System for comparing topic suggestion algorithms using multiple evaluation properties

DMITRY KAZARIN

Master’s Degree Project
Stockholm, Sweden September 2014
Recommender systems are core components for many companies. These companies constantly improve their functionality aiming to maximize user satisfaction from their products. The only evaluation of the recommender systems usually done by the companies is performed in an online experiment on actual data leaving them without any offline tools to consider and postponing the quality assessment to the time when the algorithm is deployed and used in production. In this work, we describe a software evaluation tool for a selected recommender algorithm applicable for offline cases. We discuss different properties that are important for the assessment of the chosen algorithm, present the user behavior that best reflects expected real life attitude, debate various data sets (available on the Internet and provided by the company) suitable for the offline evaluation. We introduce an extensible software tool for offline assessment that is integrated into test environment created and maintained by Salesforce.com. The tool aims to be flexible allowing data sets, user behavior and metrics to be easily switched or used for evaluation of other recommender algorithms. We also describe a set of recommendations on how the selected algorithm could be improved supporting these enhancement suggestions with an evaluation performed using the implemented tool.
Acknowledgements

I want to thank Swedish Institute for sponsoring my master studies at the Royal Institute of Technology (KTH) in Stockholm, Sweden. My gratitude also goes to the amazing team at Salesforce.com I had a pleasure working with during my thesis project. Special thanks to my mentors Glyn Burton, Joel Palmert, my colleague Zandra Hird and my manager Kapil Agarval for their guidance, patience, support and valuable feedback.
Contents

Abstract i

Acknowledgements ii

1 Introduction 1
   1.1 Problem description ......................... 1
   1.2 Purpose ....................................... 3
   1.3 Goal ......................................... 3
   1.4 Method ....................................... 3
   1.5 Delimitations ................................. 3
   1.6 Outline ...................................... 4

2 Theoretical background 5
   2.1 Definition and classification of recommender systems .......... 5
   2.2 Related work .................................. 6
   2.3 Evaluation types, properties and metrics ....................... 7
      2.3.1 Prediction Accuracy .......................... 7
      2.3.2 Coverage .................................... 9
      2.3.3 Confidence ................................... 9
      2.3.4 Novelty ..................................... 9
      2.3.5 Trust, Serendipity and Diversity .................. 10
      2.3.6 Utility, Risk and Robustness .................. 10
      2.3.7 Privacy, Adaptivity and Scalability ............... 10
   2.4 Drawing conclusions ................................ 11
   2.5 Description of the algorithm used for topic suggestions .... 12
   2.6 Applicable properties ............................ 13

3 Research methods and methodologies 14
   3.1 Basic research methods .......................... 15
   3.2 Philosophical assumptions ........................ 15
   3.3 Research methods ............................... 16
   3.4 Research approaches ............................. 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>Research Strategies</td>
<td>17</td>
</tr>
<tr>
<td>3.6</td>
<td>Data collection and data analysis methods</td>
<td>17</td>
</tr>
<tr>
<td>3.7</td>
<td>Quality Assurance</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Data sets</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>Expected data</td>
<td>19</td>
</tr>
<tr>
<td>4.2</td>
<td>IEEE Xplore</td>
<td>20</td>
</tr>
<tr>
<td>4.2.1</td>
<td>IEEE Xplore structure</td>
<td>21</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Parsing an article page</td>
<td>21</td>
</tr>
<tr>
<td>4.3</td>
<td>Quora</td>
<td>23</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Quora scripts</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>Evaluation tool design</td>
<td>26</td>
</tr>
<tr>
<td>5.1</td>
<td>Tool design</td>
<td>26</td>
</tr>
<tr>
<td>5.2</td>
<td>User behavior</td>
<td>28</td>
</tr>
<tr>
<td>5.3</td>
<td>Accuracy evaluation</td>
<td>28</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Data set parameters</td>
<td>29</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Accuracy evaluation test</td>
<td>30</td>
</tr>
<tr>
<td>5.4</td>
<td>Diversity evaluation</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>Topic suggestion algorithm evaluation and recommended improvements</td>
<td>33</td>
</tr>
<tr>
<td>6.1</td>
<td>Accuracy evaluation</td>
<td>33</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Number of generated suggestion</td>
<td>34</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Topic threshold</td>
<td>35</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Topic Index</td>
<td>36</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Adjective parameter</td>
<td>37</td>
</tr>
<tr>
<td>6.2</td>
<td>Diversity metric</td>
<td>38</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions and future work</td>
<td>40</td>
</tr>
<tr>
<td>7.1</td>
<td>Conclusions</td>
<td>40</td>
</tr>
<tr>
<td>7.2</td>
<td>Future work</td>
<td>41</td>
</tr>
<tr>
<td>A</td>
<td>Script for retrieving data from IEEE Xplore</td>
<td>43</td>
</tr>
<tr>
<td>B</td>
<td>Script for retrieving topic URLs from Quora</td>
<td>47</td>
</tr>
<tr>
<td>C</td>
<td>Script for retrieving posts data from Quora</td>
<td>49</td>
</tr>
<tr>
<td>D</td>
<td>Accuracy evaluation</td>
<td>52</td>
</tr>
<tr>
<td>E</td>
<td>Diversity evaluation</td>
<td>62</td>
</tr>
</tbody>
</table>
Bibliography
Chapter 1

Introduction

In recent years various recommender systems became an important part of our lives. They recommend TV programs to watch, songs to listen to and are an essential core for different e-commerce websites, like Amazon \[1–5\]. More and more companies adopt recommender systems and Salesforce.com is not an exception from that tendency.

