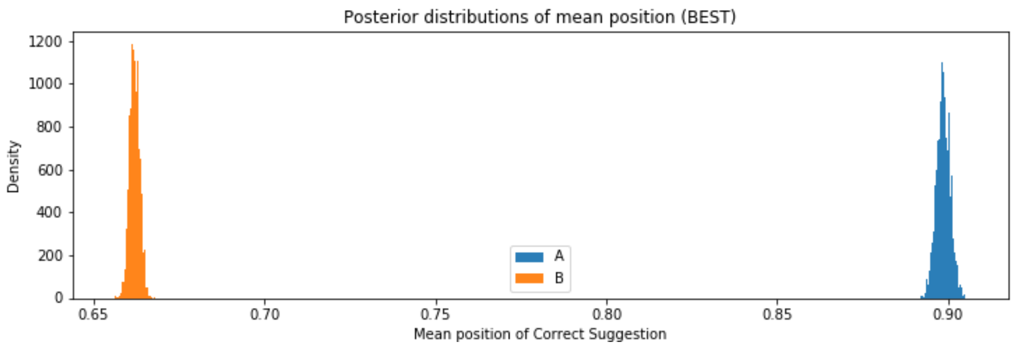


Figure 5.13: Posterior distribution of the mean index of the correct suggestion

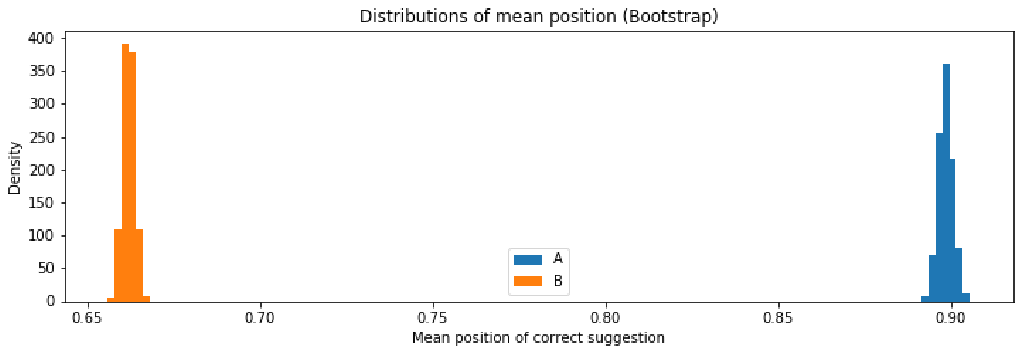


Figure 5.14: Histogram from Bootstrap of the mean of the index for the correct suggestion

Using the posterior distributions for group A and B, calculated using BEST and presented in Figure 5.13, the distribution of relative decrease was calculated. The distribution can be seen in Figure 5.15. The mean of the distribution is 0.2633, which is the value with the highest probability to correspond to the decrease between group A and B. 0.2633 corresponds to a decrease of 26.33 %.

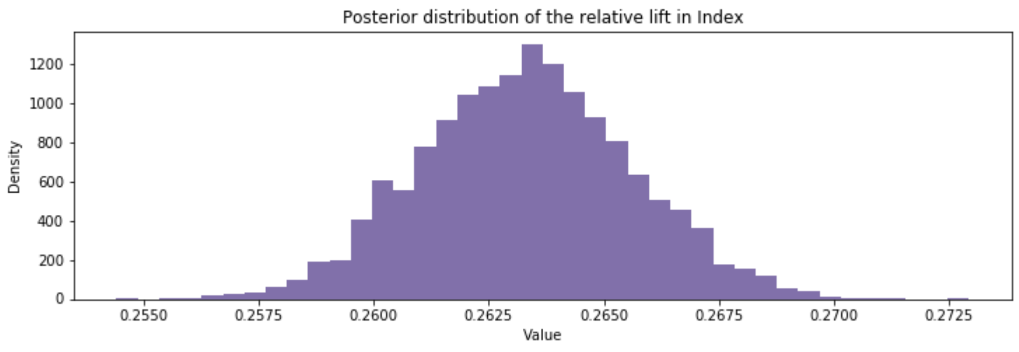


Figure 5.15: Posterior distribution of the relative decrease of index

5.3.4 MRR

The Mean Reciprocal Rank was measured during the A/B test. Using both Bootstrap and BEST, the parameter was analyzed. The posterior distributions for group A and B, analyzed by BEST, can be seen in Figure 5.16. The distributions for the same groups, analyzed by Bootstrap, can be seen in Figure 5.17. To compare the results of the tests, the mean of each distribution are presented in Table 5.6. The comparison show that the distributions are equivalent. No overlap is present in the distributions from A and B, for both tests, resulting in the increase being statistically significant.

Group	BEST	Bootstrap
A	0.7887	0.7885
B	0.8270	0.8270

Table 5.6: Table with the distribution means of MRR, from BEST and Bootstrap

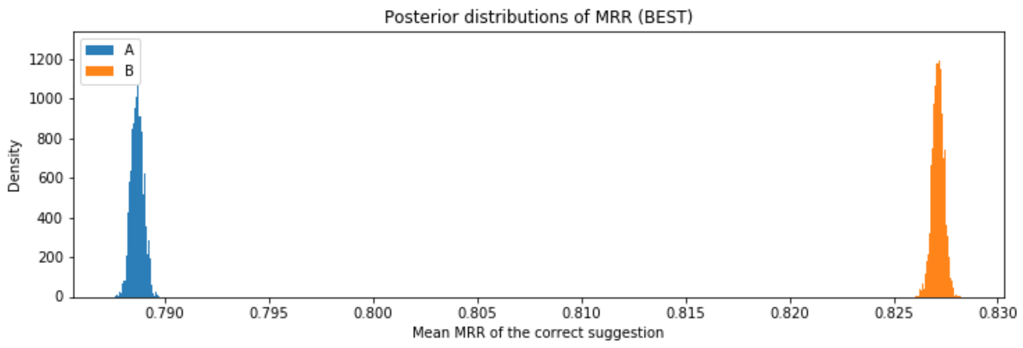


Figure 5.16: Posterior distribution of the MRR

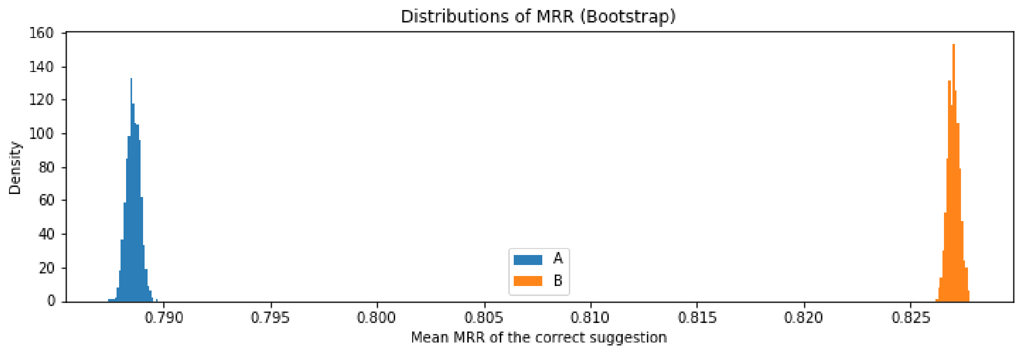


Figure 5.17: Histogram from Bootstrap of MRR

Using the posterior distributions for group A and B, calculated using BEST, the distribution of relative increase was calculated. The distribution is presented in Figure 5.18. The mean of the distribution is 0.04651, which is the value with the highest probability to correspond to the increase between group A and B. 0.04651 represents an increase of 4.65 %.

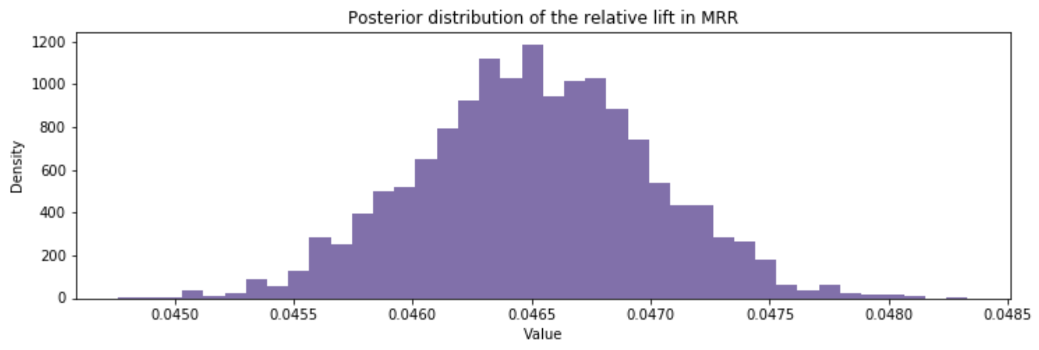


Figure 5.18: Posterior distribution of the relative increase of MRR

5.3.5 Search

The portion of requests that were searches was a parameter that was measured during the A/B test. The posterior distributions from group A and B, calculated using BEST, can be seen in Figure 5.19. The distributions from group A and B, calculated using Boostrap, can be seen in Figure 5.20. By comparing the means of the distributions in Table 5.7 for BEST and Bootstrap it is clear that both approaches produce equivalent results. The distributions for group A and B do not show any overlap, showing that the results are statistically significant.

