International Journal of Database Theory and Application
Vol.9, No.9 (2016), pp.139-148
http://dx.doi.org/10.14257/ijdta.2016.9.9.13

Query Categorization from Web Search Logs Using Machine Learning Algorithms

Christian Højgaard[1], Joachim Sejr[2], and Yun-Gyung Cheong[3*]

[1]*Pentia A/S, Copenhagen, Denmark,*
[2]*Schultz A/S, Valby, Denmark,*
[3]*Sungkyunkwan University, Suwon, South Korea*
[1]*chhoejgaard@gmail.com,* [2]*joachimsejr@gmail.com,* [3]*aimecca@skku.edu*

Abstract

This paper presents a data-driven methodology to disambiguate a query by suggesting relevant subcategories within a specific domain. This is achieved by finding correlations between the user's search history and the context of the current search keyword. We apply automatic categorization on each query to identify a list of categories which can describe the query given. To predict the categories of a user input query, we employed machine learning algorithms. We present the preliminary evaluation results and conclude with future work.

Keywords*: web query, disambiguation, categorization*

1. Introduction

Query disambiguation is the process of inferring the intention of a query [10]. Query disambiguation is a difficult problem to solve, since it requires knowledge about the intention of the user, and different users may not necessarily have the same intention with the same query. Also, the intention of a query for the same user may change over time depending on the context, which is dynamic. It is also difficult to determine when it makes sense to disambiguate a query. A query that is incorrectly disambiguated does not support the user in obtaining relevant information, instead it hinders the user [15]. This means that disambiguation has to be applied carefully. User intentions could be obtained through interviews, but it is only possible to cover very limited areas of information and it is difficult to model user intentions over a longer period of time. The challenges of query disambiguation can then be summed up as: user variations, dynamic user behavior, disambiguation application and unclear user intentions. This makes query disambiguation a challenging problem to solve, since such human factors are difficult to model.

The problem of solving query disambiguation is related to other problems within the field of information retrieval. Query disambiguation is related to Word Sense Disambiguation (WSD), which focuses on disambiguating words with more than one meaning, as query disambiguation also needs to consider this sub-problem. Since the same query may also have different meanings depending on the user, personalization of the search is also related to query disambiguation. Query disambiguation is an interesting problem, because of the large amount of information that is available to users when searching while web search queries are short and ambiguous. Zhang *et al.* [12] reports that the average length of a web search query is about 2.9 words. This means that the user's intention is not always clear from the query, and that in turn means that it is difficult to provide the user with the right information. Solving the query disambiguation problem will therefore allow users to obtain information more efficiently and thus allow

[1] Christian Højgaard and Joachim Sejr are the first authors, and Yun-Gyung Cheong is the corresponding author.

ISSN: 2005-4270 IJDTA

the user to save time, and provide the user with more relevant information and less irrelevant information. Major search engines such as Microsoft Bing and Google are also shifting search towards understanding the words used in queries to better support the needs of the users [11].

To address the query disambiguation problem, various computational approaches have been presented, such as utilizing similarities in sessions between different users [10], clustering of search results and computing semantic similarities to the current query [1], and building user profiles containing contextual information [13]. Glover *et al.* [16] present a method for locating documents within a specific category or topic in web search engines through the use of query modifications. The method is based on a classification procedure that can recognize pages in a specific category. The classification is automatically trained on features extracted from documents and sites within a category. The evaluation shows that the method is effective to predict the category; modified query has 50% precision for personal home pages and over 80% precision for calls for papers, compared to the less than 8% and 2% when no query modifications were used. The approach has promising results, though the challenge of robustly mapping a query to a category is still an open question.

Our approach is data-driven, based on machine learning for creating potential solution to the query disambiguation problem, as proposed in [9]. We define the disambiguation goal as identifying relevant categories within a domain. We focus on automatic analysis of query logs to relate the query to specific categories of the domain that the user is searching within. Our approach disambiguates the query by associating it with the context of the user based on the user's query history. Specifically, we infer the relevant categories of a given query as the user's intention within medical and travel domains. For instance, suppose that the current user query is 'diabetes' where the previous searches were 'cycling sore knee' and 'alleviate pain'. Our system infers that the user intends to know about 'medication' of diabetes. On the other hand, the same 'diabetes' query can mean the subcategory as 'food' or 'diet' when the previous search keywords were 'cooking weight loss' and 'children obesity'. By categorizing searches we perform a high-level analysis of the queries and obtain a general description of a given query.

