Evaluating Personalised Website Ranking Using Small Scale User Feedback: A User Study

Bachelor-Thesis by Khalil Rahmouni from Tunisia
Date of Submission:

1. Referee: Prof. Dr. Johannes Fürnkranz
2. Referee: Dr. Eneldo Loza Mencía

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Department of Computer Science
Knowledge Engineering Group
Fachbereich Informatik

Evaluating Personalised Website Ranking Using Small Scale User Feedback: A User Study

Submitted Bachelor-Thesis by Khalil Rahmouni from Tunisia

1. Referee: Prof. Dr. Johannes Fürnkranz
2. Referee: Dr. Eneldo Loza Mencía

Date of Submission:

Thesis Statement pursuant to §22 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form. In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, 30.10.2018

(Khalil Rahmouni)

Abstract

This thesis evaluates the ranking quality of a web browser extension search engine that uses explicit relevance feedback to learn a personalized model. A user study is conducted to collect a small scale data that will be used in the evaluation process and the comparison with classification SVM and SVM Rank. We conclude that the learned personalized model enhances the ranking performance and outperforms the original rank, classification SVM and SVM Rang in a small-scale data.

Contents

List of Tables

List of Algorithms

List of Figures

1 Introduction

With the enormous growth of website number, returning good ranked results to a user search query is becoming a harder task. To deal with this problem, most of the commercial search engines record certain user activities such as queries and clicks, which consequently results in the rise of privacy issues. In the last years, a new type of search engines referred to as private search engine appeared with the goal of enhancing web searching privacy. On the other side renouncing to store user behavior has affected the ranking quality of the search results. In this context, the Knowledge Engineering Group of TU Darmstadt developed a web browser extension search engine named Poodle that store user behavior locally in the browser and uses them to learn a model that delivers a personalized results rank. Essentially the learned model is based on explicit relevance feedback provided by the user, the use of such feedback has been relaxed over the years because of their high cost and the fact that there is no guarantee to obtain them as most users are unwilling to provide such information especially on the web. This fact could be neglected assuming that users accept to be cooperative when using Poodle as a search engine. Adopting such approach motivated us to examine the added benefit for the performance of Poodle ranking using data collected through a user study.
The rest of this thesis is organized as follows. In chapter 2, machine learning methods used in this thesis will be introduced, including text preprocessing techniques, classification algorithms, and evaluation metrics. In chapter 3 we discuss related work. In chapter 4 we describe the functionality and the different features of Poodle. In chapter 5 the user study and the collected data are described. In chapter 6 we present the results and evaluate Poodle ranking. In chapter 7 we conclude this work.

2 Preliminaries

2.1 Machine Learning

Machine learning is a field of artificial intelligence that focuses on acquiring knowledge from available data to develop an algorithm, which can be used to make a prediction on new and unseen data. Machine learning is applied in many computing tasks such as natural language processing, medical diagnosis, email filtering, computer vision, information retrieval, and others. Within information retrieval, we are interested in the text classification problem because it is the key idea behind Poodle functionality. The text classification problem is formulated as follows:
Given a set of text documents D and a set of classes C, for each pair $[d_j, c_p]$ such that $d_j \in D$ and $c_p \in C$, a binary function F assign 1 to it if it predicts that the document d_j is a member of class c_p or 0 if it predict that the document d_j is not a member of class c_p

$$F : D \times C \rightarrow \{0, 1\} \tag{2.1}$$

2.2 Text Preprocessing

Text classification is a complex problem, that requires high dimensional. This is why the original text document has to be simplified through text preprocessing techniques. In the rest of this section, the basic steps of text preprocessing are presented.

2.2.1 Tokenization

Tokenization is the process of converting a stream of characters into a stream of words. Thus, the goal of tokenizer is the identification of the words in the text. The common way to do it is to first split the text into phrases using a vertical bar, question mark, and full stop. Then the phrases are split into words using space and comma, usually, punctuation is discarded after the tokenization.
At first sight, this strategy seems to be sufficient, however, some particular cases must be treated carefully. For example punctuation within words, digits, the apostrophe for possession and hyphens. In such situations, the output word could have many forms, this is why we must use the same tokenization strategy for both document and query. In addition to these particular cases, there are some specific words that we wish to recognize as terms, such as web URLs, IP addresses and email addresses. For these cases, a list of exceptions is needed.

2.2.2 Stop Word Removal

Typically, words that occur frequently in most documents of the corpus are not a good differentiator, which make them useless for purposes of retrieval. Such words are called stopwords and are filtered out of the index terms. A basic example of stopword are conjunctions, prepositions, and articles. Furthermore, stopwords removal contribute to a decrease in the corpus's size which improves the performance of the retrieval system. This is why stopwords list has been expanded to include some verbs, adverbs, and adjectives, for example, the SMART Retrieval-System[10] stopword list for the English language contains 571 words. Nevertheless, stopword removal might reduce recall, for example, the query 'to be or not to be' might contain only the term 'be' after the stopword removal which makes it impossible to find any relevant result to the query, that is why some web search engines use a full-text index.

2.2.3 Stemming

Almost in every text document, occurs the same word several time but in a different morphological form. This might reduce the performance of the retrieval system, for instance, if a user specifies a word in a query which occurs in another variant in the relevant document, there is a possibility that the relevant document will not be returned to the user. This problem can be avoided by detecting these variations and mapping them to their common root or stem. This process is referred to as stemming. For example, the word connect is the stem of the words: connected, connecting, connection and connections. Stemming has the advantage of reducing the corpus size as it substitutes distinct terms by the same word, thus making information retrieval a faster process.
We can distinguish four types of stemming algorithms: affix (i.e., prefixes and suffixes) removal, table lookup, successor variety, and n-grams. Among these different strategies, the affix removal is known for its simplicity and efficiency. There are many affix removal algorithms, the most popular one For English is the Porters Stemmer algorithm [1][2]. The Porters Stemmer algorithm uses a suffix list to define rules which are applied to remove the suffix, these rules are grouped into 5 different phases to ensure efficiency, for example, the rule:

$$s \rightarrow \phi \tag{2.2}$$

convert plural words to their respective singular form by removing the letter 's' at the end of the word. Notice that the rule that will be applied is the one which its left-hand side matches with the longest suffix sequence of letters of the word, for example given these two rules and the word stresses:

$$\begin{aligned} sses &\rightarrow ss \\ s &\rightarrow \phi \end{aligned} \tag{2.3}$$

The first rule is applied and we get the right stem stress instead of 'stresse'.
In 2006 Porter designed a detailed framework of stemming, called Snowball[16]. The framework allows programmers to develop their own stemmers for other character sets or languages.

2.3 Vector-Space Model

Every text classification problem requires another representation form of the documents that are suitable for the learning algorithm and the classification task and other than a sequence of strings and characters. The conventional solution to this problem is the use of the vector space model. Vector space model is the representation of a set of documents as a vector in a standard vector space. Let D be the set of all available documents.

$$D = \{d_1, \ldots, d_m\} \tag{2.4}$$

We define the vocabulary T as the set of the terms occurring in all the documents.

$$T = \{t_1, \ldots, t_n\} \tag{2.5}$$

For each document $d_i \in D$ we define for every term $t_k \in T$ a weight $w_{i,k} \in R$ that reflect the importance of the term in the document. All the weights of the document d_i form the vector w_i which is a representation of the document d_i in the vector space model.

$$w_i = (w_{i,1}, \ldots, w_{i,n}) \in R^n \tag{2.6}$$

2.3.1 Term Weighting

Text documents are composed of many terms. Using these terms to describe the document is not that easy because words within text are not equally important some words are more ambiguous than others. To identify the importance of a term for the text description, we assign a weight w to each term that characterizes its importance. The computation of the term weight depends on the used model. In the Boolean model, we assign 1 to terms that occur in the document and 0 to terms that do not occur. Other models use the frequency of the terms to compute the weights, for example, the bag-of-words model. We can calculate the weights Using the frequency of words differently, depending on where we interpret its influence:

- Locally: the more a term occurs in a document, the more it is relevant. For this case we define the term frequency weight TF as the fraction of the count of the term t in the document d $f_{t,d}$ and the total length of document d.

$$TF(t,d) = \frac{f_{t,d}}{\sum_{i \in d} f_{i,d}} \tag{2.7}$$

- Globally: the more a term occurs in different documents of the corpus, the less it is relevant. For this case, we define the inverse document frequency IDF as the logarithm of the fraction of the total number of the documents in the corpus N and the number of the document where the term t occurs.

$$IDF(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.8}$$

It is common to combine TF and IDF and use the TFIDF approach, which is the multiplication of TF and IDF as it gives high weights to terms that appear frequently in a small number of documents in the document set.

$$TFIDF(t,d,D) = TF(t,d) * IDF(t,D) \tag{2.9}$$

2.3.2 Cosine Similarity

Computing similarity between vectors in the vector space is a fundamental task for many text mining application. For example, a web retrieval system has to return a relevant result to the user's query, which can be achieved by measuring the similarity between the query vector and the documents vectors. In recent years, many similarity metrics have been proposed such as Cosine, Jaccard, Dice, Overlap[17]. Among all these metrics the cosine similarity measure is the most widely used one. Instead of using the distance between two vectors to determine their similarity, the cosine similarity uses the cosine of the angle between the two vectors.

Given two vectors d_1 and d_2 the cosine similarity between them is calculated as follows:

$$\cos(\theta) = \frac{d_1 \cdot d_2}{\|d_1\| \cdot \|d_2\|} = \frac{\sum_{i=1}^{n} d_{1,i} d_{2,i}}{\sqrt{\sum_{i=1}^{n} d_{1,i}^2} \cdot \sqrt{\sum_{i=1}^{n} d_{2,i}^2}} \tag{2.10}$$

2.4 Classification Algorithm

A Classification algorithm is a program responsible for learning a model that can be used to classify new data. Based on the used learning mechanism, Classification algorithms can be split into two groups:

- Unsupervised learning, where we are not trying to predict specific features and instead would like to group similar instances together for example clustering.
- Supervised learning, where a set of training data must be provided to the algorithm as input.

