<table>
<thead>
<tr>
<th>No.</th>
<th>Project Domain</th>
<th>Project Name</th>
<th>Company</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Information retrieval, NLP</td>
<td>Answering new questions using past Yahoo! Answers</td>
<td>Yahoo Lab</td>
<td>2</td>
</tr>
<tr>
<td>46</td>
<td>Data mining, Information retrieval</td>
<td>Automatic answers summarization in Yahoo Answers</td>
<td>Yahoo Lab</td>
<td>3</td>
</tr>
<tr>
<td>47</td>
<td>Information retrieval</td>
<td>Improving query evaluation using WAND</td>
<td>Yahoo Lab</td>
<td>4</td>
</tr>
<tr>
<td>48</td>
<td>Recommendation systems, low-rank matrix factorization, matrix completion</td>
<td>Low-rank sparse matrix factorization based recommenders</td>
<td>Yahoo Lab</td>
<td>5</td>
</tr>
</tbody>
</table>
Company: Yahoo! Labs

Project Domain: Information retrieval, NLP

Project: Answering new questions using past Yahoo! Answers

Supervisor: David Carmel

Project Description: Yahoo Answers (Y!A) is a community question answering site, containing hundreds of millions of questions about diverse topics, such as sports, healthcare, entertainment, politics, science and many others. In Y!A, askers post questions that consist of a title, a short summary of the question, and a body, containing a detailed description of the question. The question can be answered by other Y!A users and the asker may choose one of them as a best answer. Once a best answer is chosen, the question is said to be “resolved”.

In this project we will build an automatic answerer (a bot) for new questions submitted on Y!A site that will answer a question based on existing answers of similar resolved questions. When a new question arrives, the automatic answerer will search for similar resolved questions and will answer the question based on their answers. The project includes several steps: (1) Building a search system over Y!A data, and over several other CQA systems, that is able to identify resolved questions that are similar to a new question; (2) Identifying whether one of the answers of the retrieved resolved questions could provide a satisfying answer to the new question. If yes, provide the answer to the user; and (3) Evaluate the effectiveness of the automatic answerers on real new questions submitted to Y!A.

A similar answering bot has been built in our lab two years ago [Shtok et al WWW’12]. In this project we would like the students to first re-implement this work to be used as a state-of-the-art baseline, and then expand on [Shtok et al WWW’12], including the following (additional directions are possible): (a) Only some new questions can be fully satisfied by similar resolved questions. Classify new questions, which could be potentially answered by previous answers (e.g., because the asker wants only a human response) and therefore should be attempted by the answering bot; (b) Expand the sources of answers available to re-use to other CQA sites such as StackOverflow and Quora. The challenge is to reconcile the different signals available in these sites for finding related questions and good answer candidate selection; (c) Extend the question retrieval to include equivalent question variations to go beyond surface similarity; and (d) Resource selections: identify the most appropriate resources to be used by the answering bot for a given new question.

Evaluation: Run the answering bot and its variants over Y!A site and measure its popularity as reflected by the number of best-answer feedbacks it gets from real users.

Programming Languages and Development Platform: Java, Eclipse, Apache Lucene, Linux

Courses:  
- 236756 Introduction to machine learning  
- 236299 Introduction to NLP  
- 236303 Project in NLP
Company: Yahoo! Labs

Project Domain: Data mining, Information retrieval

Project: Automatic answers summarization in Yahoo Answers

Supervisor: Oleg Rokhlenko

Project Description: With emerging online information sources, information seeking behavior is changing. A variety of sources have emerged, including social or community question–answering (CQA) sites such as Yahoo Answers, Quora, Stack Overflow or Baidu Zhidao. A common and defining characteristic of these sites is that anyone can pose their information need on almost any topic as a question, and receive answers from the community of users that belong to that particular site.

In general, a CQA service has the following workflow. First, the asker posts a question in a CQA service and then people in the community can answer the question. After enough number of answers is collected, a best answer can be chosen by the asker or voted by the community. The resulting question and answer archives are large knowledge repositories and can be used to complement online search. In an ideal scenario, a search engine can serve similar questions or use best answers as search result snippets when similar queries are submitted. To support such applications, we have to assume the best answers from CQA services are good and relevant answers for their pairing questions. However, the assumption might not be true, since many times the best answer choice is motivated by incentives not related to the asker’s information need.