However, our approach entails some essential assumptions. First, we define disambiguation as finding a domain specific category for a search query, which we claim, is enough to disambiguate a query, if the domain specific categories cover the domain well. Second, our machine learning methods assume that all the input categories, from the categorized queries, are correct. This rough assumption is necessary for our machine learning method for the training phase considers the category input as ground truth.

We tested our method on the medical domain within the general web search domain - more specifically we focus on diseases, and we will use the two terms, medical domain and disease domain, interchangeably. In our definition, a query is categorized as in the medical domain if it contains disease names. The medical domain is interesting because of its practical and important usage in everyday life and the user tend to have little knowledge about the expected outcome. We also investigated the effectiveness of our approach in the travel domain. We define a query as being in the travel domain, if the query contains one or more travel destinations (countries, cities, landmarks *etc.*). The travel domain is different from the disease domain in that it is far less technical.

2. Our Approach

Our approach analyzes user search histories based on query categories and time intervals. Our approach consists of 5 steps as listed below (see Figure 1).

1. Pre-process the raw data to find the relevant data for disambiguation and extract search histories that contain one or more searches within the domain targeted for disambiguation.

2. Perform automatic categorization of search queries.
3. Divide searches into groups based on time intervals. The search queries in the search histories are placed in the intervals based on their search time relative to the current search query.
4. Learn the patterns in the data to find correlation between query and search history (see Figure 2). The patterns are learned from the categories in the time interval groups that are linked to the categories in the domain targeted for disambiguation through the occurrences in the search histories.
5. Using the trained solutions in the step 4, predict a subcategory of the current query based on the user search history.

The first three steps aim to create a representation of the query disambiguation problem. The fourth step learns the correlation between query and search history. The final step applies the solution on new data to find the relevant sub-category. The Figure 1 [9] represents search history instances for the medical domain. The searches are prepared for the disambiguation training by dividing them into time intervals based on their associated timestamps. For each of the categories present in the search history, the time interval(s), which they are connected to, is appended to the category and the category is associated to the medical category of the current search which in this example is 'treatment'.