Furthermore, classification algorithms that require training data in the learning phase can be also split into two categories based on the training data availability:

- Batch learning: This type of learning requires the availability of the entire training data before starting the learning phase. Therefore training data has to be collected and summarized in an extra step. The Batch algorithm aims for optimizing a cost function that is defined on the training data set, bypassing much time through the training data which require good memory and hardware resources.
- Incremental learning: In contrast to batch algorithms, the training data are not all at once provided to the learning algorithm as input. But rather, the training data is provided as a continuous stream. Each training instance is treated once, there is no need for storage and reprocessing. The most recent model that reflects all the training instance seen so far is maintained until a training instance that violate the model appear then it is immediately adjusted.

In the rest of this section, we present four learning algorithms that are widely used in the text classification problem.

2.4.1 Naive Bayes

The Naive Bayes classifier is a probabilistic classifier based on the bayes theorem. To classify a test document d, the probability $P(c \mid d)$ that a document d belongs to a class c is computed for every class $c_i \in C$. Once these probabilities have been computed for all classes, the document d is assigned to the class with the highest probability.

$$c = \arg\max_{c \in C} P(c \mid d) \tag{2.11}$$

To calculate the probability $P(c \mid d)$ we use the bayes theorem and we obtain the following:

$$c = \arg\max_{c \in C} = \frac{P(c) * P(d \mid c)}{P(d)} \tag{2.12}$$

The denominator $P(d)$ in equation 10 can be omitted because it is the same for all classes and does not affect the argmax. It remains to estimate $P(c)$ and $P(d \mid c)$, for this purpose we use the maximum likelihood, which estimate the class prior probability $P(c)$ as follows:

$$P(c) = \frac{N_c}{N} \tag{2.13}$$

where N_c is the number of documents in class c and N is the total number of documents. To estimate the conditional probability $P(d \mid c)$ some simplification have to be done first, the most common one, is the application of the positional independence assumption, which assume independence among the index

terms that compose the documents. This assumption does not hold with real documents, this is why classifiers based on it are called Naive Bayes classifiers. After the simplification we obtain:

$$P(d \mid c) = P(t_1, t_2 \dots t_n \mid c) = \prod_{i=1}^{|d|} P(t_i \mid c) \tag{2.14}$$

where t are the terms od document d. Now we can estimate $P(t \mid c)$ as the relative frequency of term t in documents belonging to class c as follows:

$$P(t \mid c) = \frac{T_{ct}}{\sum_{t \in V} T_{ct}} \tag{2.15}$$

where $T_c t$ is the number of occurrences of t in training documents from class c and V is the corpus vocabulary. The problem with this estimation is that it assign 0 to the estimation of terms that did not occur in the training data and by result all the classes probabilities will be 0. To avoid this problem the Laplace smoothing is used, which adds 1 to each count.

$$P(t \mid c) = \frac{T_{ct} + 1}{\sum_{t \in V}(T_{ct} + 1)} = \frac{T_{ct} + 1}{(\sum_{t \in V} T_{ct}) + |V|} \tag{2.16}$$

Putting all together we get:

$$c = \arg\max_{c \in C} P(c) \prod_{i=1}^{|d|} P(t_i \mid c) \tag{2.17}$$

where $P(c)$ is claculated as in equation (13) and $P(t_i \mid c)$ as in equation (16).Usually we add logarithms of probabilities instead of multiplying probabilities to avoid ending in a floating point underflow:

$$c = \arg\max_{c \in C} \log(P(c)) + \sum_{i=1}^{|d|} \log(P(t_i \mid c)) \tag{2.18}$$

The training and testing Nive Bayes algorithm pseudo code[6] is represented in Algorithm 1 and Algorithm 2.

2.4.2 Rochio

The Rocchio algorithm [5] was initially invented to improve the performance of the retrieval system using explicit relevance feedback. Under the assumption that relevant documents are similar and non-relevant documents are dissimilar from the relevant documents, the initial query is reformulated such that it gets closer to the neighborhood of the relevant documents and away from the neighborhood of the non-relevant documents.

The standard Rocchio method to compute the modified query q_{i+1} is given as

$$q_{i+1} = \alpha \cdot q_i + \beta \cdot \sum_j r_j - \gamma \cdot \sum_j i_j \tag{2.19}$$

Algorithm 1 TrainNaiveBayes(C,D)

$V \leftarrow ExtractVocabulary(D)$
$N \leftarrow CountDocs(D)$
for each $c \in C$ do
    $N_c \leftarrow CountDocsInClass(D, c)$
    $prior[c] \leftarrow N_C \setminus N$
    $text_c \leftarrow ConcatenateTextOfAllDocsInClass(D, c)$
end for
for each $t \in V$ do
    $T_{ct} \leftarrow CountTokensOfTerm(text_c, t)$
end for
for each $t \in V$ do
    $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_t (T_{ct}+1)}$
end for
return V, prior, condprob

Algorithm 2 TestNaiveBayes(C,V,prior,condprob,d)

$W \leftarrow ExtractTokensFromDoc(V, d)$
for each $c \in C$ do
    $score[c] \leftarrow \log prior[c]$
end for
for each $t \in W$ do
    $score[c] += \log condprob[t][c]$
end for
return $\arg\max_{c \in C} score[c]$

where $r_j \in R$ and R is the set of relevant documents, $i_j \in I$ and I is the set of non-relevant documents and α, β, γ are tunning constants. Typically α is fixed to 1 and γ shoould be smaller than β because relevant documents contain more beneficial information than the non-relevant documents.

The Rocchio relevance feedback has been adjusted to deal with text classification problem. We just have to interpret the training set as relevance feedback, we consider terms that belong to training document of a given class c_p as positive feedback, and terms that belong to training documents outside the class c_p as negative feedback. For every class $c \in C$ a prototype vector $\vec{\mu}$ called centroid is calculated as the vector average of its members:

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} d \tag{2.20}$$

, where D_c is the set of documents with class c. The resulting set of centroid vectors represents the learned model. The pseudo-code of the Rocchio algorithm is shown below in Algorithm 3. To classify a new document d', it is first represented as a vector in the vector space, then the cosine of d' and every class centroid vector is calculated, d' will be assigned to the class with which its vector has the highest cosine.

$$\text{Assign d' to class c} = argmax_{c'} \cos(\vec{c'}, \vec{d'}) \tag{2.21}$$

As shown in Figure 2.1, we have two classes positive and negative, for each class the centroid vector is computed, to classify the document represented with the blue vector the cosine of the angle between

Algorithm 3 TrainRocchio(C,D)

1: for each $c_j \in C$ do
2: $D_j \leftarrow \{d : \langle d, c_j \rangle \in D\}$
3: $\overrightarrow{\mu_j} \leftarrow \frac{1}{|D_c|} \sum_{d \in D_j} d$
4: end for
5: return $\{\overrightarrow{\mu_1}, \ldots, \overrightarrow{\mu_j}\}$

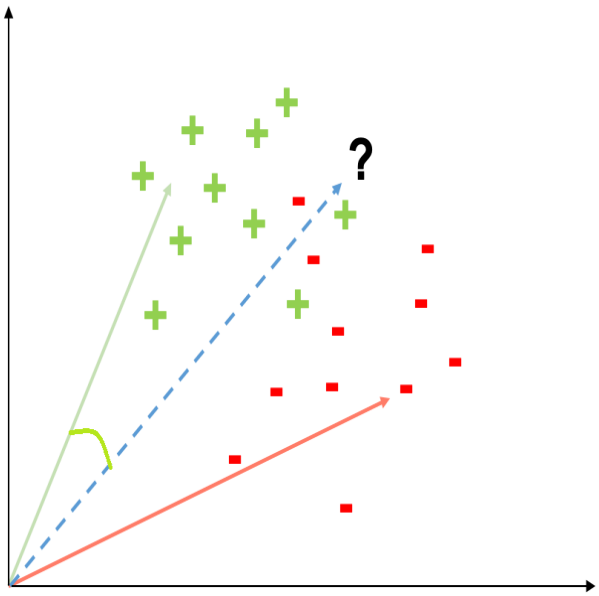

Figure 2.1: Rocchio Classification

the blue vector and each centroid vector is calculated. The document is then assigned to the class with which the document has the biggest cosine value.

2.4.3 Perceptron

The Perceptron algorithm is a binary classifier that belongs to the gradient descent algorithm class. Gradient descent algorithms are iterative learning algorithms which aim to optimize a function of the data that computes a goodness criterion, in each step the derivative of the function is calculated and the parameters of the model are updated in the direction of the steepest gradient. This will enhance the performance of the model because the direction of the steepest gradient improves the goodness criterion better than any other direction. The pseudo code of the Perceptron learning algorithm[7] is given in Algorithm 4, we consider the problem of classifying a text document to the class yes or no with $\overrightarrow{d}$ is the vector representation of the text document:

Algorithm 4 The Perceptron Learning Algorithm

```
function decision(d⃗, W⃗, θ)
    if W⃗ · d⃗ > θ then
        return yes
    else
        return no
    end if
end function
W⃗ = 0
θ = 0
while not converged yet do
    for all elements d⃗_j in the training set do
        d = decision(d⃗_j, W⃗, θ)
        if class(d⃗_j) = d then
            continue
            else
            if class(d⃗_j) = yes and d = no then
                θ = θ − 1
                W⃗ = W⃗ + d⃗_j
            end if
            else
            if class(d⃗_j) = no and d = yes then
                θ = θ + 1
                W⃗ = W⃗ − d⃗_j
            end if
        end if
    end for
end while
```

The goal of the Perceptron algorithm is to learn a weight vector $\vec{w}$ and a threshold θ such that the dot product of the weight vector and the representative vector of a document $\vec{d}$ compared with the threshold θ provides the classification decision.

$$\begin{aligned} &\text{Predict yes iff } \vec{w} \cdot \vec{d} > \theta \\ &\text{Otherwise no} \end{aligned} \tag{2.22}$$

At the beginning of the learning state, The weight vector $\vec{w}$ and the threshold θ are initialized to zero, then for every instance of the training set, make perceptron a prediction. If the prediction is false we update the model by moving the weight vector in the direction of the greatest change. For the case, where the class is yes and it is predicted no, we add the document vector to the weight vector otherwise we subtract the document vector from the weight vector.