Group	BEST	Bootstrap
A	0.7651	0.7606
B	0.7651	0.7606

Table 5.7: Table with the distributions of searches, from BEST and Bootstrap

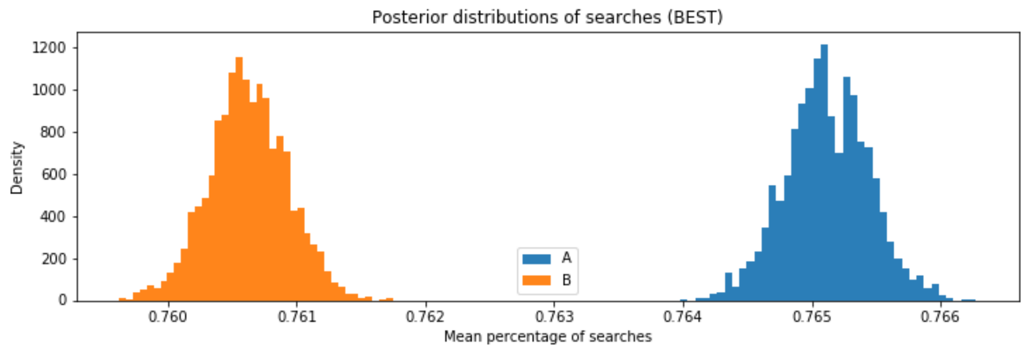


Figure 5.19: Posterior distribution of the mean number of requests being searches

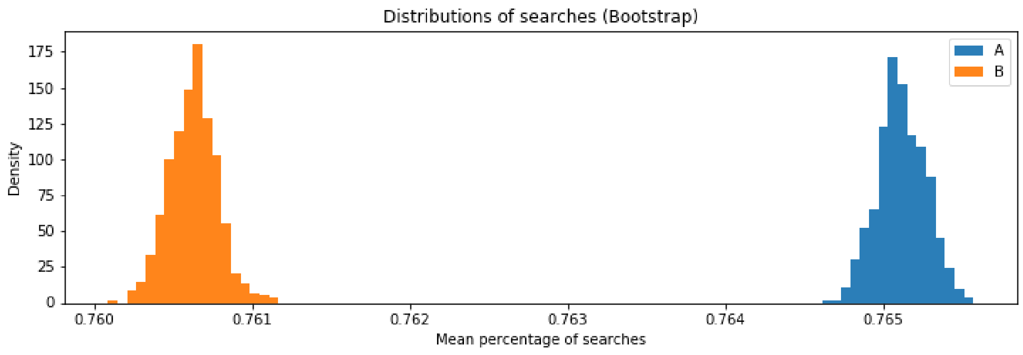


Figure 5.20: Histogram from Bootstrap of the mean of number of requests being searches

Using the posterior distributions from BEST, a distribution of the relative decrease of searches is calculated, a distribution that can be seen in Figure 5.21. The mean of the distribution is 0.00591, which is the value that has the highest probability of being the decrease between the groups. 0.00591 would correspond to a decrease of 0.59 %.

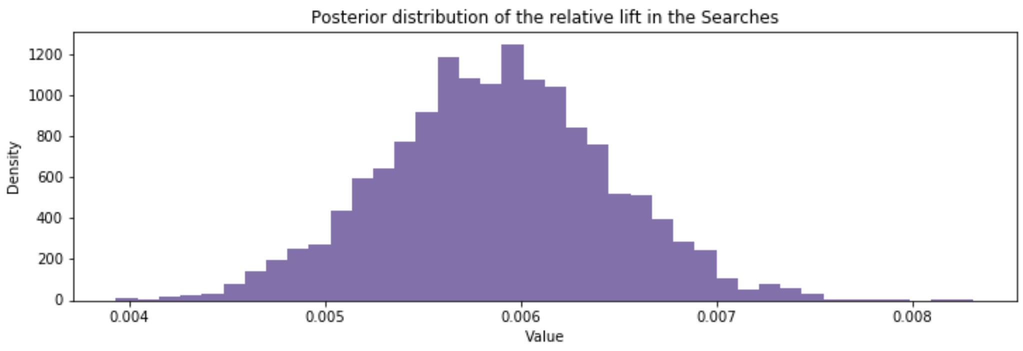


Figure 5.21: Posterior distribution of the relative decrease of search

5.3.6 Enter Addresses

The portion of requests that were a full entered address without the help from the suggestions was a parameter that was measured during the A/B test. The posterior distributions from group A and B, calculated using BEST, can be seen in Figure 5.22. The distributions from group A and B, calculated using Boostrap, can be seen in Figure 5.23. By comparing the means of the distributions in Table 5.8 for BEST and Bootstrap it is clear that both approaches produce equivalent results. The distributions show overlap, meaning that there is a possibility that the results found are a results of the variance within the groups (noise in the data) and not due to the weights.

Group	BEST	Bootstrap
A	0.1015	0.1017
B	0.1015	0.1017

Table 5.8: Table with the distributions of the enter adresses, from BEST and Bootstrap

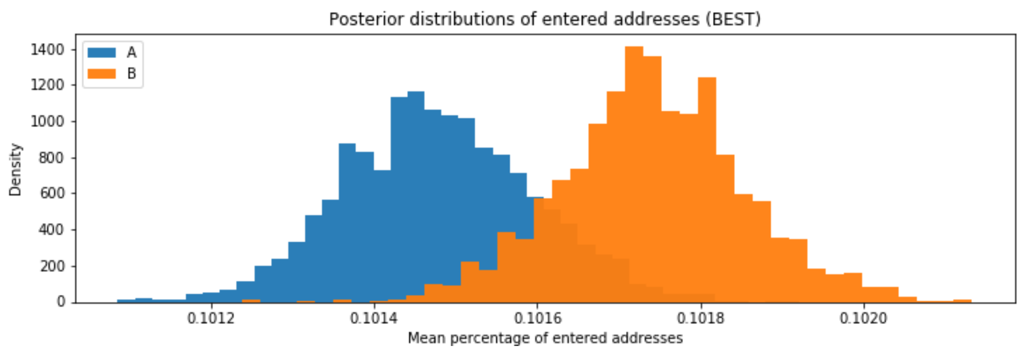


Figure 5.22: Posterior distribution of the mean number of requests being enter addresses

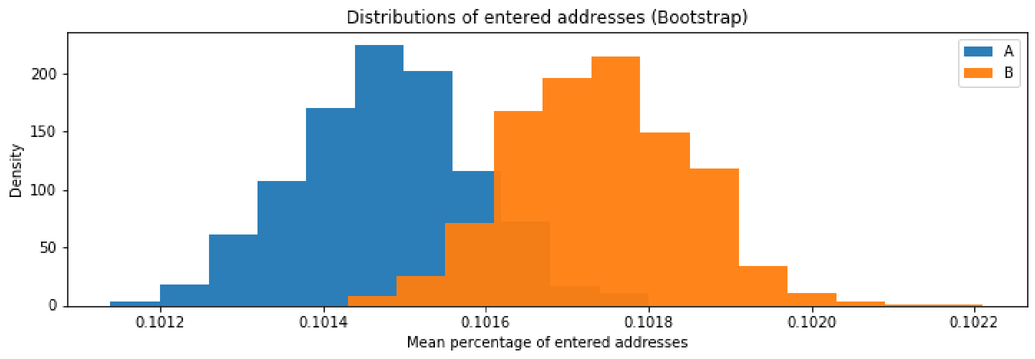


Figure 5.23: Histogram from Bootstrap of the mean of number of requests being entered addresses

Using the posterior distributions from BEST, a distribution of the relative increase of enter addresses is calculated, which can be seen in Figure 5.24. The mean of the distribution is 0.00258, which is the value that has the highest probability of being the decrease between the groups. 0.00258 would correspond to an increase of 0.26 %.

