EUROPEAN SCIENCE FOUNDATION

Research Networking Programmes

Short Visit Grant ☐ or Exchange Visit Grant ☒

(please tick the relevant box)

Scientific Report

The scientific report (WORD or PDF file – maximum of eight A4 pages) should be submitted online <u>within one month of the event</u>. It will be published on the ESF website.

<u>Proposal Title</u>: Boosting for Learning to Rank with Query Log Data

<u>Application Reference N°</u>: 4907

1) Purpose of the visit

The purpose of my visit to the University of Amsterdam, concretely the Information and Language Processing Systems (ILPS) research group, was mainly to help me to start up my research on developing better "learning to rank" algorithms. At my home university, Czech Technical University in Prague, there is no expert or research group in the field of Information Retrieval. The small research group I am a part of has just begun to profile as such, hence the visit to ILPS was a remarkable opportunity for me to find out how a big and successful research group, which ILPS certainly is, works, and bring the lessons learned and try to implement them in our group.

Another very important motivation for me was to hopefully start up on a joint (long-term) project, which would allow me to keep in touch and continue to do the research with ILPS. I am happy to write down that this wish had come true. I can only wish the current findings of our joint effort had turned out more positive.

2) Description of the work carried out during the visit

During the visit I was studying ensemble learning to rank models, which were found to work the best in the field of learning to rank. I was focusing mainly on LambdaMART, the current state-of-the-art learning to rank algorithm, and was working on its implementation. I could have used an existing implementation, such as the one in RankLib (http://sourceforge.net/p/lemur/wiki/RankLib/), which is implemented in Java, but in order to be able to change things and understand all the gory details of it, I really needed to do the implementation myself. Since my language of choice is Python I found it very difficult to create an efficient implementation. Having an efficient implementation is very important because I am working with a dataset (which is provided by Seznam.cz the Czech commercial search engine) with hundred of thousands of queries and millions of documents, which can easily make the training times to exceed several days. In the end it was not wasted effort and the implementation is now available as an open-source project called RankPy (https://bitbucket.org/tunystom/rankpy). In terms of efficiency it is a favorable competitor to RankLib.

After finishing the baseline implementation I started my first experiments with it. LambdaMART is an iterative gradient boosting regression tree ensemble model and as such it has myriad of parameters, such as number of trees, maximum depth of trees, learning rate, etc. My obvious interest was to see how robust the algorithm is to different parameters settings and how robust is against label noise (which boosting algorithms usually fall victim to, and their performance degrades with increasing noise). I also conducted several experiments to compare the LambdaMART model, which is a list-wise learning to rank model, with different “point-wise” ensemble models (either classification or regression models), such as RandomForest, MART, McRank. Figure 1. illustrates the training and test performances of the listed models on the commercial dataset.

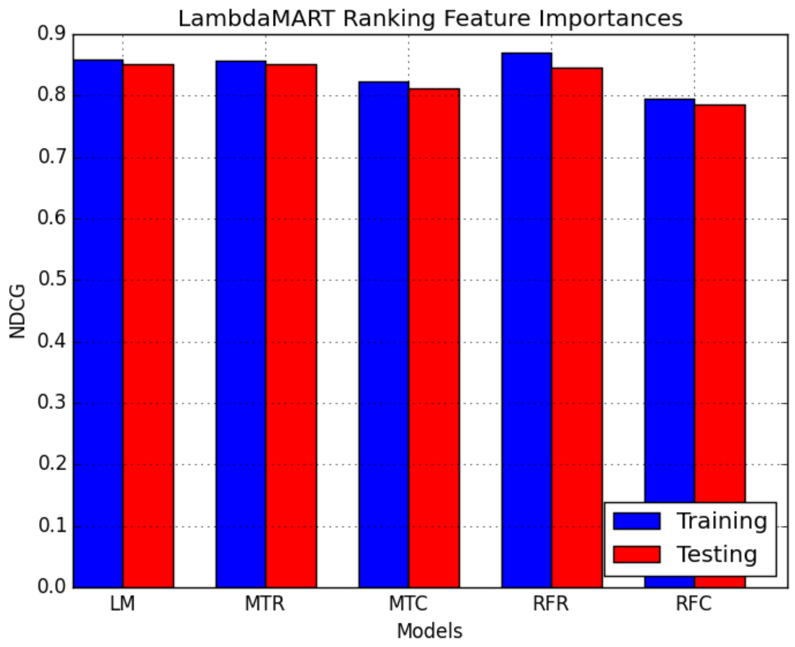

Figure 1: Performances of models (from left to right): LambdaMART, MART, McRank, Randof Forest regressor, Random Forest classifier.