Figure 2.2 illustrates the miss-classification correcting process of the perceptron algorithm, the vector $\vec{x}$ is first assigned to the 'NO' class as it lies on the 'no' side of the decision boundary S. The correction step adds $\vec{x}$ to $\vec{w}$, now lies $\vec{x}$ on the 'yes' side of S' the decision boundary of the new weight vector $\overrightarrow{w+x}$

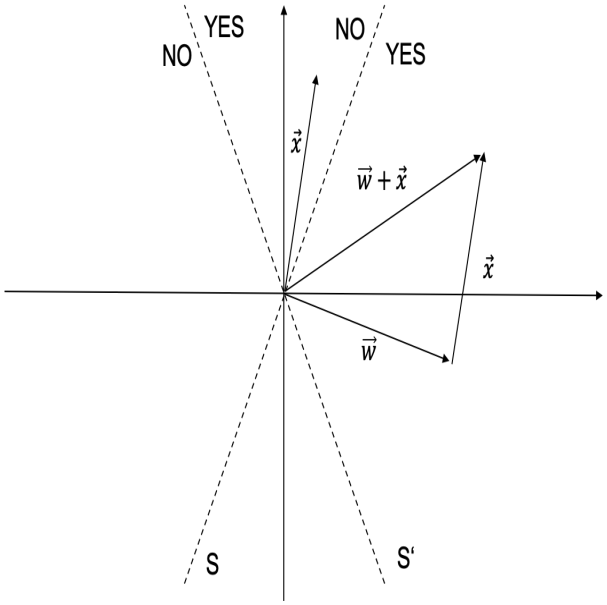

Figure 2.2: Perceptron Learning

2.4.4 Support Vector Machine

Support Vector Machine is a learning method founded by Vapnik[18], the first application of SVM was as a binary classifier where the algorithm tries to find a decision boundary that correctly classifies the training data, a multiclass classification was later implemented by combining multiple binary classifiers. For linear separable problems, there is usually many possible separators, as Figure 2.3 shows the line L1, L2 and L3 classify correctly the two different data. The strategy for choosing the separator boundary differ from one learning algorithm to another, for instance, the Perceptron algorithm finds just any linear separator, unlike Naive Bayes which search for the best linear separator according to some measure. The SVM algorithm defines the decision surface to be maximally far away from any data point, the distance between the decision surface and the closest data points is called 'margin' and the data located on the border of the margin are referred to as the support vectors, all these SVM characteristic are illustrated in Figure 2.4.

By maximizing the margin avoid SVM uncertain classification decision as the data near the decision surface could be almost classified in either way.

2.4.5 SVM Rank

Support Vector Machine has been extended for many application such as regression and ranking. The SVM Rank distinguish from the classification SVM by 2 factors, the training data presentation, and the output. Instead of using a set of data objects and their class labels as training data, the ranking SVM training data are formed as an ordered set which is denoted as:

$$R = \{(X_1, Y_i), \ldots, (X_m, Y_m)\} \tag{2.23}$$

where y_i is the ranking of X_i, such that $Y_i < Y_j$ if $X_i \succ X_j$. In the context of web search this could be extracted as follow: consider a user who clicked on the third displayed link, we can assume that he has scanned the links from top to bottom and then decided to click on the third one, so he preferred the

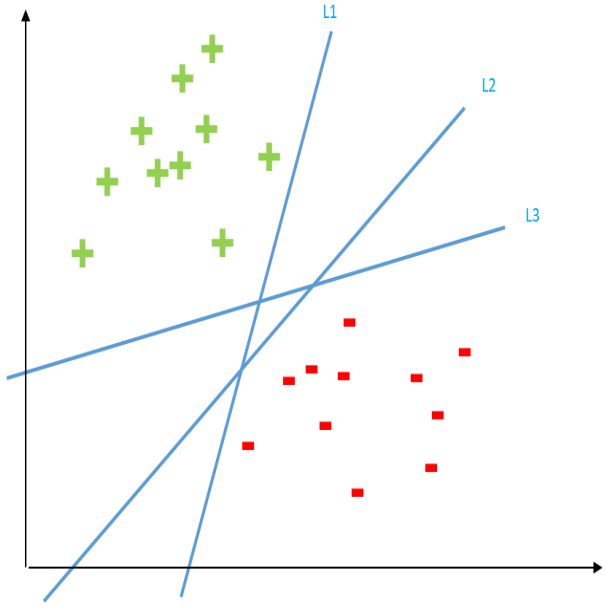

Figure 2.3: Linear classifiers in two-dimensional spaces

third link over the first and the second, this could be expressed as a partial relation $link_3 \succ X_2$ and $link_3 \succ X_1$ and used as training data. The SVM Rank goal is to learn a function F such as :

$$\forall\{(X_i, X_j) : Y_i < Y_j \in R\} : F(X_i) > F(X_j) \iff W \cdot X_i > w \cdot X_j \tag{2.24}$$

where w is a weight vector that is adjusted by learning so that a maximum number of the training data relation are satisfied if there exists a function F that satisfies all the relation we say that the ordering R is linearly rankable.

Figure2.5 [8] illustrates how two different weight vector W_1 and W_2 ranks four data points in a two dimensional example, the projection of the points onto the weight vector is used for the ranking, which implies that the order of the points for W_1 is: 1,2,3,4 while it is 2,3,1,4 for W_2. The margin δ is the distance between the closest two projections within all the points if two different weights vectors generate the same ranking the one with the maximum margin is selected.

The SVM Rank returns for each data a score computed by the learned function that determines the global ordering of the data.

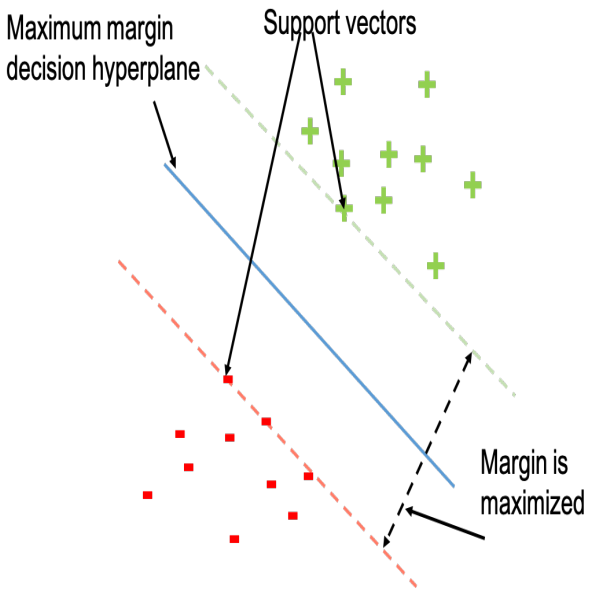

Figure 2.4: SVM support vectors and margin

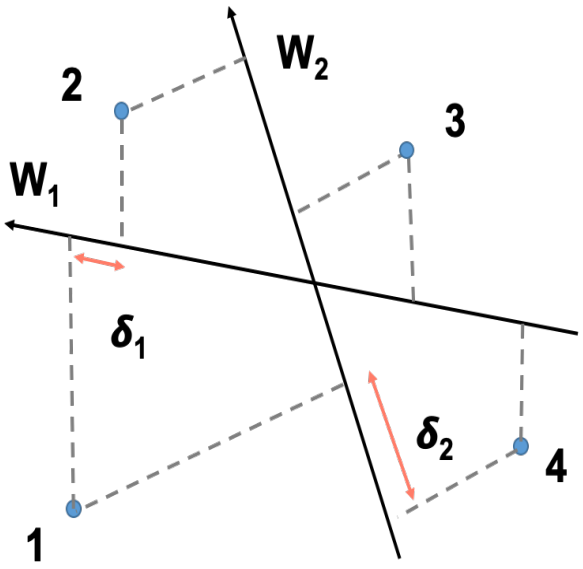

Figure 2.5: Ranking 4 points with two different weight vectors

2.5 Evaluation Metrics

Assessing the performance of a retrieval system is necessary to find out if the applied methods return good results or not and compare it with existing or new introduced systems. For this purpose some metrics has been defined, we present here 3 metrics that are widely used to evaluate the ranking quality of retrieval systems.

- Average Precision: generally precision is the percentage of relevant items in the returned set, in the context of web search precision is computed for every relevant document in the returned list, the average of all these precision values represent the average precision for the given query.

$$avgP = \frac{1}{|R_i|} \sum_{k=1}^{|R_i|} P(R_i[k]) \tag{2.25}$$

[6] In the best case where all the relevant documents are ranked ahead all the non relevant documents the average precision reach the max value 1. Consider the example illustrated in table 1 where ✓denote a relevant document and ✗denote a non-relevant document, we get the following precision values: 1/2 for d1, 2/3 for d2, 3/6 for d3, 4/7 for d5 and 5/8 for d4, the average of all these values is the average precision of this query which is 0.5726. The average precision can also be averaged over all the queries, this single values is referred to as the Mean Average precision which is also used to evaluate ranking systems.

- NDCG at K: Discounted Cumulative Gain (DCG) was invented by Järvelin and Kekäläinen[19] and is commonly used in web search applications. DCG has an advantage over the Average Precision that it accept not only binary relevance, but also multi levels relevance such as perfect, excellent, good, fair and bad. In addition DCG associate positions with weights called discounting factor, which increase proportionally with the position and aim to penalize relevant documents occurring in lower ranking position by dividing its relevance with the corresponding position discounting factor. DCG has the following general form:

$$DCG = \sum_{i=1}^{k} \frac{r_i}{\log(i+1)} \tag{2.26}$$

where k is the maximum rank considered and r_i is the relevance at position i. To be able to compare DCG values of different queries, we need to normalize them by dividing with the optimal DCG values(ODCG)

$$NDCG = \frac{DCG}{ODCG} \tag{2.27}$$

- Average Position: A simple method to evaluate a given ranking is to average the positions of the relevant documents:

$$AvgPosition = \frac{1}{N} \cdot \sum_{i=1}^{k} pos(r_i) \tag{2.28}$$

[20] where N is the number of the relevant document and $pos(r_i)$ is the position of the relevant document r_i. This could approximate the area where most of the relevant document are located.