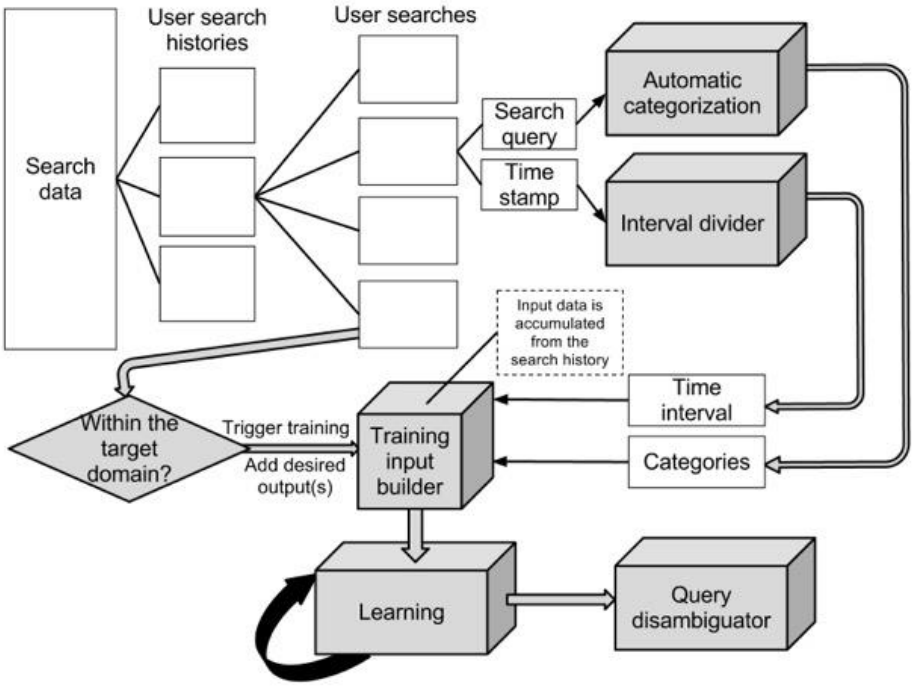

Figure 1. Processing Diagrams

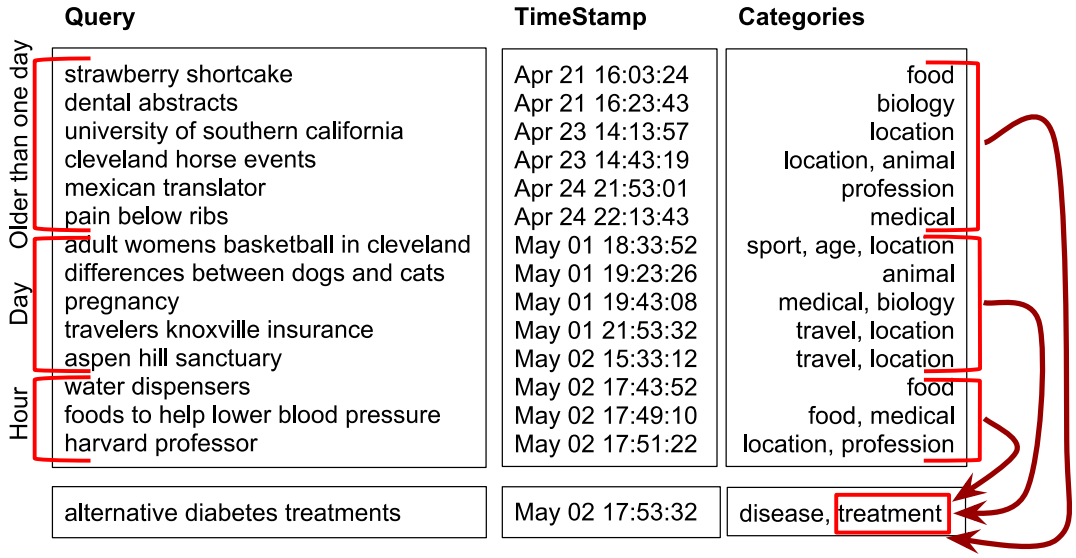

Figure 1. The Search Query History for the Medical Domain [9]

2.1. Data Pre-processing

For our evaluation, we used the AOL web search logs[2] which became available the public by accident. The AOL search data contains approximately 650,000 user search histories and 36 million searches over a period of up to three months. The data contains an anonymous user ID, query typed by the user, a time stamp for the query, the rank of the search result page that the user clicked on, and the URL of the search result page that the user clicked on such as 'www.wikipedia.org'. The raw AOL search data contains lots of search queries and search histories, therefore we need to extract the data that are relevant for query disambiguation. First, we eliminated searches with no query, queries with only numbers, site lookup searches (Google, Yahoo, Hotmail, Amazon *etc.*), and duplicate searches. Then, we reduce the data by removing irrelevant search histories. The relevant search histories should include medical or travel keywords, such that contain at least one secondary keyword (*e.g.* diabetes diet). This ensures that the data used for training and evaluation has one or more categories. The reduced data set were divided into three groups: a training set of about 40% of the total data, an evaluation set of about 30% of the total data, and a validation set of about 30% of the total data. The final medical domain contained about 1,700 user search histories and the final result for the travel domain contained about 39,000 user search histories.

2.2. Categorization

The task of the categorization is to find multiple categories that a given query falls in. For instance, we may want the query 'play soccer diabetes' to belong to a sports category and a disease category at the same time. For this, we must associate as many words as possible to different categories, which makes the manual coding almost impossible. We considered four databases as the source of category tagging: WordNet [3], Wiktionary[4], DBpedia [7], and Freebase [6], and found that Freebase and DBpedia [7] are most relevant to our query disambiguation problem. The application of these automatic categorization on the pre-processed data results in 10 categories in the medical domain: anatomy, disease, drug, food, medical, protein, risk, symptom, specific treatment, and general treatment. For travel, 6 categories were identified: accommodation, attraction, destination, event, tourist, and transport.

Categorization converts the search queries from our AOL data set to a list of categories. For each search query we get zero or more categories. The first step in our categorization is to remove stop words from the search query. Stop words are some of the most common words used that have little lexical meaning and may distort our results. For instance, "but", "be", and "want" are ignored using a list of stop words. A simple check for the stop words are performed on every word in the query and any positive hits are ignored. Next step is to check the whole query using Freebase or DBpedia and then WordNet afterwards. The whole query is checked because the search queries are occasionally multiple words making up only one entity. If the whole query returns as a hit, some performance is saved because the query will not have to be checked as separate words. If the whole query does not return a result, the query is split up into separate words. The words are then checked for being disease or travel words using only Freebase or DBpedia. This is done by making combinations of the separated words of up to four words in a row and checking these combinations for being disease or travel entities. Any positive hits are removed and the search query is labelled as containing a disease category. The rest of the separated query is checked in combinations of up to three words in a row, using WordNet and either Freebase or DBpedia. The combinations only go up to 3 words in a row due to performance issues. Every time a combination of words returns as a hit, the corresponding categories are added to the query's list of categories.