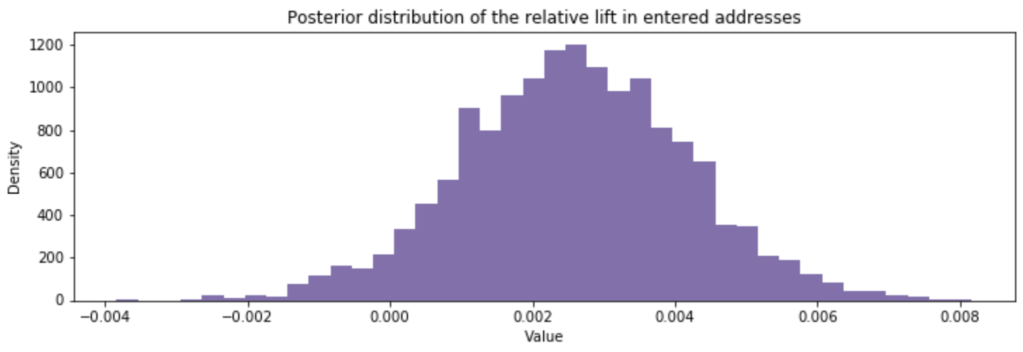


Figure 5.24: Posterior distribution of the relative decrease of search

5.3.7 Summary

In this section, the parameters from the A/B test have been analyzed. In Table 5.9 a summary of the analysis is presented. All metrics, except Search, showed an improvement in the test group compared to the control group. However, search did not show statistical significance, resulting in the metric being considered to be unchanged in the test group and control group.

Metric	**Increase**
Written Characters	-4.33 %
Error Rate	-61.55 %
Index	-26.33 %
MRR	4.65 %
Search Rate	-0.59 %
Entered Addresses	0.26 %

Table 5.9: Summary of the statistical analysis of the metrics

6 Discussion

6.1 Results

The result from the data collection was presented in Section 5.1. The training data consisted of 500 000 requests, which was a fixed number of requests to ensure that the execution time of the genetic algorithm was manageable. Data for the A/B test was collected during a fixed number of days, resulting in over 15 million requests. This was considered beforehand to be sufficient to draw conclusion about the groups. As the analysis and discussion about the results from the A/B test will show, the size of the data seems sufficient.

When analyzing the distribution of the different suggestion types there are several things worth discussing. Firstly, all three distributions (training data, control group and test group) exhibit the same behavior when looking at the suggestion type distribution overall. History is by far the most used suggestion type. Favorites and Bookmarks are a small part of the overall distributions of requests. History is a suggestion type that does not require direct user action to be saved, making it reasonable to assume that it could contain more sites than the other suggestion types. Due to privacy and anonymization, it was not possible to log how many suggestions existed in each suggestion provider. However, this suggests that different suggestion types have different importance to the user.

History is the largest provider overall, but there are several types of History suggestions. The two main types are: regular and typed history suggestions. The typed history suggestion indicates that the user, at least once, has written the entire URL to visit the site. Examining the distribution from the control group, presented in Figure 5.2, the regular history types are shown much more than they are chosen. *history_title* and *history_url* are combined 61.0 % of the visible suggestions, but only 25.3 % of the chosen suggestions. Examining the typed history the opposite pattern emerges. *history_typed_url* and *history_typed_title* are 31.7 % of the visible suggestions while they are 63.4 % of the chosen suggestions. This indicates that the typed history entries are more important to the user than the regular history. Typing an URL requires the user to know what site he/she wants to visit, while regular history can be a result of searching or clicking a link, which could be the cause for typed history to be more important.

Both the training data and the control group data are unweighted and collected on similar population. The distribution of visible suggestion types and chosen types were almost identical, which confirms that the population and the results from the training data are representative for the control group. Since the population of the training data collection was split randomly in half for the collection of A/B data, it can be assumed that the population and results are also representative for the test group.

The estimation of new weights was performed on the training data. The new weights are remarkably different from the unweighted system, suggesting that the genetic algorithm and the user data do not consider the different types to be equally important to the user. Highest weighted are favorites and bookmarks. Typed history is slightly higher weighted than regular history. By just examining the weights, an increase of bookmarks and favorites is expected. However, the distribution of visible and chosen suggestions for the test group indicates otherwise. Compared to the control group, visible bookmarks have decreased from 1.2 % to 1.0 % and favorites have decreased from 6.0 % to 3.4 %. Considering the weights, it is remarkable that this does exist in the distribution.

Another observation in the distribution of the test group is the increase of *history_typed_url*. 46.8 % of all visible suggestions are of that type, which is nearly half of all visible suggestions. The chosen suggestions are as high as 67.5 %, which was hard to predict from the weights and the distributions of suggestion types in the training data. It is important to reason about if the increase of a type is the direct result of the user data or if the genetic algorithms skewed the weights incorrectly. The error used to optimize the weights were based on how many suggestions of a different type were above the chosen suggestion. If only one type of suggestion was present, it would drastically decrease the error. And if only one type of suggestion was visible to the user, all chosen suggestions would be of that type. However, in the control group it was noted that *history_title* and *history_url* are 61.0 % of the visible suggestions, but only 25.3 % of the chosen suggestion type. This means that there is no consistent correlation between visible suggestion type and chosen suggestion type. Hence, 46.8 % of the visible suggestions in the test group being *history_typed_url* resulting in 67.5 % of the chosen suggestion could be an indication that a higher visibility leads more clicks, but it could also indicate that that the suggestion type is more important to the user.

The evaluation of the A/B test was done using both Bootstrap and BEST. Bootstrap and BEST presented distributions that were almost identical in all cases, indicating that the prior distribution chosen for BEST was suitable for the observed data. There were few unclear results from any of the parameters. *Search* and *Entered Address* showed little change, where entered address even had overlap in the distributions making the results inconclusive. The analysis indicated that the change of search and entered addresses are insignificant. However, the analysis of the other parameters strongly indicated that the results measured between the groups was the result of the change in the weights. There is a need to discuss what the results imply and why those results were obtained.

The largest improvement can be seen in the error rate, with a decrease of 61.55 %. Considering the genetic algorithm optimized only using the error, there is no surprise that this was the metric that improved the most. However, it does indicate that the genetic algorithm has optimized correctly. As discussed above, it is complicated to draw conclusions from the error alone, since a long list of few suggestion types would produce a low error even if the chosen suggestion was far from the top. But the position of the correct suggestion was also improved, a decrease of 26.33 %. This implies that the decrease of the error affected the positioning of the chosen suggestion.

MRR and position are both based on the same aspect of the suggestion list, but the results show very different improvements. To understand why this occurs, we must first consider how

MRR is calculated for a suggestion list:

$$MRR = \frac{1}{position_{chosen} + 1}$$

Where $position_{chosen}$ is the position of the chosen suggestion, and the addition in the denomiator is to compensate for the position being 0-indexed. If a change occurs higher in the list MRR will be affected much more than a change further down the list. Since the position improved more than MRR it can be assumed that the improvements of the positioning occurs further down the suggestion list.

The number of written characters decreased by 4.33 %. The number of written characters affect how much information the suggestion system receives from the user. With 4.33 % decrease of written characters the suggestion system has 4.33 % less characters to match to the suggestions. Hence, the improvements of the other metrics exists, even though the test group had less information from the users than the control group. A decrease of the number of characters shows that there is a relationship between when the user stops writing and the position of the correct suggestion. But comparing the position improvement of 26.33 % to the improvement in number of written characters of 4.33 %, it is clear that there exists no linear relationship between the metrics. But the literature showed that a user only looks at the top of the list. If the position is improved further down the list, then perhaps this improvement is not considered by the user. However, MRR values changes higher in the list above those occuring lower in the list. Comparing the improvement in MRR of 4.65 % to the improvement in number of written characters of 4.33 %, a clearer relationship exists. Since the number of written characters has been the main goal to improve, but cannot be optimized directly throught offline simulation, a metric that best corresponds to it is of interest. MRR seems to correspond better than both position and error, and it would be of interest to investigate if this insight could be used to optimize the weights more efficiently. A combination of MRR and Error could be calculated using:

$$MRR_{Error} = \frac{1}{Error + 1}$$

Which would utilize the type consideration from the error and the prioritization of highly placed suggestions from MRR. Further investigation on a metric utilizing both characteristics is of interest.