Table 2.1: Example of 10 document ranking

Documents	d6	d1	d2	d10	d9	d3	d5	d4	d7	d8
Relevance	✗	✓	✓	✗	✗	✓	✓	✓	✗	✗

3 Related Work

In this section, we will discuss two relevant areas of related work: learning to rank via user feedback and personalized search.
Using user feedback to improve the quality of web search result rank have been well studied in online learning, we distinguish here between two type of feedback:

- explicit feedback, where the feedback information is provided directly by the users
- implicit feedback, where the feedback information is derived from the user interaction with the search results, for instance, the clicks generated or the time spent on a website.

in [10] a comparison between two web search systems, one uses explicit feedback and one uses implicit feedback, have been conducted by means of a user study. The results show that there were no significant differences and supported the hypothesis of the possibility to substitute explicit feedback with implicit feedback. In the work of [11], Joachims et al. a new approach to learning a ranking function based entirely on clickthrough data was introduced. The recent work in [12] Joachims et al. examined the reliability of implicit feedback generated from clickthrough data by performing eye tracking studies to analyze the user decision process and comparing the implicit feedback with relevance judgments provided by human assessors. The results show that it is not recommended to interpret clicks as a direct indication of relevance, a better approach is to interpret them as a user preference. Complementing explicit feedback with implicit feedback was proposed in the work of [13], Bell et al. The combination of both feedback arts is realized via a factorized neighborhood model and show consistent improvements over baseline methods
Personalized search is a promising path to enhance the performance of ranking quality, various techniques have been developed in this domain. In [14], White et al. user interactions are used to re-rank search results and predict the user search interests, which are represented as a list of Open Directory Project (ODP) categories. In[15], Sontag et al. a generative probabilistic model is introduced, which evaluates the search results relevance using user profile that is learned via long-term search history.

4 Poodle

Using Google search engines is preferred by most internet user because it delivers fast and relevant results. However, with these advantages comes privacy concerns, especially in the realm of search tracking, data storage, and use of personal information. For example, in order for Google to provide highly detailed and relevant results, they need to track all user's search history such as: what you search for, the ads you are interested in, What links you click, Which images you view, Which videos you watch.

This problem motivated the TU Darmstadt knowledge engineering group to develop a web search tool that returns personalized search result while ensuring privacy. The project was divided into two steps, first, a javascript machine learning framework named JSLearn was implemented then a web browser application named Poodle was developed based on the JSLearn framework. The framework supplies reusable, extendable parts as well as ready-to-use parts. Most importantly it contains several classifier and text processing algorithms such as stemming algorithms and tokenizing algorithms.

In the following, the functionality and features of Poodle will be explained in details.

4.1 Poodle functionality

To ensure privacy, the results shown by Poodle are taken from a discrete search engine named 'Startpage'[21]. Startpage is a popular discrete search engine that returns Google results without tracking user's IP address, using cookies to identify users and storing user's data.

Every time a user submits a search query on the Poodle side, send Poodle a request to 'Startpage' with the same query and post the results as soon as a response is available. The results that Poodle shows will only match with the Startpage results when Poodle has not yet learned a model, otherwise, it will depend on the latest learned model. To learn its model uses poodle relevance feedback provided by the user as training data, every link is fitted with two buttons: the like button and the dislike button. By means of these buttons, users have the possibility to assess any link. Poodle employs a binary classification in most of its features with the classes like and dislike, where the liked links are assigned to the like class and the disliked links are assigned to the dislike class. Every link is treated as a text document using its title and snippet. The snippet is the small text associated with each link, it provides a summary of the search result and helps the user decide which links are of interest. Figure 4.1 illustrates how a link is displayed to the user, we can recognize the two buttons on the right side, the title written in blue and the snippet in the black box. All the links are represented as a vector in the vector space, where the terms are deduced after the application of text preprocessing steps and associated with its term frequency TF. In such a scenario, it was not possible to use the TF-IDF weight although it improves the performance because the corpus cannot be defined in advance. Poodle has at its disposal three learning algorithm: Naive Bayes, Perceptron and Rocchio, all these algorithm work simultaneously but only the results of the selected one will be shown, the learning process is effected incrementally, each new incoming training instance update the last learned model, which results in a reclassification of all the links.

Moreover, Poodle has the ability to record all user interaction during the search as well as the different states of the ranking. This information could be extracted as a history file and used for further research. Figure 4.2 illustrates an example of Poodle history file, where the user searched for the query word 'Darmstadt' and rated one link, the first block corresponds to the results ranking before the rating and the second block corresponds to the ranking after the rating. Each block contains the complete URL list of the search results and the user settings. For every URL in the history file the user rating, the score of each classifier and the description text (snippet and title) are stored.

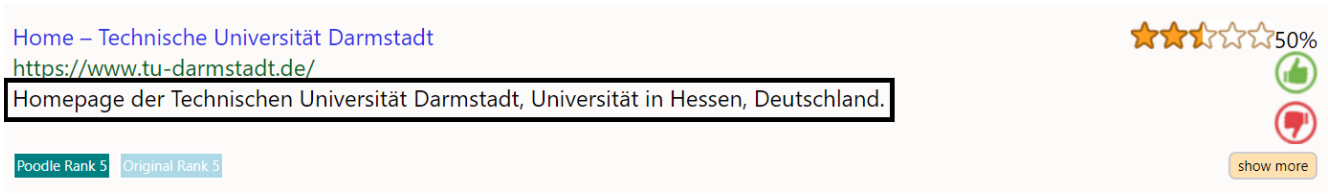

Figure 4.1: snippet


```
{
"statistik" : {"darmstadt":{"Tue Sep 11 2018 21:15:35 GMT+0200 (Mitteleuropäische Sommerzeit)":{"setting":"classifier: incrementalRocchio |
,"https://en.wikipedia.org/wiki/Darmstadt":{"positiv":0,"negativ":0,"position":2,"Displayed_position":2,"NaiveBayes":50,"Rocchio":50,"Percep
,"https://de.wikipedia.org/wiki/Darmstadt":{"positiv":0,"negativ":0,"position":3,"Displayed_position":3,"NaiveBayes":50,"Rocchio":50,"Percep
,"https://en.wikivoyage.org/wiki/Darmstadt":{"positiv":0,"negativ":0,"position":4,"Displayed_position":4,"NaiveBayes":50,"Rocchio":50,"Perce
,"https://www.jazzinstitut.de/?lang=en":{"positiv":0,"negativ":0,"position":5,"Displayed_position":5,"NaiveBayes":50,"Rocchio":50,"Perceptro
,"https://www.darmstadt-tourismus.de/":{"positiv":0,"negativ":0,"position":6,"Displayed_position":6,"NaiveBayes":50,"Rocchio":50,"Perceptron
,"https://www.tu-darmstadt.de/":{"positiv":0,"negativ":0,"position":7,"Displayed_position":7,"NaiveBayes":50,"Rocchio":50,"Perceptron":50,"B
,"https://www.sv98.de/":{"positiv":0,"negativ":0,"position":8,"Displayed_position":8,"NaiveBayes":50,"Rocchio":50,"Perceptron":50,"Beschreib
,"https://www.hlmd.de/":{"positiv":0,"negativ":0,"position":9,"Displayed_position":9,"NaiveBayes":50,"Rocchio":50,"Perceptron":50,"Beschreib
,"https://www.mathildenhoehe.eu/":{"positiv":0,"negativ":0,"position":10,"Displayed_position":10,"NaiveBayes":50,"Rocchio":50,"Perceptron":5
}
,"Tue Sep 11 2018 21:15:44 GMT+0200 (Mitteleuropäische Sommerzeit)":{"setting":"classifier: incrementalRocchio | numOfResults: 10 | Weightin
,"https://en.wikipedia.org/wiki/Darmstadt":{"positiv":0,"negativ":0,"position":2,"Displayed_position":5,"NaiveBayes":56.74433333333334,"Rocc
,"https://de.wikipedia.org/wiki/Darmstadt":{"positiv":0,"negativ":0,"position":3,"Displayed_position":2,"NaiveBayes":59.55100000000001,"Rocc
,"https://en.wikivoyage.org/wiki/Darmstadt":{"positiv":0,"negativ":0,"position":4,"Displayed_position":8,"NaiveBayes":56.474333333333334,"Ro
,"https://www.jazzinstitut.de/?lang=en":{"positiv":0,"negativ":0,"position":5,"Displayed_position":7,"NaiveBayes":56.55166666666667,"Rocchio
,"https://www.darmstadt-tourismus.de/":{"positiv":0,"negativ":0,"position":6,"Displayed_position":6,"NaiveBayes":56.74433333333334,"Rocchio"
,"https://www.tu-darmstadt.de/":{"positiv":1,"negativ":0,"position":7,"Displayed_position":1,"NaiveBayes":63.744,"Rocchio":62.06233333333333
,"https://www.sv98.de/":{"positiv":0,"negativ":0,"position":8,"Displayed_position":4,"NaiveBayes":57.19233333333333,"Rocchio":52.42466666666
,"https://www.hlmd.de/":{"positiv":0,"negativ":0,"position":9,"Displayed_position":9,"NaiveBayes":55.386,"Rocchio":50,"Perceptron":50,"Besch
,"https://www.mathildenhoehe.eu/":{"positiv":0,"negativ":0,"position":10,"Displayed_position":10,"NaiveBayes":55.386,"Rocchio":50,"Perceptro
}
}
}
,
"END" : 0
}
```


Figure 4.2: Poodle history file

4.2 Poodle Features

The latest version of Poodle offer 3 main features, which we present in more details in this section

4.2.1 Personalized Ranking

Unlike any other search engine, Offer Poodle the opportunity to the user to manipulate the ranking of the query search results by using the like and dislike buttons. Clicking on the like button of a link will push it to the top in contrast to the dislike button, which will push the link to the bottom. This will affect also the rank of the other links, by ameliorating the rank of the links that are similar to the liked link and lower the rank of the links that are similar to the disliked link. Every click will update the Poodle model and will somehow change the ranking. Figure 4.3 and Figure 4.4 illustrate an example of Poodle Personalized Ranking function, on the first Figure we can see that the all links have the same score at the start of the search, when mouseover on a link the rating buttons are displayed to the user, on the second Figure we can see that the third link on the first Figure jumped to the first place after getting a positive rate from the user, this could be recognized by the number in the middle of the button, other links position also changed as a consequence of the user feedback, this could be identified by the arrow on the right side of the link and by the two boxes on the left side of the link that indicates Poodle rank and the original rank, for example, the third link on the second Figure was placed seventh on the original rank.