2.3. Data Preparation

Our system takes the user search history as input to disambiguate the current query. The input consists of the categories that the query falls in and the query search time relative to the current query. We employ 6 types of time intervals: within 15 minutes, within 1 hour, within 24 hours, within 7 days, within 31 days, and the rest. The output produced by the trained disambiguation system is a list of possible categories relevant to the current input query along with a value for each category indicating the level of relevance. The category with the highest relevance value can be chosen as the user intent of the current query.

2.4. Categorization Learning

We employed simple artificial neural networks (ANN) and the Naive Bayes classifiers to learn the search query categories. The simple backpropagation, resilient backpropagation and NEAT are implemented using the Encog framework [5] which is an advanced neural network and machine learning framework. For the backpropagation training we use a standard sigmoid activation function and a bias for hidden layer and output layer. For setting the weights in the networks we use an Nguyen-Widrow randomization, which is an effective neural network weight initialization methods and it has been proven to decrease training time in many cases [14]. For the NEAT implementation, all settings (relating to species, crossover selection, mutation application *etc.*) are controlled internally by the Encog framework.

To tackle the risk of over-fitting, we ended the training process once the algorithm starts to over-fit to the training data. We compute a validation score alongside the network error, by performing en evaluation using the neural network of the current epoch and the validation data. We then compute a linear regression model on the validation score for the last 100 validations. Once the slope of the linear regression becomes negative, we terminate the training. Variations in the network topology were tested, although not detailed in this article due to limited space. The Naive Bayes Classifier is implemented using the Weka framework [8] which is a collection of machine learning algorithms for data mining tasks such as classification. The implementation consists of a set of classifiers - one for each domain output category. Each domain output category can in turn produce two outputs, relevant and not relevant, each of which has a value associated to it.

3. Evaluation

We carried out evaluation experiments with regard to the representation of the domain to explore the significance of the attributes of the representation. This gives an indication of when it makes sense to disambiguate. The evaluation procedure is listed below:

- Input as three time intervals: within one hour, within one day and older than one day
- Input as six time intervals: within fifteen minutes, within one hour, within one day, within one week, within one month and older than one month
- No limit on the number of categories associated to a single query
- A limit of five categories associated to a single query
- Only use the categories of the previous search as input
- Only use the categories of the previous search within fifteen minutes as input

For the disease domain, it is common for a query to have more than one category associated to it, and we therefore define that the query disambiguation solution in this domain should suggest two relevant categories: a primary category and a secondary category. In the travel domain, only one suggestion is used. For the irrelevant categories, we define that the query disambiguation solution should suggest half of the categories (rounded up) as irrelevant. For comparison we perform an evaluation using two simple "disambiguation" solutions: a random category suggestion mechanism and a static category suggestion mechanism that always suggests the categories with the highest number of occurrences in the data as the relevant categories. This will provide us with a baseline, when we analyze the results.

To evaluate the effectiveness of our approach, we used the F_1 measure, which balances between precision and recall. F_1 is computed as *(2 X precision X recall) / (precision + recall)*, where precision is the proportion of correct assignments among all search queries assigned to a particular category C_i, and recall means the proportion of correct assignments of a particular category C_i among all the search queries that should be assigned to the category C_i. We also computed 'relevance hits' as the successful prediction rate of the relevant category and 'irrelevance hits' as the rate of irrelevant category suggestions. For comparison we employed two simple disambiguation solutions as the baseline: a random category suggestion mechanism and a static category suggestion mechanism that always suggests the categories with the highest number of occurrences in the data as the relevant categories.

3.1. Results

For the disease domain, the best results are obtained from the neural networks and with no limit on the number of categories and three time intervals (F_1 of 0.19, Relevance hits of 42.15%, Relevance misses of 57.84%). No significant differences were found among different time interval scheme: the three time intervals (*i.e.*, within one hour, one day, older than one day) and the six time intervals (*i.e.*, within 15 minutes, one hour, one day, one week, one month, and older than one month). The best F_1 is obtained using six time intervals, while the best relevance hit rate, the lowest relevance miss rate and the best irrelevance hit rate is achieved using three time intervals. We experimented with only one specific type of disease setting and found no improvements compared with the general disease domain setting. The results were however improved when only the previous search is used (F_1 of 0.21, Relevance hits of 46.15%, Relevance misses of 53.84%). This is also the case when the previous search within fifteen minutes is used, where the best results for the disease domain are obtained (F_1 of 0.22, Relevance hits of 46.95%, Relevance misses of 52.97%). All the results obtained for the disease domain are significantly better than the random category suggestions (F_1 of 0.16, Relevance hits of 26%, Relevance misses of 73.99%). Compared to static category suggestions (F_1 of 0.08, Relevance hits of 45.94%, Relevance misses of 54.05%), our approach performed worse

for the relevance hit rate, the lowest relevance miss rate and the best irrelevance hit rate, but better in terms of F_1.