To find out what might be the alternatives to the best answers, we propose summarizing answers on a question page. For example, for the question “Which actress has the most seductive voice?” a summary of different people’s opinions ranked by popularity might be a better way for expressing the question’s answers.

In this project we want to implement several state-of-the-art algorithms for query focused multi-document summarization with application to the CQA field, a novel algorithm that will aim to outperform the known techniques, compare their performance and provide a scientific report summarizing the work.

Programming Languages and Development Platform: Java, Eclipse, Linux

Courses:
- 236756 Introduction to machine learning
- 236299 Introduction to NLP
- 236303 Project in NLP
Company: Yahoo! Labs  
Project Domain: Information retrieval  
Project: Improving query evaluation using WAND  
Supervisor: David Carmel

Project Description: While the amount of searchable data is constantly increasing, users have come to expect sub-second response time and accurate search results regardless of the complexity of the query and the of the data set. Thus, search system runtime performance is an increasingly important concern.

The WAND algorithm [Broder et al CIKIM’03] for query evaluation iterates in parallel over query term postings and identifies candidate documents using a preliminary evaluation, taking into account only partial information on term occurrences. Once a candidate document is identified, it is fully evaluated and its exact score is computed. As in the standard Document-at-a-time approach, WAND iterates in parallel over query term postings but the nature of the preliminary evaluation is such that it is possible to skip quickly over large portions of the posting lists. If the result of this “fast and rough” evaluation is above a certain threshold, varied dynamically during the execution, then a full evaluation is performed and the exact score is computed.

In this project, the students will implement two variants of WAND over the Apache Lucene search system and will experiment with their search performance over a large dataset of WEB pages. In the following, the students will investigate several features of the WAND algorithm and their effect on its performance. Some of the features to explore are:

1. Term upper bounds, used by WAND to determine the preliminary document score. WAND does not specify precisely how term upper bounds should be determined. We will explore several policies for setting those bounds in an optimal manner.
2. Many results requests: WAND is optimized to a typical search scenario when users are only interested in a few results (typically 10). When users ask for more results, WAND performance is degraded, mostly because the large time it takes to warm-up the heap of search results. We will explore several approaches for warming up the heap in such a case.

After finding many good results, WAND is actually reduced to an AND search. Can we improve beyond AND? This is an interesting question that should be further investigated. Some preliminary ideas can be tested.

Programming Languages and Development Platforms: Java, Eclipse, Apache Lucene, Linux

Courses:
- 236756 Introduction to machine learning
- 236299 Introduction to NLP
- 236303 Project in NLP
Company: Yahoo! Labs

Project Domain: Recommendation systems, low-rank matrix factorization, matrix completion

Project: **Low-rank sparse matrix factorization based recommenders**

Supervisors: Oren Somekh and Michal Aharon

Project Description: In recent years Matrix Factorization (MF) became a leading approach for recommender systems. In particular, given a matrix representation of a set of users-items interactions or ratings ($D \in \mathbb{R}^{NXM}$), MF approach finds two low rank factorizing matrices ($W \in \mathbb{R}^{NXK}$ and $H \in \mathbb{R}^{KXM}$), whose multiplication approximates the known entries and also predicts the unknown entries.

Inspired by recent developments in image processing, we intend to increase the number of columns of $W \in \mathbb{R}^{NXK}$ while maintaining a sparsity constraint of having no more than $k < K$ nonzero entries in each row. Such representation is expected to provide more accurate recommendations. In addition, such an approach can lead to a more meaningful factorization (making it a bit less “latent”), in the sense that the specific vectors chosen for the users’ representation may lead to revealing some characterization about the users. If so, such an approach would naturally lead us to better handling of challenges like diversity, clustering, topic extraction and more.

In this project we plan to study this low-rank sparse representation using real-life datasets (e.g., Netflix dataset) in terms of recommendation quality and resulting clustering quality. The project will be graded according to the achieved results, the novelty and creativity of the proposed algorithms, and comprehensiveness of the research.

Programming Languages and Development Platforms: Java, Eclipse, Linux

Courses:
- 234122 - Introduction to Systems Programming
- 236321 - Software Engineering Methods
- 236703 - Object Oriented Programming
- 236862 - Sparse and Redundant Representations and Their Applications in Signal and Image Processing