The robustness against noise in labels was tested by training models (in this case MART and LambdaMART) on the dataset with different amount of (uniform) noise added to the document relevance labels. Figure 2 illustrates the results of evaluating the performances of the “noised” models (models trained on noised data) on the original dataset in terms of NDCG (with and without cutoff threshold). These two models were chosen because they are both from the family of gradient boosted tree models but apart from LambdaMART, MART is a “simple” regression model (the target variable is the document relevance). It may seem surprising that MART is beating LambdaMART the more noise is added to the relevance labels (for 0% noise the models reach similar results with an insignificant favor (in absolute numbers) towards MART). My explanation is that for MART the noise in documents is acting independently, but for LambdaMART, a noised document influences all the other documents for the same queries. These dependencies, which result from the way the lambdas in LambdaMART are computed (put simply, lambda of a document is a 'cumulative' gradient of the loss function with respect to the output of the ranker evaluated at the given document) make the noise to have greater impact on the performance of the model.

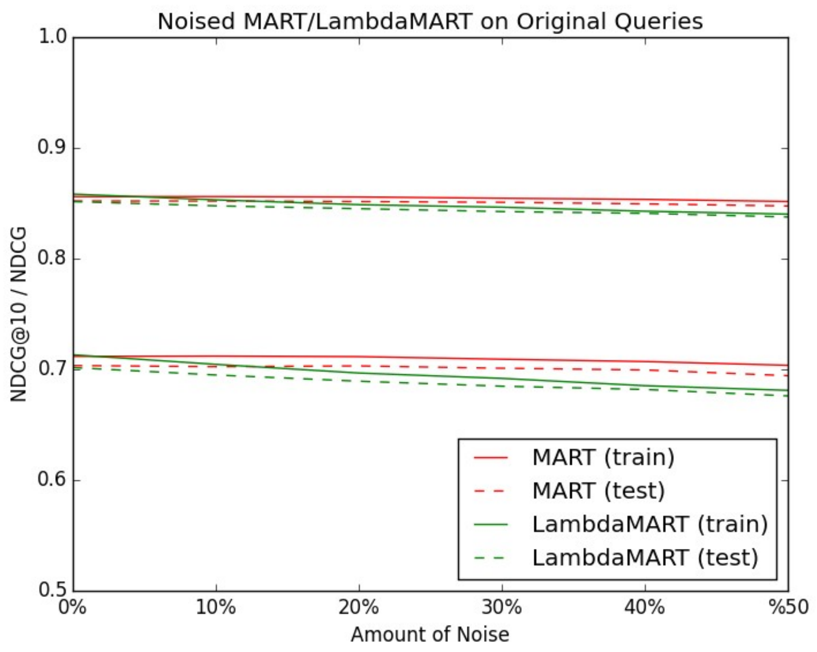

Figure 2: Influence of (uniform) label noise on performance.

The very surprising behaviour of LambdaMART is that it does not overfit with the increasing number of trees. In this regard it has similar behaviour to a random forest model. Figure 3 below illustrates the typical training (in red) and validation (in blue) NDCG performance of LambdaMART during training. The learning curves will eventually flatten out, but even going extreme and training up to 10,000 trees I was not observing any sign of overfitting, which would be detected by dropping red line.

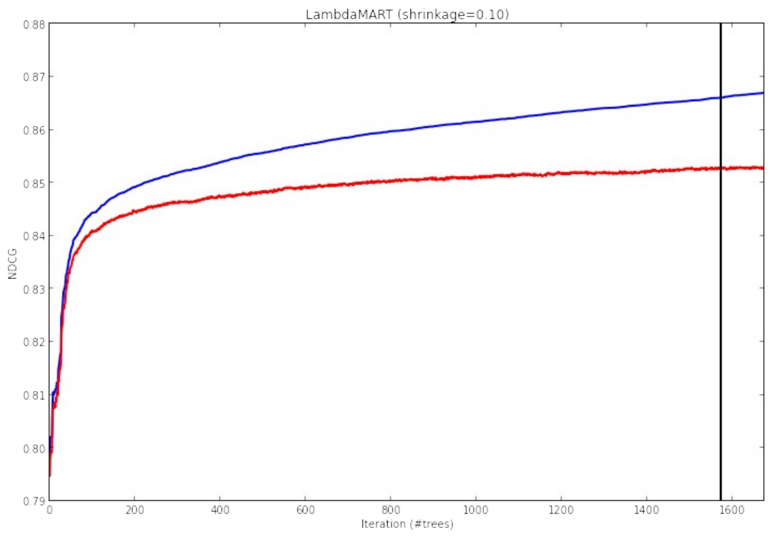

Figure 3: LambdaMART learning curves.