The travel domain also produced the best results when neural networks were used and with no limit on the number of categories and six time interval (F_1 of 0.37, Relevance hits of 59.32%, Relevance misses of 40.67%). The neural network employing the six time interval scheme outperformed that of employing the three time interval in all measurement aspects. Reducing the domain to only cover one destination improved F_1 but lowered all the other measurements. When only the previous search or the previous search within fifteen minutes is used, the results were similar to those where the entire search history is used (F_1 of 0.36, Relevance hits of 60.61%, Relevance misses of 39.38%). The performance of our approach for the travel domain were significantly better than the random category suggestions (F_1 of 0.21, Relevance hits of 23.64%, Relevance misses of 76.35%) and static category suggestions (F_1 of 0.14, Relevance hits of 55.28%, Relevance misses of 44.71%).

Table 1. Disambiguation Results for the Disease Domain for Input of Six Time Intervals: within 15 Minutes, One Hour, One Day, One Week, One Month, and Older than One Month

Category Limit	n	Learning method	Threshold value	F_1 measure	Relevance hits	Relevance misses	Irrelevance hits
5	2823	ANN	0.1	0.1839	37.76% (21.82% + 15.94%)	62.20%	48.06%
5	2823	ANN	0.3	0.1634	27.02% (19.73% + 7.29%)	61.21%	48.06%
5	2823	ANN	0.5	0.1079	13.99% (12.00% + 1.98%)	39.74%	48.06%
5	2823	ANN	0.7	0.0553	5.95% (5.31% + 0.63%)	19.69%	48.06%
5	2823	ANN	0.9	0.0190	1.84% (1.77% + 0.07%)	6.48%	48.06%
∞	2823	ANN	0.1	0.1996	40.20% (22.98% + 17.21%)	59.79%	49.76%
∞	2823	ANN	0.3	0.1787	30.28% (21.14% + 9.13%)	58.30%	49.76%
∞	2823	ANN	0.5	0.1199	15.62% (13.31% + 2.30%)	41.02%	49.76%
∞	2823	ANN	0.7	0.0613	6.23% (5.73% + 0.49%)	17.92%	49.76%
∞	2823	ANN	0.9	0.0157	1.45% (1.38% + 0.07%)	3.64%	49.76%
5	2823	NBC	0.1	0.1880	39.07% (20.75% + 18.31%)	60.89%	50.58%
5	2823	NBC	0.3	0.1848	37.79% (20.47% + 17.32%)	61.06%	50.58%
5	2823	NBC	0.5	0.1712	35.56% (19.94% + 15.62%)	60.07%	50.58%
5	2823	NBC	0.7	0.1629	32.05% (18.66% + 13.39%)	56.85%	50.58%
5	2823	NBC	0.9	0.1395	21.99% (14.52% + 7.47%)	48.28%	50.58%
∞	2823	NBC	0.1	0.1966	39.92% (21.78% + 18.13%)	60.00%	49.94%
∞	2823	NBC	0.3	0.1977	39.31% (21.71% + 17.60%)	60.00%	49.94%
∞	2823	NBC	0.5	0.1923	38.15% (21.43% + 16.71%)	60.11%	49.94%
∞	2823	NBC	0.7	0.1910	36.34% (20.72% + 15.62%)	58.73%	49.94%
∞	2823	NBC	0.9	0.1790	31.42% (19.19% + 12.22%)	55.36%	49.94%

3.2. Discussions

The average relevance hits with regards to the different time intervals for the disease domain, suggest that the recent searches might have a greater impact on intention of the current search than the older searches. The F_1 measure reflects an equal importance of the categories, which may explain why some evaluation settings achieve a greater hit rate but a lower F_1 measure. The results obtained from the disease domain are generally worse than those obtained from the travel domain. This is due to the fact that the disease domain has more output categories than the travel domain, which entails that it is more difficult to determine the correct category. For the travel domain, the results are better than the naive static category suggestions. This is however not the case for the disease domain. The results showed better performance when there no restrictions were given on the number of categories for a query. This may indicate that more information helps better prediction, even if it may introduce more noise.