If a user does not consider any of the suggestions to be correct he/she can do a search of the input or enter the address manually. It could also be the case that the user did not consider the suggestions and only wanted to do a search or enter an address. But an increase or decrease of any of the two parameters could indicate how the suggestion system impacted the choice of doing a search or entering a address. The results show little to no change in the number of searches or entered addresses. The decrease of position and decrease of written characters did not convert the user to chose a suggestion instead.

6.2 Method

This section will discuss and criticize interesting aspects of the method presented in Chapter 4.

6.2.1 Optimization Algorithm

The estimation of weights in the suggestion system was seen as an optimization problem. The chosen algorithm to solve the problem was a genetic algorithm. It was noted that several algorithms exists. There were three algorithms considered for this task: Linear Regression (LR), Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). Before being able to compare and discuss the choice of optimization algorithm some theory regarding LR and PSO needs to be presented.

Linear Regression

Linear regression is a linear approach to quantify the relationship between variables. In the simplest case, where an input variable X is mapped to an output variable Y, the following formula is used:

$$Y = b_0 + b_1 X$$

Where b_0 and b_1 are the constants to be predicted. The linear regression approach will predict the straight line that describes the relationship of X and Y most accurately. [41]

The simple linear regression with one input variable can be extended to a multiple linear regression:

$$Y = b_0 + b_1 x_1 + b_2 x_2$$

Where Y is described by multiple input variables. [48]

The difference between the observed value in the data and the value fitted by the straight line is called *residual.* The residual is used when predicting the straight line. For simple regression with one input variable and one output variable, *residual sum of squares* (RSS) is minimized. RSS is defined as:

$$RSS = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Where n is the number of observed values, Y_i is the observed value from the data and $\hat{Y}_i$ is the value predicted by the linear regression. For a multiple regression model *root mean squared error* (RMSE) is used. RMSE is an extended version of RSS:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2} = \sqrt{\frac{1}{n} RSS}$$

Multi-output regression predicts the relationship between multiple input variables and multiple output variables. There is an excessive amount of approaches to achieve this. A state-of-the-art study from 2015 [7] present different approaches as well as commonly used performance evaluation measures.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is search-based metaheuristic. The algorithm is based on the idea of swarm intelligence, where a swarm of particles cooperates and searches for a solution. Each particle in the swarm consists of [6]:

- Position
- Velocity
- Personal best position

Each particle also has access to the best position found by the swarm, called the global best position. To update the swarm to converge to a solution the following steps are performed [5]:

1. Evaluate fitness score of each particle
2. Update the individual and the global best position

3. Update velocity for each particle
4. Update position for each particle

After a fitness function has been decided, step 1 and 2 are trivial to perform. The key part of PSO lies within updating the velocity. The new velocity is based on the current velocity (*inertia*), the best position for the particle and the best position for the entire swarm. A randomness is also applied to the velocity calculation to increase exploration. Formally, the velocity is described as [5]:

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x_i}(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

Where v_i is the velocity of particle i. w, c_1 and c_2 are chosen constants. r_1 and r_2 are random values. x_i is the position of the particle and $\hat{x_i}$ is the position of the best position for the particle. g is the best global position.

PSO is a widely used algorithm, with applications is multiples areas. [49] present a state-of-the-art survey on PSO and its applications. A drawback compared to genetic algorithms is the lack of a crossover operation prohibits the sharing of properties between candidates.

Choice of Optimization Algorithm

There were several factors to considered when choosing optimization algorithm. This section will discuss and compare the different algorithms.

LR maps the linear relationship between output variables and input variables. In the case of a suggestion system the weights are the input variables and the measurement of error is the output variable. It is known that the relationship between the weights and the suggestion score is linear, suggesting that a linear relationship between weights and the error would be suitable. However, the data collected before weighting does not vary the weights. The *base* weights, making up half of the weights, are set to 0 in the training data. If LR would be used to optimize the problem with 16 weights the following formula could be used:

$$Y = b_0 + b_1 X_1 + ... + b_{16} X_{16}$$

It is obvious that if half of the weights X in the formula are set to 0 in all of the training data and the other weights would not vary, information would be lost.

PSO and GA are both metaheuristics that uses different techniques to search the problem space. PSO is a simpler algorithm with less steps, making the implementation lighter. PSO also tries to explore the space close to the best solution, as the particles move towards the solution. One drawback with PSO that the literature raises is that particles do not share properties between each other, while GA uses crossover to achieve the sharing. GA consists of several parts and for each part there are several approaches to choose from. Each part should be chosen to suit the specific problem. However, it is only the fitness score that are completely problem specific. In PSO the velocity and the fitness score are the parts that need to be problem specific. PSO will always save the best global position of the particles, ensuring that the best solution will never be lost during the algorithm. This can be achieved in GA using elitism.

LR had several problems in this domain, and was therefore not chosen. Choosing between PSO and GA was not trivial. Both algorithms are very similar, and there are no clear problems or advantages with any of the algorithms. But GA had some characteristics that made it a better choice. Firstly, the open source project DEAP had implementations of several different approaches to each step of the GA. This opened the possibility to test several approaches on the test data without requiring a substantial amount of work. Secondly, since GA had several independent steps in the algorithm, compared to PSO which is only dependent only on the velocity, it seemed like a less narrow algorithm.

6.2.2 Genetic Algorithm

A discussion of why the genetic algorithm was chosen has already been made. This section will focus on the discussion regarding the choices made within the algorithm.

One of the advantages of GA is that it does not require much adaptation to the specific problem. The problem domain, especially the relationship between the weights and the error, was unexplored and partly unknown. Therefore, making problem specific assumptions was not desirable and thus avoided.

There are several selection methods that seemed suitable for this problem. After reviewing the literature, tournament selection and roulette wheel selection were selected to be evaluated. The literature pointed towards the roulette wheel to be the better of the two. But the evaluation, as presented in Appendix A, saw that tournament selection performed better. Evaluating the results from the appendix, it can be seen that roulette wheel exhibits much more randomness (exploration) and converged poorly.

Several crossover methods were presented in the theory, several based on the k-point crossover. When a crossover point exists, a positional bias also exists. The probability of the first gene and last gene of a chromosome ending up in the same child is very low, but the probability increases as the genes are closer to each other. This would cause the positioning of the weights in a chromosome to affect the results. Since the main goal of the GA was to find the relationship between the weights, an existing implicit relationship was not desired. To avoid the positional bias the uniform crossover was selected.

The mutation probability for a gene was chosen through testing. Literature suggested a small value, for a bit flip mutation operator the recommended interval would be between 0.001 and 1/16. A bit flip mutation operator is not possible for a numerical representation but its recommended interval can serve as a guideline for mutation probability. Through testing of different values of the mutation probability it was concluded that 0.01 gave best results on the training data, and it also lies within the recommended interval.

The genetic algorithm did prove to be an effecient algorithm for this problem. However, there are opportunities of improvement. The termination criteria was chosen so that each execution had the same prerequisites and the duration of the execution was the same. The execution time was several hours, restricting the number of executions. An interesting termination criteria would be to wait until the algorithm had converged to some solution. That termination criteria could potentially change the results in the configuration evaluation.

6.2.3 Error

The error rate measurement was the foundation for the genetic algorithm, since it was the only parameter the algorithm was using to optimize the weights. These weights was later used to investigate if the position of the chosen suggestion affects typing. Therefore, it was of high importance that the error measurement correlated with the position of the chosen suggestion. It was considered that a suitable measurement would be the position of the correct suggestion or a well established measurement of error, such as the MRR. Since MRR did not consider the types, and no other measurement found in the literature exhibited the desired behavior, the error measurement was created.

As discussed in Section 6.1, MRR seems to correspond better to the number of written characters than both error and position. Since it was unknown before this study what optimization measurement was going to prove most efficient, it would have been of interest to investigate multiple measurements in the study. It would have been possible to run the GA several times using different optimization measurement. To evaluate the results an A/B/C/D... test could have been created with a suitable amount of groups. The results could have been evaluated similarly to the current evaluation, but conclusions regarding the best metric could have been made. Unfortunately, the time constraint did not allow for this to be explored.

6.2.4 Evaluation of A/B Testing

Both BEST and Bootstrap were chosen to analyze the results from the A/B test. Bootstrap does not assume the distribution of the observed data, it was chosen to validate if the assumptions made when using BEST were correct. BEST was chosen since it provides several advantages: it is intuitive to analyze if the posterior distributions show statistical significance and the posterior distributions can be used to calculate the relative increase between the groups. Using two different approaches was time consuming, requiring several hours for each metric. But using two independent approaches to validate each other provided a higher confidence in the results.