4.2.2 Labeling

Finding the right result for a search query could be a hard task, due to the huge increase of the internet site number and ambiguity of the search query. This forces users to either reformulate the search query

or going through all the displayed results until finding the wished result. To reduce this problem effect, Poodle comes up with the labeling function. As figure x shows, the user can create some labels that are shown on the left side of the screen, then assign suitable links to each label. The links that are manually assigned to a label, have a boldly marked label. By clicking on a label only links that belong to this label will be displayed.

4.2.3 Context search

Another way to rank search results, is to re-rank them in context to a chosen link, in this case, all the feedback already stored are ignored and only the selected link is considered. To use this function, the user has to click on the show more button of a link. Context search can go down to two steps. The first step only considers the first search result selected and the second step considers the first and second search result selected as context.

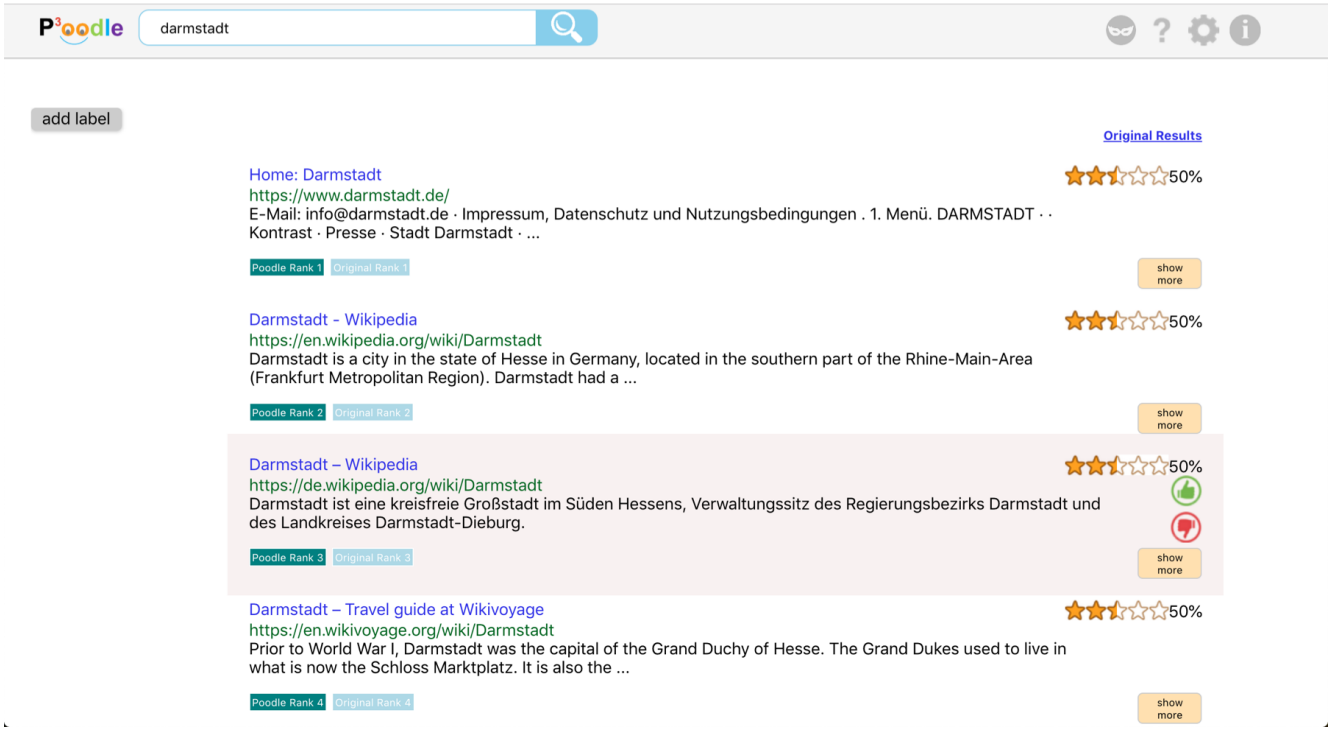

Figure 4.3: snippet

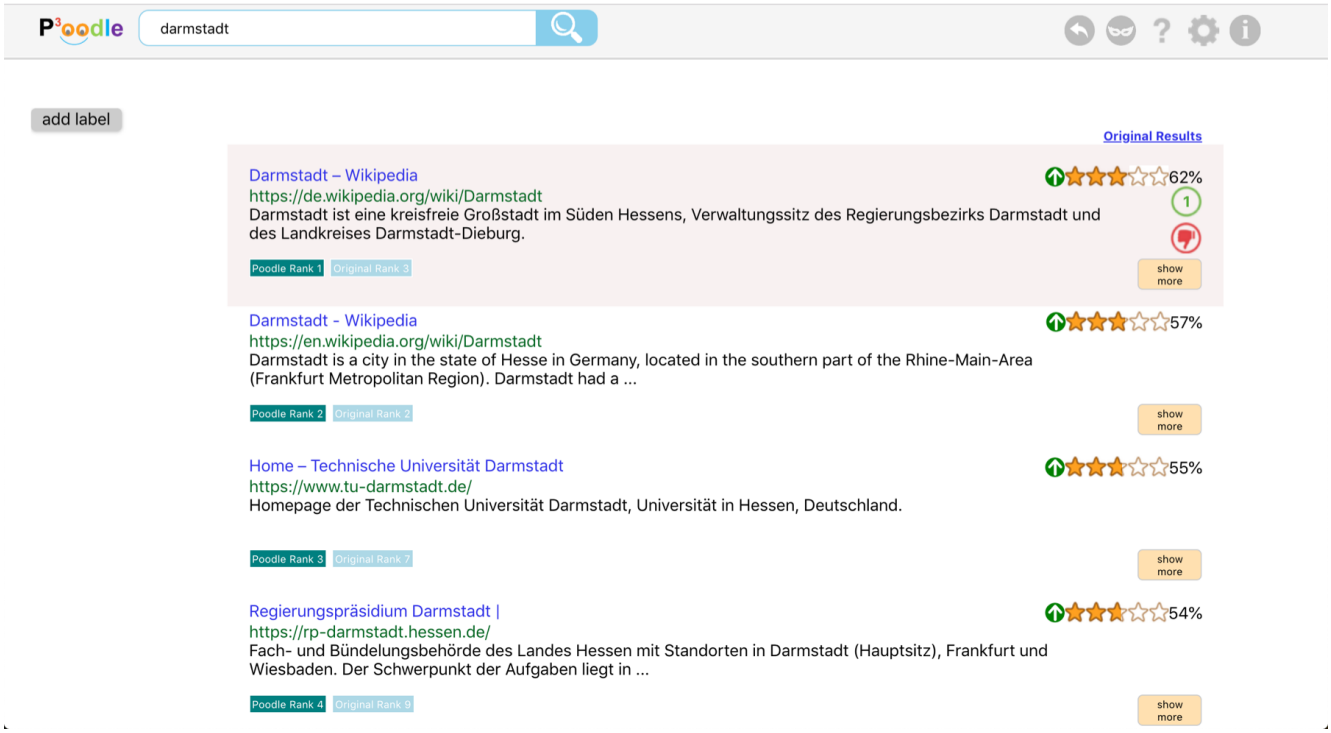

Figure 4.4: snippet

5 User Study

The ranking is an important task that every search engine has to execute. That is why it is essential to evaluate the quality of the ranking process as it gives the opportunity to assess the currently employed methods, compare it with other familiar methods and maybe open the door to developing new ranking strategies with higher quality results. The evaluation requires the availability of a data-set that include queries, the retrieved documents, and judgments. Usually ranking searcher use in the evaluation phase, one of the open source benchmark data-sets for learning to rank such as LETOR[22], in order to evaluate Poodle ranking, we decided to collect our own data-set through a user study because it allows us to generate data with a personalized characteristic.

The goal of the user study in the first place is to gather a good sample of history files that we can use later for the evaluation, as well as investigating the participant's opinion about Poodle and which improvements they would like to see in the future version. The Poodle setting used during the user study is the default setting expect the number of the search results which we fixed at 20, the used algorithm which varies from one participant to another, and the use of implicit feedback which we turned off as we are only interested on studying the benefits of explicit feedback. At the beginning of the user study a quick demonstration of Poodle usage is presented to the participants, then we asked them to answer a questionnaire.The first part of the questionnaire must be answered through Poodle search, at the same time participants were asked to rate search results of their choice positively or negatively using the rating buttons. This part includes on total 5 questions, which response should reflect the participant's tendencies, this is why we avoided informational questions during the task design. The complete list of questions is given in the following:

1. Find a Holiday destination.

2. Find an appropriate reservation for the selected destination.

3. Find some attractions that you would like to visit the selected destination.

4. Search again for a holiday reservation for another destination.

5. search for attractions for the second destination.

The first 3 questions aim to let Poodle learn a personalized model for the participant using the generated clicks, while the last 2 questions were designed to test if the gained information is reflected in the returned results. We expected that in the last two questions the top-ranked results are similar to the liked links in the first 3 questions. Poodle results were satisfying, most of the participants could notice the personalization effect, for example, a participant who searched for a cheap holiday offer to his wished destination get back cheap offer links ranked on the top when searching for another holiday destination offer. In the second part of the questionnaire, the usability of Poodle was investigated, some improvement suggestion was proposed and some bugs were identified by the participants. At the end of the study the history file is exported and stored. A total number of 9 students took part in the user study generating, on average per participant 6 queries and 12 clicks. The detailed recorded information per history file is given in Table 2

Table 5.1: History files summary

History File ID	Query number	Like clicks	Dislike clicks	Classifier
1	6	11	6	Rocchio
2	4	9	9	Rocchio
3	7	5	8	Rocchio
4	6	8	2	Naive Bayes
5	7	6	4	Naive Bayes
6	7	7	4	Naive Bayes
7	6	8	2	Perceptron
8	7	8	2	Perceptron
9	9	11	3	Perceptron

6 Poodle Ranking Evaluation

In order to evaluate Poodle ranking using the history files collected from the user study, for each file, queries were grouped by question, we computed for each group the average precision, NDCG, the average position of positive rated examples and the average position of negative rated example before and after each user click, then we averaged these values over all the history files on a first stage and on a second stage only between files with the same used classifier.