4. Conclusion

We present a query disambiguation technique based on categorization of the search query and the time of the search query relative to the current query that we are trying to disambiguate. We apply automatic categorization techniques (Freebase and DBpedia) on queries to identify a list of categories as the user's intentions of the query. We then

annotate the categories with the time interval(s), relative to the current query. We applied neural networks and Naive Bayes Classifier to learn the category of a given query from a training set. Our evaluation shows that the neural networks produced a higher precision than Naive Bayes Classifier in predicting the category of the current user query. However, the performance was not sufficient to be applicable for solving practical query disambiguation problems. It is therefore inconclusive if using only the categories of queries found in a user's search history is sufficient to disambiguate a new query, as the inaccuracy of the categorization might disrupt the learning of the disambiguation.

Acknowledgement

This research was supported by the MISP(Ministry of Science, ICT & Future Planning), Korea, under the National Program for Excellence in SW)(R2215-16-1005) supervised by the IITP(Institute for Information & communications Technology Promotion)

References

[1] G. Bordogna, A. Campi, G. Psaila, and S. Ronchi, "Query disambiguation based on novelty and similarity user's feedback", In FQAS, **(2009)**, pp. 179-190.
[2] "The New York Times article information", http://www.nytimes.com/2006/08/23/technology/23search.html
[3] "WordNet", http://wordnet.princeton.edu
[4] J. Wales, "Wiktionary, http://www.wiktionary.org".
[5] "Encog framework", http://www.heatonresearch.com/encog
[6] "Freebase", http://www.freebase.com
[7] "DBPedia", http://dbpedia.org
[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update", SIGKDD Explor. Newsl., vol. 11, no. 1, **(2009)**, pp. 10-18.
[9] C. Hojgaard, J. Sejr and Y. G. Cheong, "Query Disambiguation from Web Search Logs", Advanced Science and Technology Letters, (Information Technology and Computer Science 2016), Jeju, South Korea, vol. 133, **(2016)**, pp. 90-94.
[10] L. Mihalkova and R. Mooney, "Query Disambiguation from Short Sessions", In Beyond Search: Computational Intelligence for the Web Workshop at NIPS, **(2008)**.
[11] S. Gaudin, "Haul its search", Accessed
[12] http://www.computerworld.com/s/article/9225245/Google_works_to_overhaul_its_search, **(2012)**.
[13] Y. Zhang, B. J. Jansen and A. Spink, "Time series analysis of a web search engine transaction log", Inf. Process. Manages, vol. 45, no. 2, **(2009)**, pp. 230-245.
[14] D. K. Limbu, A. Connor, R. Pears and S. MacDonell, "Con- textual relevance feedback in web information retrieval", In Proceedings of the 1st international conference on Information interaction in context, IIiX, ACM, New York, NY, USA, **(2006)**, pages 138–143.
[15] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights", vol. 3, **(1990)**, pp. 21-26.
[16] M. Sanderson, "Word sense disambiguation and information retrieval on Research and development in information retrieval", In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR), **(1994)**, pp. 142-151.
[17] E. J. Glover, G. W. Flake, S. Lawrence, W. P. Birmingham, A. Kruger, C. L. Giles and D. M. Pennock, "Improving category specific web search by learning query modifications. Symposium on Applications and the Internet, pages 23–31, 2001.

Authors

Christian Højgaard, received his B.S. degree in Software Development in 2010 and his M.S. degree in Software Engineering in 2012 at the IT University of Copenhagen. Christian has since then worked as a software consultant and is currently employed at Pentia A/S in Aarhus, Denmark, where he primarily works on large web-based solutions using Sitecore (Content Management System).

Joachim Sejr, finished the B.S. degree in Software Development in 2010 and the M.S. degree in Software Engineering in 2012 at the IT University of Copenhagen. Joachim has since then worked at Schultz in Copenhagen with a system for managing social benefit for unemployed Danish citizens.

Yun-Gyung Cheong, received the B.S. degree in 1996 and the M.S. degree in 1998 in information engineering from Sungkyunkwan University (SKKU). In 2007, she received the Ph.D. degree in computer science from North Carolina State University, Raleigh, NC, USA. She is an Assistant Professor at Sungkyunkwan University, Korea. Her research interests lie in artificial intelligence with emphasis on its use in discourse planning for narrative, games, and user interfaces.