Bootstrap used 1000 samples during sampling. MCMC used 10 000 samples to converge and then the following 15 000 samples as the result. It would have been of interest to run bootstrap with a similar number of samples. This would have required much more execution time, but the y-axis (density) would have been the same scale for both analysis, enabeling a better comparison.

6.3 The work in a wider context

This study dealt with the collection and handling of user data. It was of critical importance to not intrude on the users' privacy. The collection of data was extended, meaning that more data was collected. It was a challenge to collect more data without making the user less anonymous. This also restricted how the data could be utilized. No user identification in any form was logged with the request, making it impossible to map or track a specific user through several requests. Only a stripped version of the URL was saved for the chosen suggestion, and no URL was saved for the other suggestions in the list. Only the number of written characters was logged, and not what the user actually typed. In the balancing of user experience and user privacy this study would like to claim that it always favored privacy, only improving user experience if privacy was not affected.

Basing behavior on user data of course means that the users affect the behavior, but it also implies that a user could purposely generate data to alter the behavior for other users. This behavior change could then have several other implications, something that the browser could be held accountable for. Most recently, Facebook was heard in the supreme court concerning if the content spread through their site, by users, are their liability or not[1]. If the suggestion system would work on a site level, ranking individual sites based on requests, a user could generate vast amount of requests to a certain site. But when the suggestion system uses the user data on a type level, only sites from that certain user are presented. Therefore, it is impossible to trick a user to a certain site by altering the user data. It could be possible to send vast amount of a certain suggestion type, making its weight heavier. But this would not enforce content to other users.

[1]https://www.cnbc.com/2018/10/16/supreme-court-case-could-decide-fb-twitter-power-to-regulate-speech.html, accessed: 12-12-2018

7 Conclusion

7.1 Research Questions

Research Question 1
How can the suggestion error rates be decreased by the introduction of weights in the system?
By optimizing on the error rate in the training data new weights were estimated. The A/B test showed that by introducing weights in the system the error rate was substantially reduced, in this study the error rate was reduced by 61.55 %. This study concludes that the introduction of weights can provide a great improvement in the error.

Research Question 2
How will users' typing be affected by improving the position of the correct suggestion?
As a result of the introduction of weights the position improved by 26.33 %. By improving the position of the correct suggestion an improvement of the users' typing occured, with a decrease of 4.33 %. This study concludes that there is a correlation between how much a user types and the position of the desired suggestion. This finding shows that the user values the suggestions while typing, terminating earlier if the desired suggestion is presented.

Research Question 3
How can the position improvement of the correct suggestion increase the usage of suggestions, as measured by the percentage of requests being an entered address or a search?
The introduction of weights to the suggestion system resulted in a decrease of typing and an improvement in position. This means that the suggestion system is able to present the desired suggestion higher in the suggestion list with less input. However, it can be concluded that the percentage of searches and entered addresses was not affected by these improvements. This study concludes that no correlation was found between improving the suggestion position and how often the users searched or entered an address.

7.2 Future Work

In this study there are aspects which could be improved and explored in future work. MRR was mentioned to correlate better to the reduction of typing than the position or the error. It has been discussed if this could be the result of both MRR and the user prioritizing the top suggestions. In Chapter 6 a proposal of a combination of MRR and the error was presented and discussed. In terms of the weights optimization there are several algorithm to evaluate, some mentioned in this thesis. This study did not aim to find the most efficient optimization algorithm for this problem, rather to investigate how an improvement would affect the system. Now that it is concluded that the weight optimization is of importance, there is a need to evaluate which optimization algorithm yields the best results.

Bibliography

[1] James E. Baker. “Reducing Bias and Inefficiency in the Selection Algorithm”. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*. Cambridge, Massachusetts, USA: L. Erlbaum Associates Inc., 1987, pp. 14–21. ISBN: 0-8058-0158-8. URL: `http://dl.acm.org/citation.cfm?id=42512.42515`.

[2] Eytan Bakshy, Dean Eckles, and Michael S. Bernstein. “Designing and Deploying Online Field Experiments”. In: *Proceedings of the 23rd International Conference on World Wide Web*. WWW ’14. Seoul, Korea: ACM, 2014, pp. 283–292. ISBN: 978-1-4503-2744-2. DOI: `10.1145/2566486.2567967`. URL: `http://doi.acm.org/10.1145/2566486.2567967`.

[3] Patti Bao, Jeffrey Pierce, Stephen Whittaker, and Shumin Zhai. “Smart Phone Use by Non-mobile Business Users”. In: *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. MobileHCI ’11. Stockholm, Sweden: ACM, 2011, pp. 445–454. ISBN: 978-1-4503-0541-9. DOI: `10.1145/2037373.2037440`. URL: `http://doi.acm.org/10.1145/2037373.2037440`.

[4] Tobias Blickle and Lothar Thiele. “A Mathematical Analysis of Tournament Selection”. In: *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995, pp. 9–16.

[5] James Blondin. *Particle Swarm Optimization: A Tutorial*. 2009.

[6] Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Xiaodong Li. “An analysis of the velocity updating rule of the particle swarm optimization algorithm”. In: *Journal of Heuristics* 20.4 (2014), pp. 417–452. ISSN: 1572-9397. DOI: `10.1007/s10732-014-9245-2`. URL: `https://doi.org/10.1007/s10732-014-9245-2`.

[7] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larrañaga. “A Survey on Multi-output Regression”. In: *Wiley Int. Rev. Data Min. and Knowl. Disc.* (2015), pp. 216–233. ISSN: 1942-4787. DOI: `10.1002/widm.1157`. URL: `http://dx.doi.org/10.1002/widm.1157`.

[8] Razvan Cazacu. “Comparative Study between the Improved Implementation of 3 Classic Mutation Operators for Genetic Algorithms”. In: *Procedia Engineering* 181 (Dec. 2017), pp. 634–640.

[9] Li Chen and Pearl Pu. “Users’ eye gaze pattern in organization-based recommender interfaces”. In: *IUI*. 2011.

[10] X. Chen, Q. Li, Y. Lin, and B. Zhou. "A synthesized method of result merging in meta-search engine". In: *2017 10th International Conference on Human System Interactions (HSI)*. 2017, pp. 206–211. DOI: 10.1109/HSI.2017.8005030.

[11] Francisco Chicano, Andrew M. Sutton, L. Darrell Whitley, and Enrique Alba. "Fitness Probability Distribution of Bit-Flip Mutation". In: *CoRR* abs/1309.2979 (2013). arXiv: 1309.2979. URL: http://arxiv.org/abs/1309.2979.

[12] Jacob Cohen. In: *American Psychologist* (1994).

[13] Nick Craswell. "Mean Reciprocal Rank". In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 1703–1703. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_488. URL: https://doi.org/10.1007/978-0-387-39940-9_488.

[14] Cameron Davidson-Pilon. *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference*. 1st. Addison-Wesley Professional, 2015.

[15] Kalyanmoy Deb and Debayan Deb. "Analysing mutation schemes for real-parameter genetic algorithms". In: *International Journal of Artificial Intelligence and Soft Computing* 4 (2014), pp. 1–28.

[16] Zoltan Dienes. "Bayesian Versus Orthodox Statistics: Which Side Are You On?" In: *Perspectives on Psychological Science* 6.3 (2011), pp. 274–290. DOI: 10.1177/1745691611406920.

[17] Allen B. Downey. *Think Bayes: Bayesian Statistics Made Simple*. Needham, MA, USA: Green Tea Press, 2012. URL: http://www.greenteapress.com/thinkbayes/.

[18] Asti Dwi Irfianti, Retantyo Wardoyo, Sri Hartati, and Endang Sulistyaningsih. "Determination of Selection Method in Genetic Algorithm for Land Suitability". In: 58 (Jan. 2016), p. 03002.

[19] A. E. Eiben and Cees H.M. van Kemenade. "Diagonal Crossover in Genetic Algorithms for Numerical Optimization". In: *Journal of Control and Cybernetics* 26 (1997), pp. 26–3.

[20] Larry J. Eshelman, Richard A. Caruana, and J. David Schaffer. "Biases in the Crossover Landscape". In: *Proceedings of the Third International Conference on Genetic Algorithms*. George Mason University, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 10–19.

[21] Yanhong Feng, Juan Yang, Congcong Wu, Mei Lu, and Xiang-Jun Zhao. "Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation". In: *Memetic Computing* 10.2 (2018), pp. 135–150. ISSN: 1865-9292. DOI: 10.1007/s12293-016-0211-4. URL: https://doi.org/10.1007/s12293-016-0211-4.