Figure 6.1 to Figure 6.4 show that Poodle ranking improves continuously over the iterations, the NDCG and the Average Precision values reach their maximum value 1 at the end of each question iteration, which means that the liked links are placed above all other results. This spectacular result is due to the higher ranks assigned to links identified as relevant by the user, a more realistic way to evaluate the ranking quality is to examine it before the relevance feedback is given, this could be observed at the first iteration of each question, we exclude here the first question because it has the same ranking as the source search engine. Although there is no user feedback at these iterations, the results are impressive in particular for the last 2 questions, the average position of positive rated links is around 2 there, for the last question there were almost no disliked links. The average precision and NDCG values exceed 0.75 which is even better than the first iteration. This confirms our expectation that Poodle will perform at its best in the fourth and fifth question

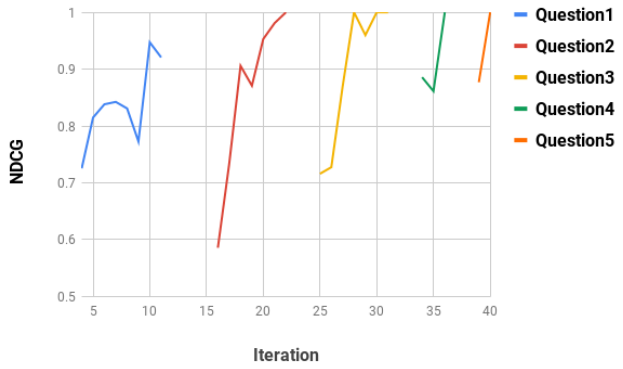

Figure 6.1: NDCG evolution per iteration for every question.

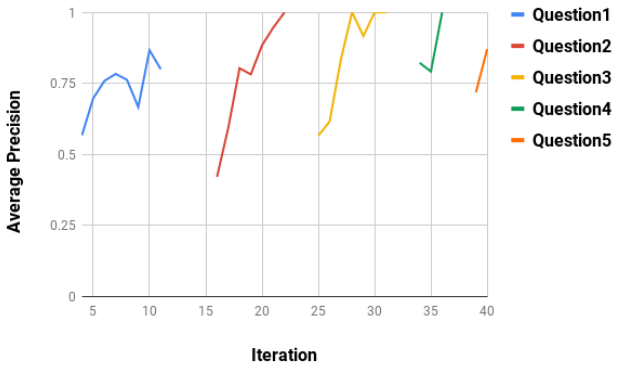

Figure 6.2: Average precision evolution per iteration for every question.

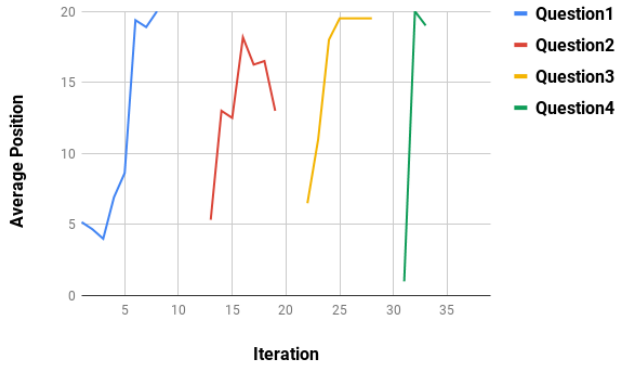

Figure 6.3: Average position of negative rated links.

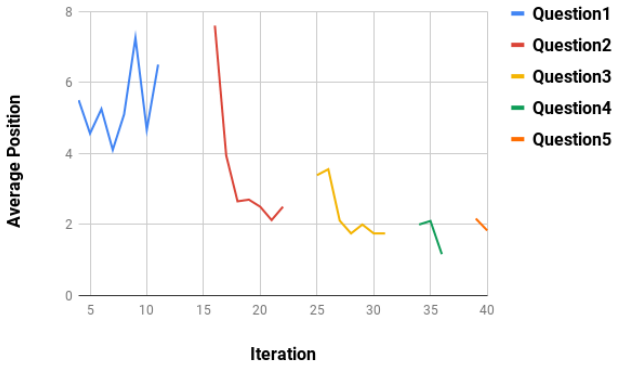

Figure 6.4: Average position of positive rated links.

Figure 6.5 to 6.7 show the performance of each classifier, we can notice that Rocchio performs significantly better than Perceptron and Naive Bayes particularly at the first iteration of each question, thereafter perform Perceptron and Naive Bayes better and succeed to catch up with Rocchio performance. Hence, it is recommended to use Rocchio as default classifier because it delivers a good result in its entirety.

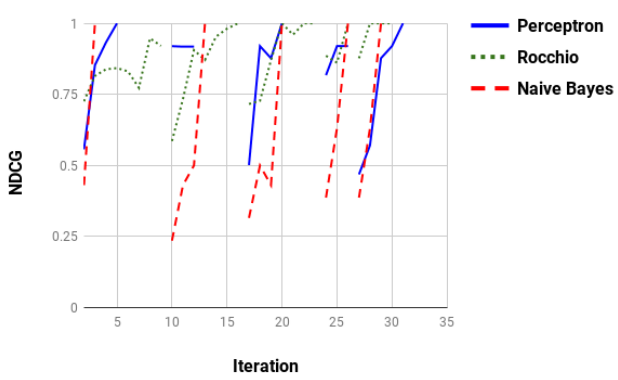

Figure 6.5: NDCG evolution for each classifier.

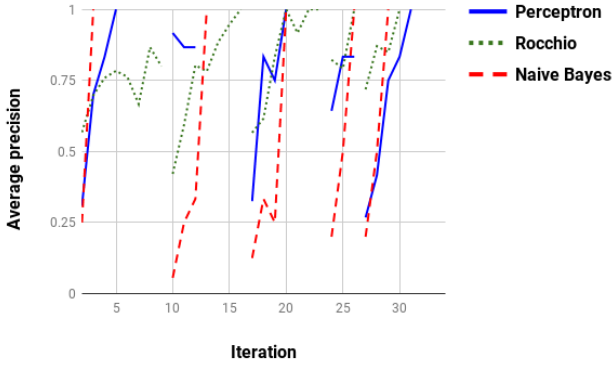

Figure 6.6: Average precision evolution for each classifier.

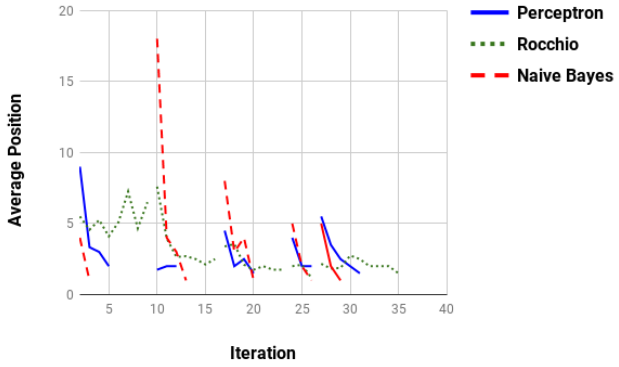

Figure 6.7: Average position of positive rated links for each classifier.

In the rest of this section, we compare Poodle with the original search engine as well as SVM and SVM Rank.

6.1 Comparison with the Original Ranking

Poodle has safety and privacy advantage over basic search engine because it saves the user interaction locally in the browser, but what if these search engine still return better results even prevented from user feedback? To verify this possibility, we compare Poodle ranking with the original ranking from 'Startpage'. For each query, we compute the average precision, the NDCG, the average position of positive rated example and the average position of negative rated example at each iteration. For the original rank these values are the same because the rank within a query did not change. The results given in Figure 6.8 to 6.11 show that Poodle outperforms the original ranking in all the graphs, except the average position of negative rated example, from the first to the last iteration of each query and with a significant difference in the fourth and fifth question's queries. Concerning the average position of negative rated example, the lack of dislike clicks and the stability of the original rank could explain the

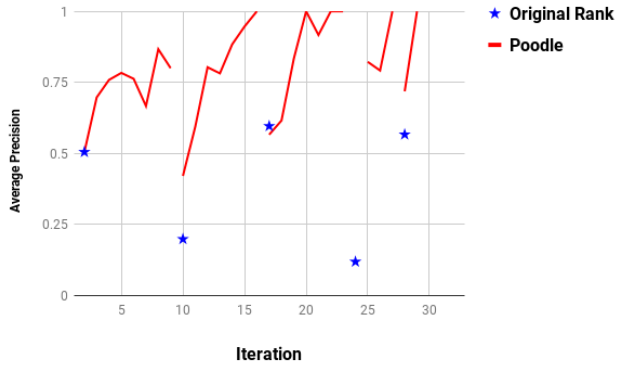

Figure 6.8: Average precision evolution of Poodle compared with Google.

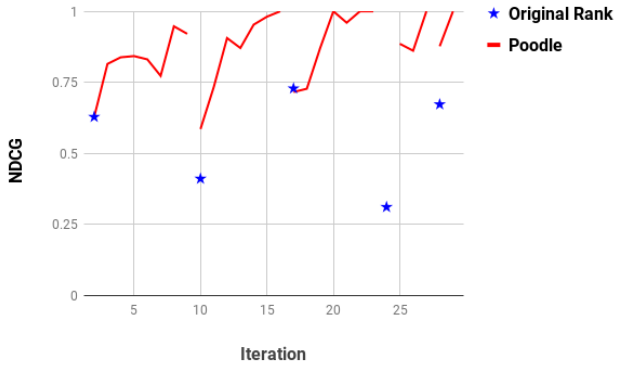

Figure 6.9: NDCG evolution of Poodle compared with Google.

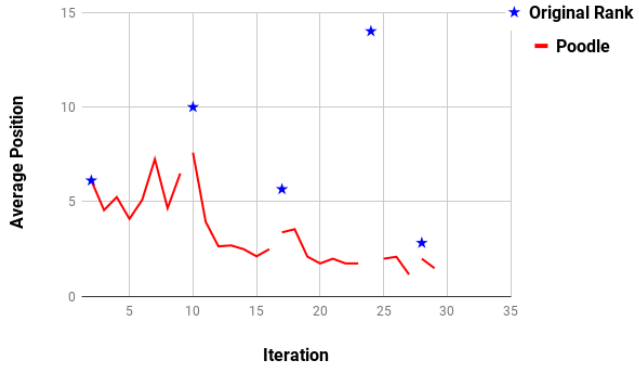

Figure 6.10: Average position of positive rated links of Poodle compared with Google.