Very interesting is also to observe the influence of the learning rate on the training process. Figure 4 is the same model that was used to plot Figure 3, but the learning rate was decreased from 0.1 (typically works the best – a rule of thumb value) down to 0.01.

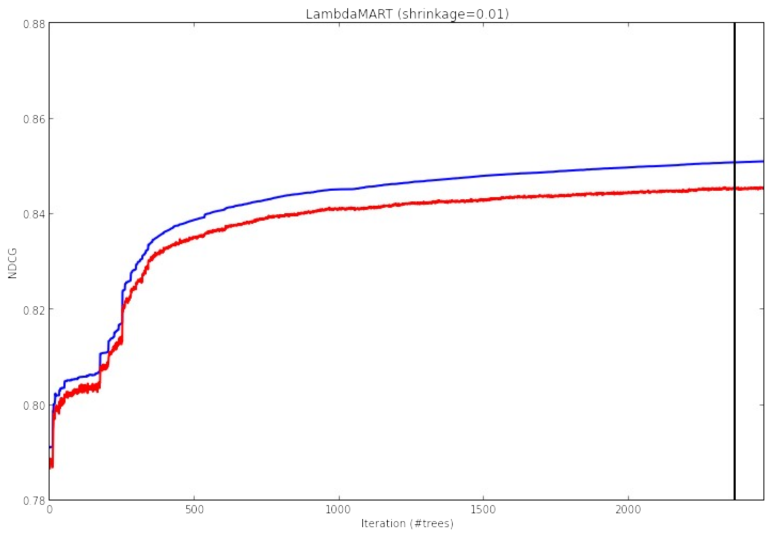

Figure 4: LambdaMART learning curves

From the progress at around 200 trees it seems like the model hit a sort of local optima, but it was capable of “crawling out” of it. Certain point of interest was to examine the importances of features (defined in Breiman, Friedman, "Classification and regression trees", 1984) from LambdaMART. Figure 5 illustrates a typical feature importances plot. This plot reveals that more than 10% (there ~300 features) of features are never used and tests revealed that we can forget more than ½ of the features without affecting the performance of the model.

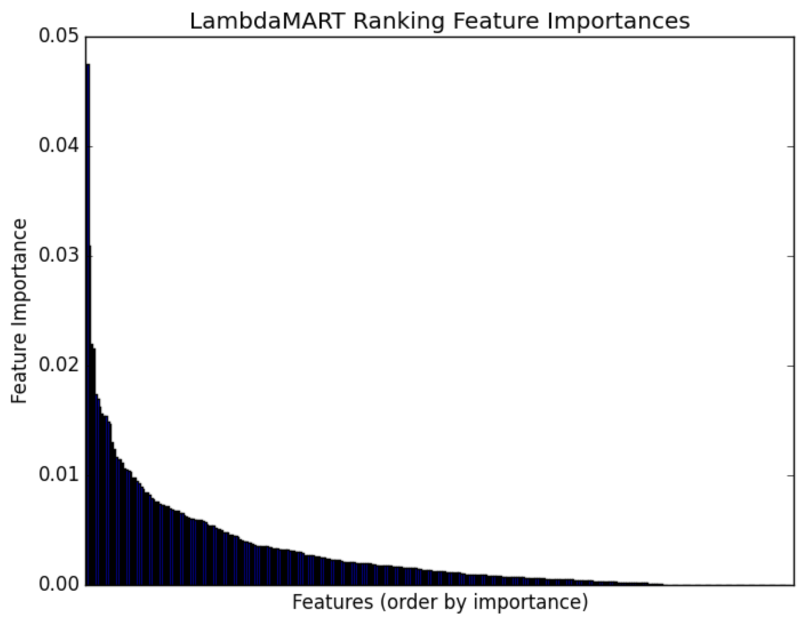

Figure 5: Feature importances found by LambdaMART

Final piece of insight into working of the LambdaMART model was to see what portion of the lambdas (accumulated gradients of the pairwise loss function used by LambdaMART with respect to the output of the ranker) of relevant documents were made from interactions (comparisons) with less relevant documents. This approach was adopted from Krysta Svore et al, "Learning to Rank with Multiple Objective Functions", WWW'11. Figure 6 illustrates the "influence" of less relevant documents on the correct ranking of the most relevant documents.