[22] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. "DEAP: Evolutionary Algorithms Made Easy". In: *Journal of Machine Learning Research* 13 (2012), pp. 2171–2175.

[23] Félix-Antoine Fortin, Simon Grenier, and Marc Parizeau. "Generalizing the Improved Run-time Complexity Algorithm for Non-dominated Sorting". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. GECCO '13. Amsterdam, The Netherlands: ACM, 2013, pp. 615–622. ISBN: 978-1-4503-1963-8. DOI: 10.1145/2463372.2463454. URL: http://doi.acm.org/10.1145/2463372.2463454.

[24] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN: 0201157675.

[25] Carlos A. Gomez-Uribe and Neil Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation". In: *ACM Trans. Manage. Inf. Syst.* 6.4 (2015), 13:1–13:19. ISSN: 2158-656X. DOI: 10.1145/2843948. URL: http://doi.acm.org/10.1145/2843948.

[26] Krauss Stefan Haller Heiko. “Misinterpretations of Significance: A Problem Students Share with Their Teachers?” In: *Methods of Psychological Research Online* (2002).

[27] Denny Hermawanto. “Genetic Algorithm for Solving Simple Mathematical Equality Problem”. In: *CoRR* abs/1308.4675 (2013). arXiv: `1308.4675`. URL: `http://arxiv.org/abs/1308.4675`.

[28] R. Hinterding. “Gaussian mutation and self-adaption for numeric genetic algorithms”. In: *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. Vol. 1. 1995, pp. 384–. DOI: `10.1109/ICEC.1995.489178`.

[29] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. ISBN: 0262082136.

[30] John H. Holland. “Genetic Algorithms”. In: *Scientific American* 267.1 (1992), pp. 66–73. ISSN: 00368733, 19467087.

[31] Noriszura Ismail and Abdul Aziz Jemain. “Handling Overdispersion with Negative Binomial and Generalized Poisson Regression Models”. In: 2007.

[32] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. “Accurately Interpreting Clickthrough Data As Implicit Feedback”. In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’05. Salvador, Brazil: ACM, 2005, pp. 154–161. ISBN: 1-59593-034-5. DOI: `10.1145/1076034.1076063`. URL: `http://doi.acm.org/10.1145/1076034.1076063`.

[33] John K Kruschke. “Bayesian estimation supersedes the t-test”. In: *Journal of Experimental Psychology: General* 142.2 (2013), pp. 573–603.

[34] Jiajuan Liang. “Testing The Mean For Business Data: Should One Use The Z-Test, T-Test, F-Test, The Chi-Square Test, Or The P-Value Method?” In: *Journal of College Teaching & Learning (TLC)* (2011).

[35] Yiqun Liu, Chao Wang, Min Zhang, and Shaoping Ma. “User behavior modeling for better Web search ranking”. In: *Frontiers of Computer Science* 11.6 (2017), pp. 923–936. ISSN: 2095-2236. DOI: `10.1007/s11704-017-6518-6`. URL: `https://doi.org/10.1007/s11704-017-6518-6`.

[36] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[37] Jenna Carr May. “An Introduction to Genetic Algorithms”. In: 2014.

[38] Sadegh Mirshekarian and Gürsel A. Süer. “Experimental study of seeding in genetic algorithms with non-binary genetic representation”. In: *Journal of Intelligent Manufacturing* 29.7 (2018), pp. 1637–1646. ISSN: 1572-8145. DOI: `10.1007/s10845-016-1204-3`. URL: `https://doi.org/10.1007/s10845-016-1204-3`.

[39] Fedric Fernando Mohammad Isa Irawan Imam Mukhlash. “Optimization of PID Controller Using Genetic Algorithm for Missile’s Automatic Steering System”. In: *Journal of Theoretical and Applied Information Technology* 95 (2017). ISSN: 1817-3195.

[40] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore. “Automating biomedical data science through tree-based pipeline optimization”. In: *CoRR* abs/1601.07925 (2016). arXiv: `1601.07925`. URL: `http://arxiv.org/abs/1601.07925`.

[41] Andrew Bruce Peter C Bruce. *Practical statistics for data scientists: 50 essential concepts*. O’Reilly, 2017.

[42] Vinaitheerthan Renganathan. “Overview of Frequentist and Bayesian Approach to Survival Analysis”. In: 2016.

[43] Mohamed Dahab Sahar Alwadei and Mahmod Kamel. “A Feature Selection Model based on High-Performance Computing (HPC) Techniques”. In: *International Journal of Computer Applications* (2017).

[44] Howard J. Seltman. *Experimental Design and Analysis.* 2010. URL: `http://www.stat.cmu.edu/%5C~%7B%7Dhseltman/309/Book/`.

[45] Dan Siroker and Pete Koomen. *A/B Testing, The Most Powerful Way to Turn Clicks Into Customers.* John Wiley & Sons Inc, 2013. ISBN: 9781118792414.

[46] Anju Thapa. *Beginner’s Performance with MessagEase and QWERTY.* 2013.

[47] Werner Van Geit, Michael Gevaert, Giuseppe Chindemi, Christian Rössert, Jean-Denis Courcol, Eilif B. Muller, Felix Schürmann, Idan Segev, and Henry Markram. “BluePyOpt: Leveraging Open Source Software and Cloud Infrastructure to Optimise Model Parameters in Neuroscience”. In: *Frontiers in Neuroinformatics* 10 (2016), p. 17. ISSN: 1662-5196. DOI: `10.3389/fninf.2016.00017`. URL: `https://www.frontiersin.org/article/10.3389/fninf.2016.00017`.

[48] Sanford Weisberg. *Applied Linear Regression.* John Wiley & Sons, Incorporated, Somerset, 2013.

[49] Genlin Ji Yudong Zhang Shuihua Wang. “A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications”. In: *Mathematical Problems in Engineering* (2015).

A Test of Different Configurations for Genetic Algorithm

To evaluate what configuration was best suitable for the specific problem, several configurations was tested on the training data. The parameters that were tested was:

- Selection method (SM)
- The mutation probability (MUPB)
- The standard devation of the gaussian distribution, used by the mutation operator (σ)

Table A shows how the different configurations varies the parameters to evaluate which configuration is best suited. All other parameters and data was the same for all configurations.

Configuration	**Selection Method**	**MPB**	σ
1	Tournament	0.01	200
2	Tournament	0.01	400
3	Tournament	0.1	200
4	Tournament	0.1	400
5	Roulette Wheel	0.01	200
6	Roulette Wheel	0.01	400
7	Roulette Wheel	0.1	200
8	Roulette Wheel	0.1	400

Table A.1: Summary of the different configurations tested in the genetic algorithm

A.1 Configuration 1

Configuration 1 uses a tournament selection, a mutation probability of 0.01 and standard deviation of 200. The results from the execution can be seen in Figure A.1. The fitness score for the best chromosome in the last generation was 0.46223.

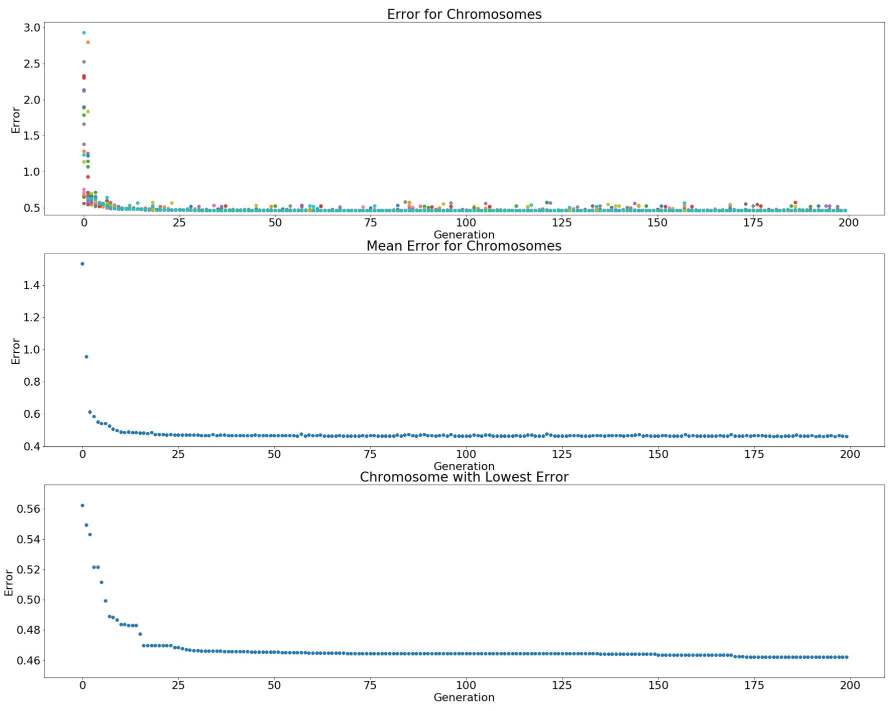


Figure A.1: Results from Configuration 1

A.2 Configuration 2

Configuration 2 uses a tournament selection, a mutation probability of 0.01 and standard deviation of 400. The results from the execution can be seen in Figure A.2. The fitness score for the best chromosome in the last generation was 0.438544.