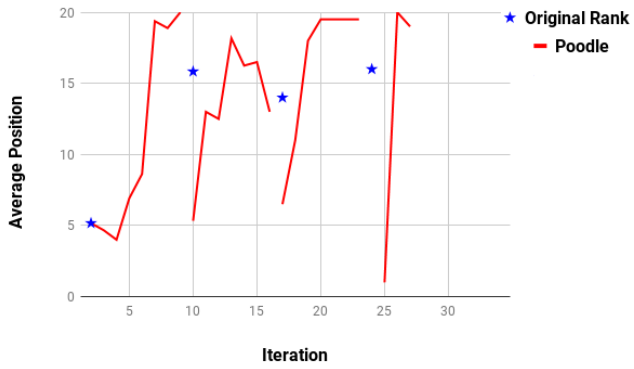

Figure 6.11: Average position of negative rated links of Poodle compared with Google.

advantage of the original rank over Poodle in the first iterations of each group, in the further iterations overtake Poodle this disadvantage and performs better.

6.2 Comparison with SVM and SVM Rank

We now compare Poodle ranking against SVM and SVM Rank to see if a more sophisticated learning algorithm with the same training data performs better than Poodle or not. First, we create the input files for both algorithms from the collected history file, this is done automatically using python scripts. At the same time we maintain the same training approach as Poodle, we repeatedly train after each new feedback and use the latest model to produce a ranking for the current query. Each line from the training file for the classification SVM, correspond to one feedback and has the following form:

$$< line >=< target >< feature >:< value > ... < feature >:< value > \tag{6.1}$$

where the target Value determines the class of the feedback, it has the value +1 for positive feedback and -1 for negative feedback, the feature-value pair denotes the order of the feature and its value. For example the following line:

$$-1\ 1:0.43\ 3:0.12\ 4:0.2 \tag{6.2}$$

represents a negative feedback where the first feature has the value 0.43 the third feature has the value 0.12 and the fourth feature has the value 0.2. Just like Poodle the feature are the terms extracted from

the link description after the application of the same text Preprocessing steps and associated with the corresponding term frequency, the order is set by the chronological appearance of each term. On the other side, all the returned results of the query search are represented by a line in the test file with the target value 0. After each train, we run the latest model with the test file of the actual query in regression mode. In the output file a prediction value to each line is returned, by ordering these values in descending order we get the predicted rank, then we compute for that rank the average precision, the NDCG value, the average position of positive rated example and the average position of negative rated example. The results are illustrated in Figure 6.12 to Figure 6.15 ,for the first few iterations SVM could not makes any predictions due to the small training data size, for these iterations we assigned 0 to the average precision, NDCG and the average position of the negative ranked example, while the average position of positive ranked example is represented by the average of all the position. SVM results were only good toward each group last iterations after user feedback are provided. At the beginning of each group, perform Poodle significantly better, especially on the fourth and fifth group where SVM in opposite to Poodle does not reflect the personalized feedback already gained.
We compare now Poodle against SVM Rank, we follow the same scheme as we did with the classification SVM, the only exception is that SVM Rank requires a different form of the inputs files. The target value is now used to generate pairwise preference constraints, an example with a high target value means that it is preferred over an example with a lower target value, for the examples with the same target values, there is no preference constraints generated, to determine the target value from the rating clicks we followed the following strategies:

- a positive rated example is preferred overall not rated example that a appear above it.
- the negative rated examples have the lowest target values amongst all the training example.

Preference constraints should only be produced within the same query, this why we add the special feature 'qid' to permit the generation of the preference constraints only for examples with the same 'qid'. For example, consider the following training data:

$$\begin{array}{l} 3\ qid:1\ 1:0.53\ 2:0.12 \\ 2\ qid:1\ 1:0.13\ 2:0.1 \\ 7\ qid:2\ 1:0.87\ 2:0.12 \end{array} \tag{6.3}$$

only the preference constraint $example_1 > example_2$ is generated because the first example target is greater than the second example target and both have the same 'qid' value. Figure 6.16 to Figure 6.16 show that SVM Rank results are similar to classification SVM with the advantage of classifying from the first iteration.

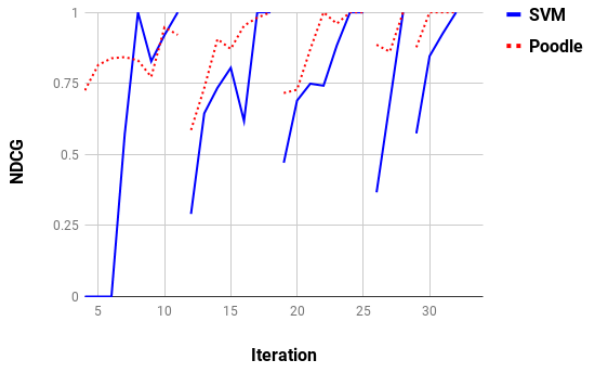

Figure 6.12: NDCG evolution of Poodle compared with SVM.

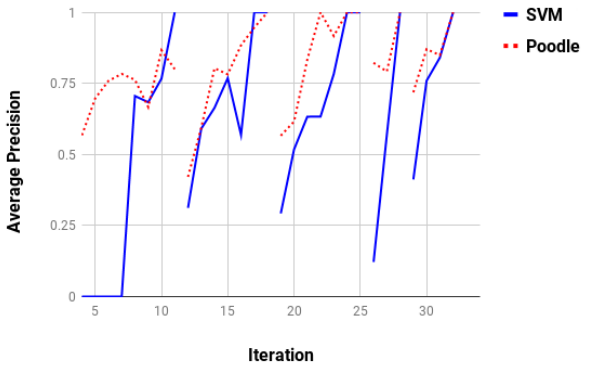

Figure 6.13: Average Precision evolution of Poodle compared with SVM.

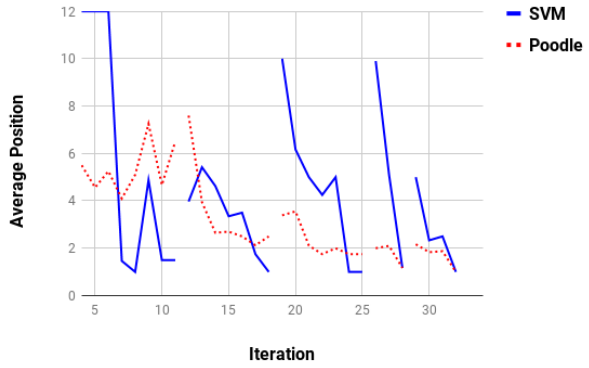

Figure 6.14: Average position of positive rated links of Poodle compared with SVM.

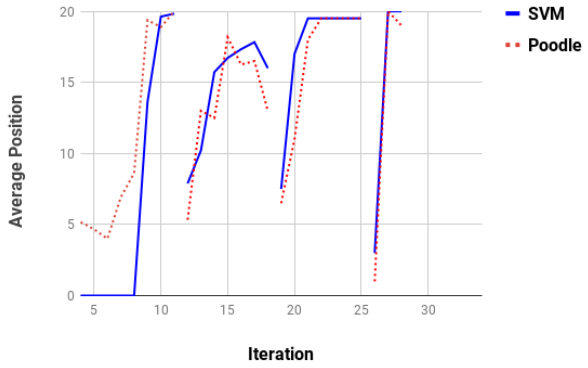

Figure 6.15: Average position of negative rated links of Poodle compared with SVM.

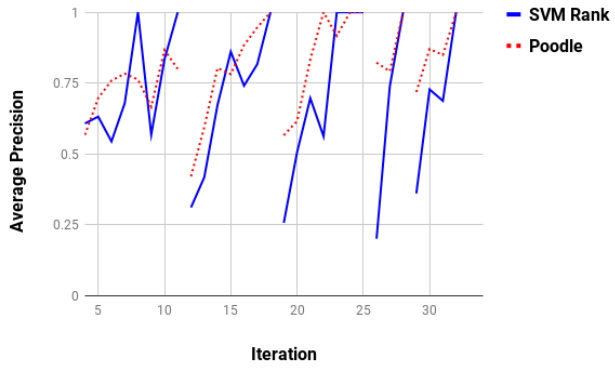

Figure 6.16: Average precision evolution of Poodle compared with SVM Rank.

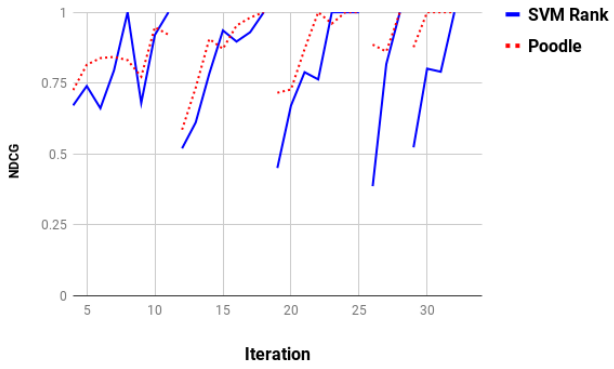

Figure 6.17: NDCG evolution of Poodle compared with SVM Rank.

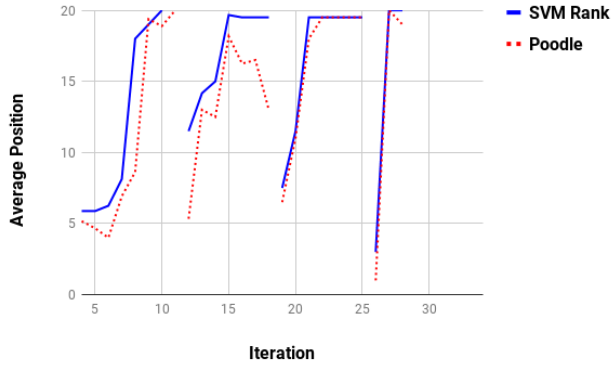

Figure 6.18: Average position of negative rated links of Poodle compared with SVM Rank.