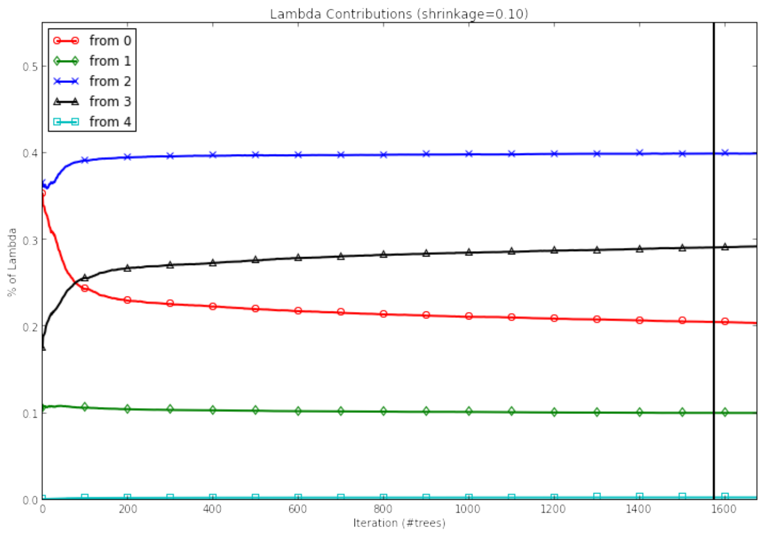

Figure 6: Lambda Influences of less relevant on most relevant documents.

This figure goes hand in hand with the learning curve plots depicted in Figure 3. If we interpret the figure such that the points in the graph depict the (average) force with which all less relevant documents (different colors show different relevance levels, maximum relevance label is 5) push the most relevant documents up at each iteration of boosting, than it can be clearly seen from the Figure 3 that LambdaMART's performance boost starts to stagnate after around 200 iterations. At the same moment the influences of all the less relevant documents start to converge on its “limiting value”, see Figure 6. It is also very important to note that the influences of less relevant documents is still very high – clearly the LambdaMART benefits from separating irrelevant documents from perfectly relevant ones (falling red curve), i.e. putting relevant documents up in the list and irrelevant documents down, but as we can see in Figure 6, the model completely fails to mimic the same behaviour with documents of higher relevance. Fixing this LambdaMART's “drawback” has become an appealing target of my research, since the mentioned work by K. Svore have not provided satisfactory solution (according to me) and I have not found any particular development in this line of thought. This issue reminds me of a similar problem in classical boosting (not gradient boosting) which Yoav Freund “solved” by his Boost by Majority algorithm (“noise-resistent” AdaBoost). This does not easily translates into the context of learning to rank, but certain similarities between the aforementioned works can be found.

My original ideas with using click logs and train LambdaMART from pairwise preferences inferred from them (and clicks throughout whole user sessions) have been postponed by my reluctance to start working on a problem before I fully understand the LambdaMART model (and gradient boosting generally) and the issues associated with learning from clicks. The former I could have done anywhere, but I think there is no better place where to find out all about the problems associated with inferring relevance from clicks than the ILPS research group itself. The visit has armed me with a lot of knowledge I did not posses, and I know that I would have taken a wrong approach and would come to wrong conclusions if I did not get it first.

I also spent substantial part of my visit working on a joint project, which is build upon the hypothesis that an ensemble of specialized rankers – each ranker is trained on queries, which share common structure/intent – can work better than one global ranker. The work that was done before my visit supported the hypothesis, but similarly to previous works on this topic (e.g. “Ranking Specialization for Web Search: A Divide-and-Conquer Approach by Using Topical RankSVM” by Bian J. et al, “Learning to Rank User Intent” by Giannopoulos et al.) it was based on linear ranking models. Since I am focused on ensemble models it was theorized whether similar results would be acquired when such models like LambdaMART were used.