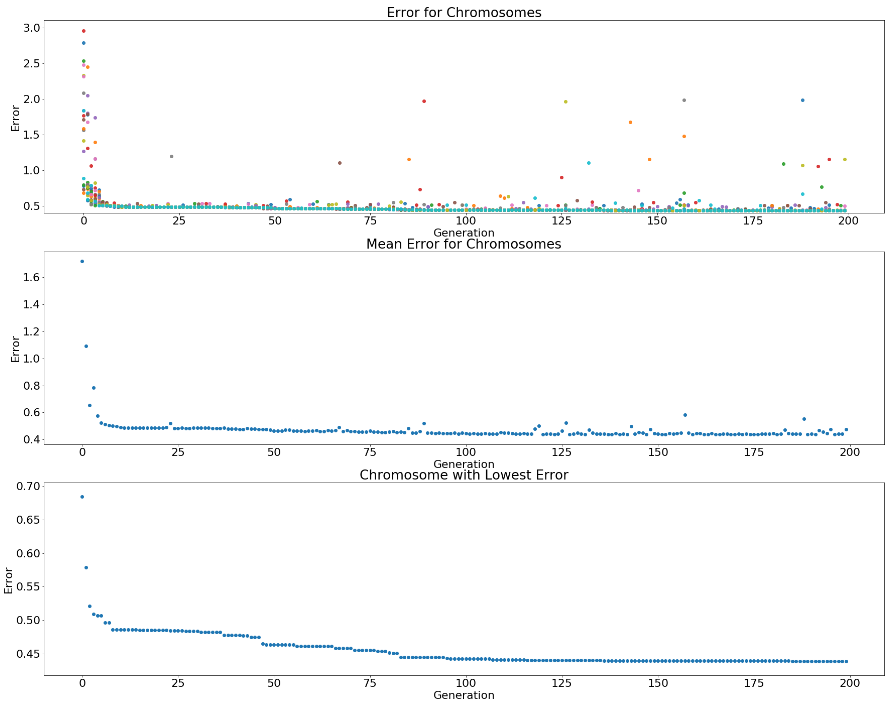


Figure A.2: Results from Configuration 2

A.3 Configuration 3

Configuration 3 uses a tournament selection, a mutation probability of 0.1 and standard deviation of 200. The results from the execution can be seen in Figure A.3. The fitness score for the best chromosome in the last generation was 0.454238.

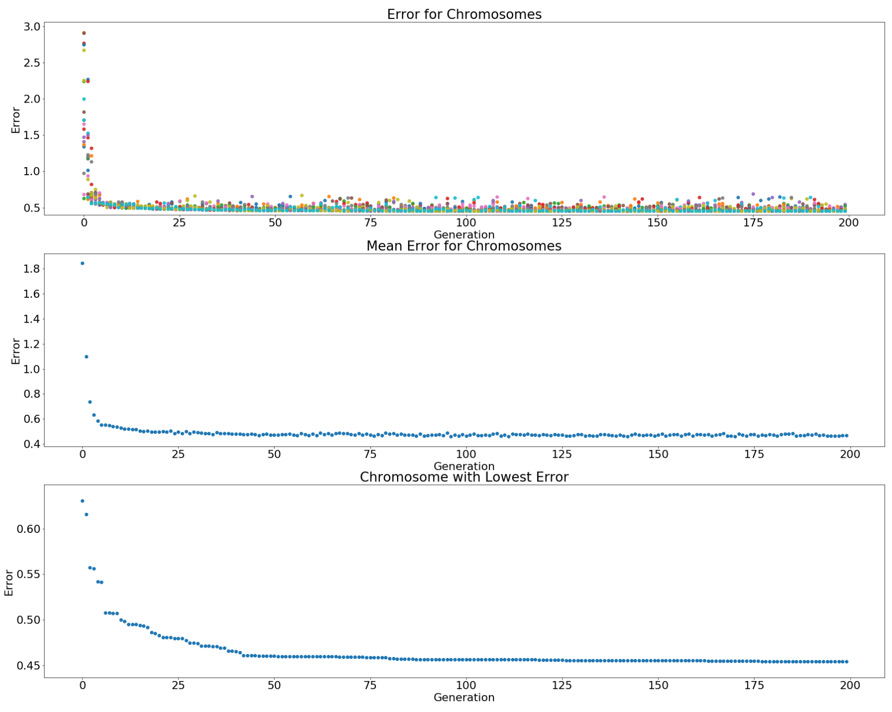


Figure A.3: Results from Configuration 3

A.4 Configuration 4

Configuration 4 uses a tournament selection, a mutation probability of 0.1 and standard deviation of 400. The results from the execution can be seen in Figure A.4. The fitness score for the best chromosome in the last generation was 0.442606.

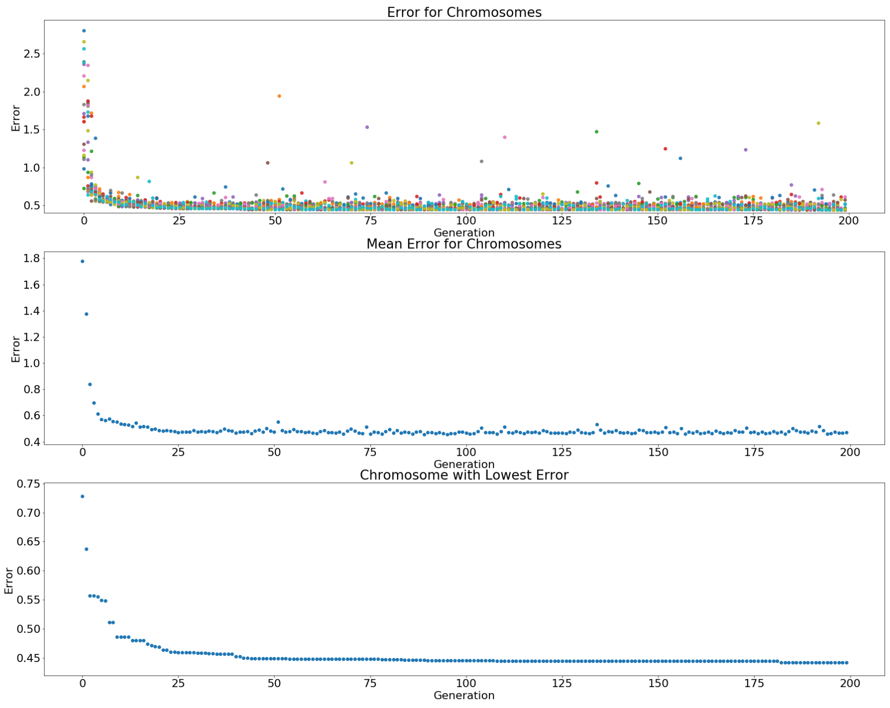


Figure A.4: Results from Configuration 4

A.5 Configuration 5

Configuration 5 uses a roulette wheel selection, a mutation probability of 0.01 and standard deviation of 200. The results from the execution can be seen in Figure A.5. The fitness score for the best chromosome in the last generation was 0.472242.