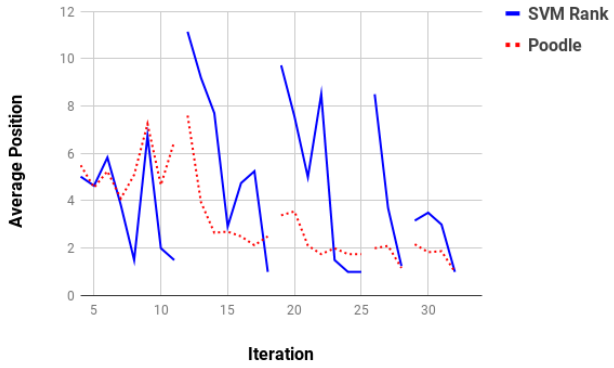

Figure 6.19: Average position of positive rated links of Poodle compared with SVM Rank.

7 Conclusion and Future Work

This thesis evaluated Poodle ranking performance using data collected from a user study, results show that poodle ranking outperformed the original rank, classification SVM and SVM Rank within small-scale queries and feedback. Nevertheless, we can not yet make a general statement, large-scale experiments are still needed. However, performing better than SVM Rank and classification SVM, point out that there is no need to add newly complicated classifiers, more interesting is to extend the available classifier with new techniques such as query expansion using new terms extracted from the relevant example and term reweighting based on the user judgments. Adopting a probabilistic model instead of vector space model might be an alternative possibility to enhance the overall performance.

Bibliography

[1] Porter M.F. An algorithm for suffix stripping.Program. 1980

[2] Porter M.F. Snowball: A language for stemming algorithms. 2001.

[3] C.J. van Rijsbergen, S.E. Robertson and M.F. Porter, 1980. New models in probabilistic information retrieval. London: British Library. (British Library Research and Development Report, no. 5587)

[4] Alper Kursat Uysal, Serkan Gunal.The impact of preprocessing on text classification

[5] J.Rocchio."Relevance Feedback in Information Retrieval", in The Smart Retrieval System: Experiments in Automatic Document Processing, Chapter 14, pages 313-323,Prentice-Hall Inc., 1971.

[6] Christopher D.Manning.Prabhakar Raghavan.Hinrich Schuetze.Introduction to Information Retrieval

[7] Christopher D.Manning.Hinrich Schuetze.Foundations of Statistical Natural Language Processing

[8] Thorsten Joachims.Optimizing search engines using click through data

[9] Pannaga Shivaswamy, Thorsten Joachims. Coactive Learning.

[10] Chris Buckley. Implementation of the Smart Information Retrieval System.1985.

[11] Thorsten Joachims. Optimizing Search Engines using Clickthrough Data.

[12] Thorsten Joachims, Laura Granka, Bing Pan. Accurately Interpreting clickthrough Data as Implicit Feedback.

[13] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In ICDM, pages 43–52, 2007.

[14] R. W. White, P. N. Bennett, and S. T. Dumais. Predicting short-term interests using activity-based search context. In Proc. of CIKM '10, pages 1009–1018. ACM, 2010.

[15] D. Sontag, K. Collins-Thompson, P. N. Bennett, R. W.White, S. Dumais, and B. Billerbeck. Probabilistic models for personalizing web search. In Proc. of WSDM '12, pages 433–442. ACM, 2012.

[16] https://snowballstem.org/

[17] Vikas Thada, Vivek Jaglan. Comparison of Jaccard, Dice, Cosine Similarity Coefficient To Find Best Fitness Value for Web Retrieved Documents Using Genetic Algorithm.

[18] C Cortes, V Vapnik. Machine learning.1995.

[19] Kalervo, Jaana Kek. IR evaluation methods for retrieving highly relevant documents.

[20] http://www.ke.tu-darmstadt.de/bibtex/publications/show/387

[21] https://www.startpage.com/

[22] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li.LETOR: Benchmark Dataset for Research on Learning to Rank for Information Retrieval

Poodle Benutzerstudie

TECHNISCHE UNIVERSITÄT DARMSTADT

Herzlich Willkommen!

Vielen Dank für ihr Interesse an unserer Benutzerstudie.

Diese Benutzerstudie zum Thema Poodle wird im Rahmen der Bachelorarbeit von Khalil Rahmouni am Informatik Fachgebiet Knowledge Engineering der Technischen Universität Darmstadt durchgeführt.

Poodle ist ein personalisiertes Such-Add-on, das von Studenten der Knowledge Engineering Group entwickelt worden ist. Im Gegensatz zu den bekanntesten Suchmaschinen, die viele Informationen von Nutzern speichern, speichert Poodle alle Daten lokal auf den Browser und verwendet einfache Algorithmen des maschinellen Lernens.

Ziel dieser Studie ist es, Daten zu sammeln um die Ranking Leistung von Poodle auszuwerten und mit anderen Algorithmen zu vergleichen.

Die Benutzerstudie dauert ca. 20 Minuten und wird in vier Schritten ablaufen:

1. Laden sie Poodle von Chrome Web Store herunter
2. Machen sie sich mit den verschiedenen Funktionalitäten von Poodle vertraut, indem sie sich die Quick Tour anschauen.
3. Beantworten sie die unten stehenden Fragen, indem sie eine Poodle Suche durchführen. Achten sie dabei auf folgende Punkte:
    - Sie können bei jeder Frage beliebig viele Suchen durchführen
    - Bewerten sie bei jeder Suche Links nach ihrer Wahl positiv oder negativ
    - Sie können einen Link mehrmals positiv oder negativ bewerten.

Figure 8.1: User Story

Fragen:

1. Sie wollen in den Urlaub reisen, aber sie wissen noch nicht wohin. Benutzen sie Poodle um ein passendes Urlaubsziel zu finden.
2. Suchen sie nach einem Urlaubsangebot für das ausgesuchte Urlaubsziel , das sie gerne buchen würden.
3. Finden sie Sehenswürdigkeiten, die sie gerne besuchen wollen.

Wir wollen jetzt prüfen, ob Poodle ihre Präferenz berücksichtigt hat oder nicht:

4. Suchen sie nach einem Urlaubsangebot für einen anderen Ort nach ihrer Wahl.
5. Finden sie Sehenswürdigkeiten an dem zweiten ausgewählten Ort.
6. Wie fanden Sie die Vorschläge, die ihnen Poodle präsentiert hat?

   ☐ Sehr gut

   ☐ gut

   ☐ es könnte besser sein

   ☐ schlecht

   ☐ sehr schlecht

7. Haben Sie die Personalisierung wahrnehmen können?

   ☐ ja ☐ nein

8. Wenn ja, können sie dies an einem Beispiel verdeutlichen?

9. Hatten Sie den Eindruck, dass die Personalisierung ihnen einen Vorteil gebracht hat? Können Sie das kurz begründen?

10. Glauben Sie, dass Sie von ihrer normalen Suchmaschine bessere, gleich gute, oder schlechtere Ergebnisse erhalten hätten.

    ☐ besser ☐ gleich gut ☐ schlechte

Allgemeine Fragen zu Poodle:

1. Wie fanden Sie die Bedienung des Add-Ons?

2. Könnten Sie sich vorstellen, statt ihrer regulären Suchmaschinen in Zukunft Poodle

Figure 8.2: User Story

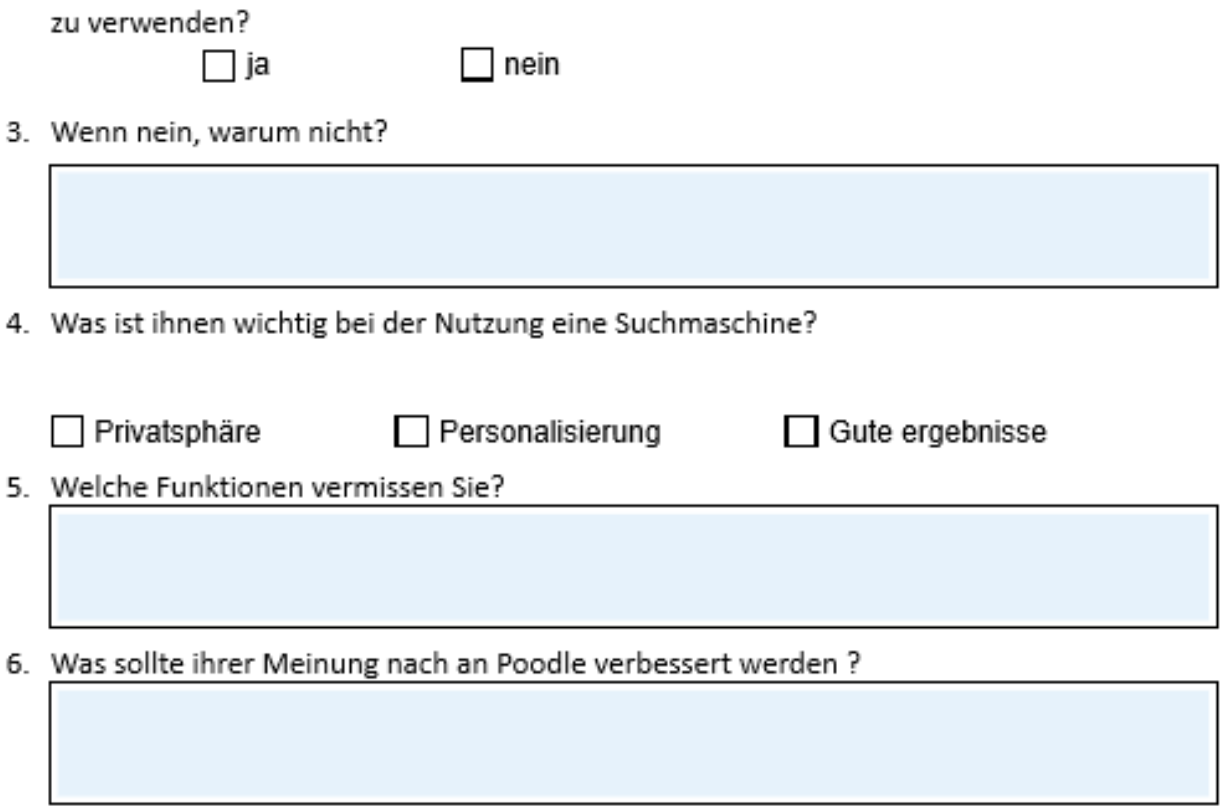

Figure 8.3: User Story