The query clustering methodology is very similar to previous works, a ranking model is trained for each query and the clustering is made by using distance metric (or similarity) defined on the models. This is very clear in case of linear models, in which case euclidean distance between weight vectors of the rankers are used as the (dis)similarity between the queries associated with the models. What is not clear is how this can be done with a models based on trees. Here, comes the ingenious idea not to use the rankers themselves, but the way the models rank the queries. I will withheld the details of our approach for future publication, but it can be said that Kendall Tau distance metric between rankings is used as the clustering metric. Figure 7 depicts the distance matrix, rearranged according to the results of an agglomerative clustering algorithm, for MSLR dataset (with filtered 10k queries), where the query models were linear models trained via Duelling Bandit Gradient Descent.

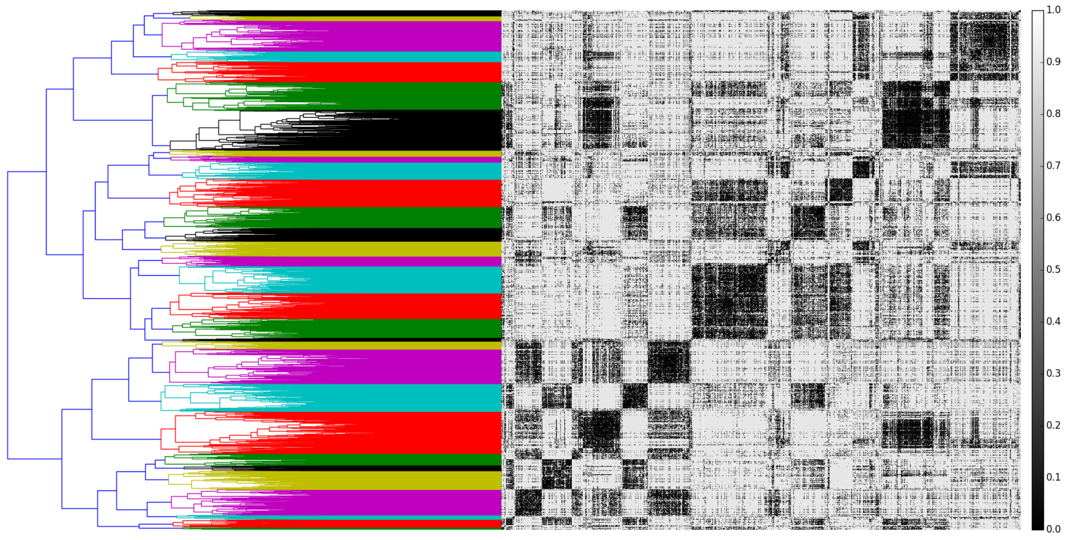

Figure 7: Query/models distance matrix. Each element correspond to a distance between associated queries/models. The black squares on the diagonal are typical for successful clustering.

The clusters were validated such that the performance of the models trained on (training) queries from the individual clusters were compared with the performance of one global model, trained on all (training) queries. The testing queries were selected out of the same cluster. It turned out that the local (cluster) rankers are defeating the global ranker, which has supported the original hypothesis. But there is a huge if – in the model comparisons we are “cheating”, because in real life we do not know which local ranker to use on a new test query, and because the ensemble ranker is not only as good as its individual rankers, but it also heavily depends on the quality of the membership function the result must be taken just as a small step towards developing a better ensemble ranking model.

As I mentioned earlier the clustering with linear models has been already accomplished, I only validated the results and made a few adjustments to make the process less time-demanding. My job was to take this idea and try to accomplish similar results with more complex models, such as MART, LambdaMART, RandomForest. Sadly, I need to state that the current results are unsatisfactory, i.e. based on them we cannot refute nor support the clustering hypothesis. Figure 8 depicts one out of many clusterings which came out for LambdaMART (all other models that were tested came out with similar results).