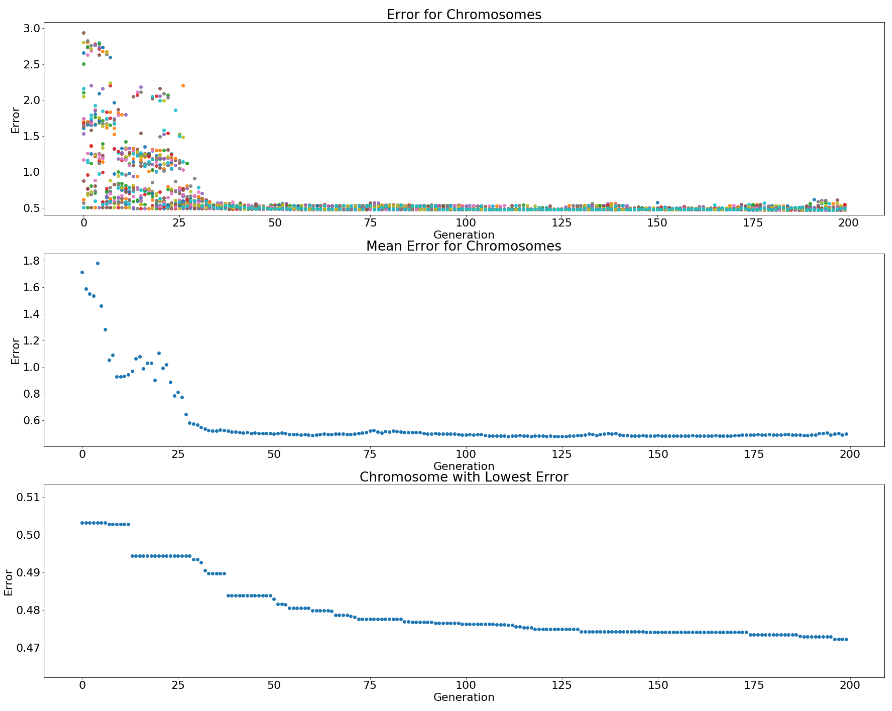


Figure A.5: Results from Configuration 5

A.6 Configuration 6

Configuration 6 uses a roulette wheel selection, a mutation probability of 0.01 and standard deviation of 400. The results from the execution can be seen in Figure A.6. The fitness score for the best chromosome in the last generation was 0.454232.

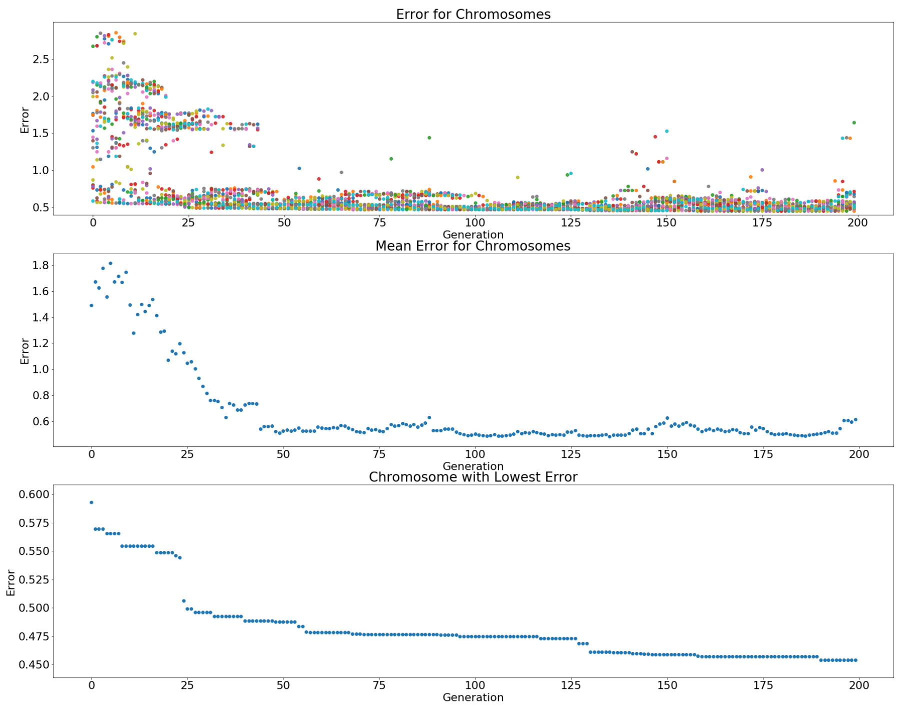


Figure A.6: Results from Configuration 6

A.7 Configuration 7

Configuration 7 uses a roulette wheel selection, a mutation probability of 0.1 and standard deviation of 200. The results from the execution can be seen in Figure A.7. The fitness score for the best chromosome in the last generation was 0.452254.

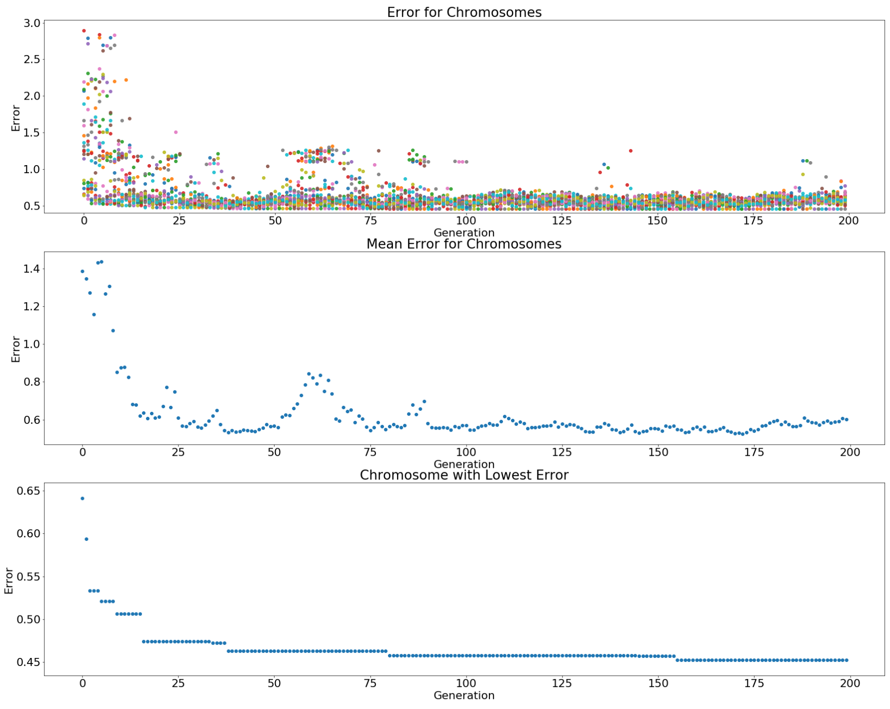


Figure A.7: Results from Configuration 7

A.8 Configuration 8

Configuration 8 uses a roulette wheel selection, a mutation probability of 0.1 and standard deviation of 400. The results from the execution can be seen in Figure A.8. The fitness score for the best chromosome in the last generation was 0.475674.

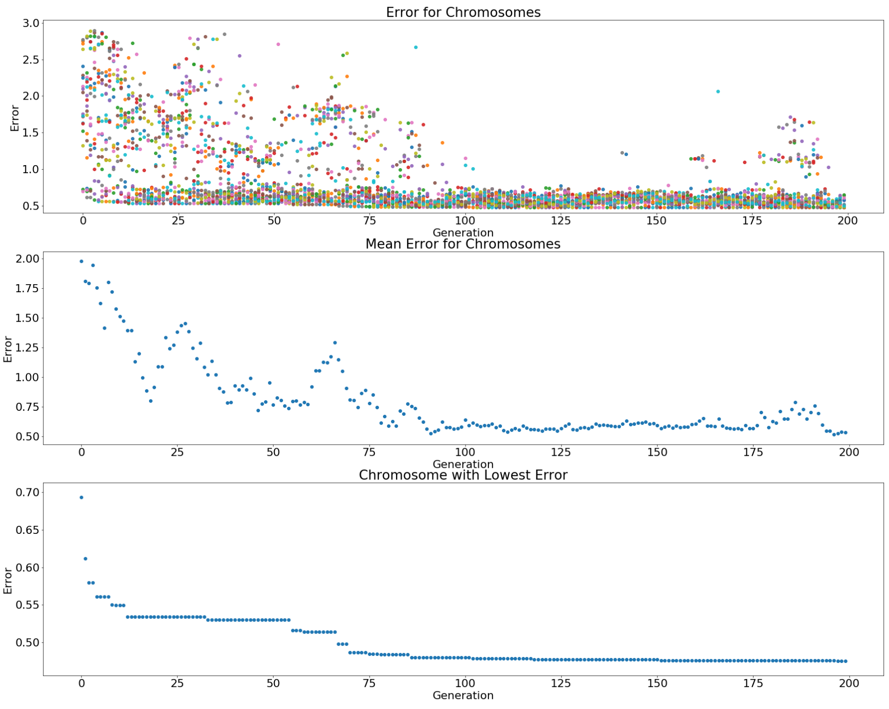


Figure A.8: Results from Configuration 8