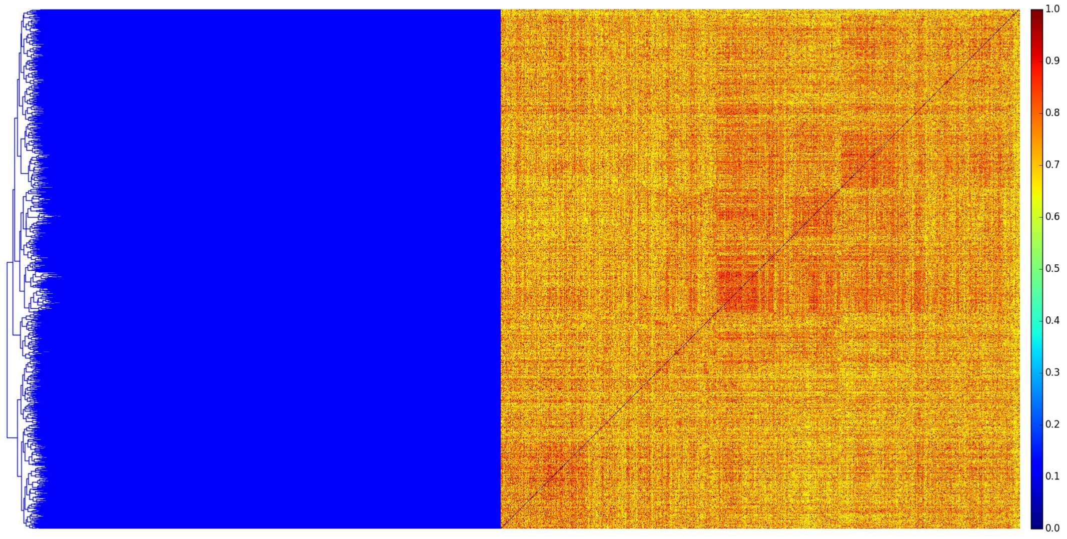

Figure 8: One out of many bad distance matrices made by clustering LambdaMART models for queries.

The figure hardly shows presence of any clusters. Why it is the case that the approach is not working with any of the more complex models was not yet fully identified. It seems that for query model comparisons there is too much variance within the structure/ranking behaviour of the ensemble models (compared to linear models), which has not been successfully accounted for in cross-validation (which is performed despite the fact that we are training the models on single queries). Overfitting is not an answer since on a single query the linear model overfits as easily as LambdaMART.

Finally, I would like to mention I was also attending the Information Retrieval courses and have an opportunity to give a presentation about LambdaMART and Learning to Rank. I have also collaborated on a paper under the working name “A Comparison of Retweet Prediction Approaches” with Hendra Bunyamin and Maarten de Rijke, which is planned to be submitted on ASONAM 2015.

3) Description of the main results obtained

So far my individual work (described in the previous section) was more of an exploratory nature, hence the results worth to mention according to me are the development of the first practically usable implementation of LambdaMART in Python. The implementation through its visualization capabilities and logged information is very practical to get insight into the inner workings of the model. Thanks to these insights I was capable to see the “flaws” in the algorithm and thanks to having implemented all the gory details myself I am more than ready to make an experimental changes that I hypothesize could make it work better.

The current results of the joint project (described in the previous section) would deserve to turn out better but the work so far resulted in a new framework for query clustering which is now used to run further experiments. The disappointment that the clustering is not working for LambdaMART is at least making me try to develop a method that would reveal whether or not the model (for its complexity) can cluster queries in some sense “internally”.

4) Future collaboration with host institution (if applicable)

I am still working and hopefully will be cooperating in the future with Masrour Zoghi, who was my contact person at the University of Amsterdam. The joint project was described in the Section 2 – the ultimate goal of the project is to devise an ensemble of query specialized rankers, which would work significantly better than one single ranking model. Current search engines are usually running on the latter, i.e. have a single ranking model and does not take into account the different nature of various queries. Hence there lies the potential to make a practical impact on the field of information retrieval if our efforts succeed.

5) Projected publications / articles resulting or to result from the grant *(ESF must be acknowledged in publications resulting from the grantee’s work in relation with the grant)*

No publication from my own work neither from the joint project have been submitted yet, but it is certainly planned. I am currently working on a broad technical report summarizing my findings and work done during my visit. I certainly acknowledge ESF's support in any publication that will results from the work I did during my stay at the University of Amsterdam. As for the joint project, I am not the leading person behind it (Masrour Zoghi is), but as a major collaborator I can make sure that the acknowledgement will be part of the publication, which I can refer to as “Query Specialized Clustering”.

6) Other comments (if any)

I feel that I need to express my deep gratitude to professor Maarten de Rijke, the leader of the Information and Language Processing Systems (ILPS) research group at the University of Amsterdam, for letting me stay and be for a short while a part of an amazing collective of smart and motivating people. I would especially like to thank to Masrour Zoghi who has become my colleague and who has played the role of my supervisor during the visit. I am very grateful that I can continue to do the research with him and discuss with him all the crazy ideas that are constantly coming on my mind.

To make long comment short, the stay at ILPS for me was a very eye opening and mind broadening experience, which will certainly influence the way I do my work. For giving me this opportunity I need to send my kindest regards to ESF as well because without its financial support it would not be possible for me.