1.1 Problem description

Salesforce.com is an enterprise cloud computing company focusing on CRM and social enterprise solutions \[6\]. One of these solutions, Salesforce Chatter, is an enterprise social network \[7\], which utilizes different recommender systems for suggesting relevant users, groups, topics, files and other types of useful information. While developing a recommender system a company may face different problems. The system might not perform well for new users or items (also known as a cold-start issue); it might not scale properly to support the needs of the application; users grasp the system in a way that was not intended by the developers or system’s suggestions are not helpful for the users. These issues are a part percent of things that could go wrong. In many cases developers need to make tradeoffs between the problems, because it is impossible to eliminate them all in one solution. To deal with the issues and tradeoffs recommender systems might be evaluated using various properties (accuracy, coverage, novelty, etc.) and different metrics for those properties (Root Mean Squared Error, Mean Absolute Error, investigation of precision-recall curves, etc.).
The only quality evaluation of the recommender systems done by Salesforce.com is performed in an online experiment [8]. When a new or an updated recommender algorithm is deployed, the acceptance rate for the recommendations is logged. Access to the logged data is available only for company’s internal deployments (two applications only Salesforce employees work with) and not for the clients’ deployments due to privacy policy used by the company. Based on the collected data, it is possible to verify accuracy of the algorithm. The nature of the data available is highly specific, represents a software development process and creates a bias on the evaluation.

Even though the online evaluation is considered highly reliable [8], it has a drawback in how much time it takes to be able to make the actual assessment. To do so, first, the algorithm has to be deployed in production and, second, enough data has to be generated by users resulting in months of waiting for Salesforce.com (the company has one release every four months). This drawback increases the price of a mistake and might lead to reputation loss. To prevent this from happening, Salesforce.com is interested in exploring new types of quality evaluations that could cut the assessment time as much as possible without any high impact on the reliability of the results.

Offline experiments are less reliable than online evaluations or user studies [8], but they provide feedback prior to algorithm production deployment and might be used for omitting bad recommendation algorithms and/or for adjusting weights of various algorithms’ properties. The main challenge with the offline evaluation is the selection of a data set to be used for the assessment. Without unbiased user-generated input available it is necessary to find a set that best reflects the content and context of the expected data and model the same behavior as a user would have in a deployed solution.

The company is interested in utilizing the offline evaluation and by creating of a software tool that could be used for assessment of different recommendation algorithms by being able to switch data sets dynamically. The tool should also adopt a wide range of evaluation properties and metrics and integrate with company’s test environment to be used by everyone in Salesforce.com.
1.2 Purpose

The purpose with the thesis is to develop a software tool that adopts flexibility and is simple in use, as well as, to present possible improvements for a selected algorithm. The improvements are results of the quality evaluation and might lead to a higher customer’s satisfaction.

1.3 Goal

The main goal of the work is to deliver a software tool for offline evaluation. Another goal is to give a set of recommendations on how to improve a selected topic suggestion algorithm used in Salesforce Chatter. The improvement suggestions must be supported by the results provided by the evaluation tool. They might expose other quality properties of the algorithm besides accuracy, which could be useful for improving customers’ fulfillment from the product.

1.4 Method

The investigation is driven by Experimental Research to figure out how different evaluation properties may affect recommender systems’ quality. Positivism is the philosophical assumption used as a point of view for the conducted work to guarantee that the reality does not depend on an observer. The work also takes advantage of the Experimental data collection method together with the Statistics as data analysis method for drawing reliable conclusions while comparing quality of recommendation algorithms. More details about method selection are presented in Chapter 3.

1.5 Delimitations

The work is restricted to an assessment of only one recommendation algorithm as a proof that the designed system works. The algorithm makes topic suggestions based on a post content from Salesforce Chatter. Even though only one algorithm
is used, the evaluation tool is designed with an attention to flexibility concerns and can easily be used to evaluate other algorithms.

1.6 Outline

This thesis is structured as following: first, in Chapter 2 we show the theoretical background necessary for understanding the work completed, Classification of recommender systems and evaluation properties, related work on the software frameworks creation for quality evaluation of recommender systems, how the investigated algorithm works as well as which properties and metrics might be applicable for the algorithm under the consideration. Second, in Chapter 3 we discuss methods used for research approaches, for finding useful material and for reaching the thesis results. Chapter 4 introduces various data sets available for the evaluation arguing about their advantages and disadvantages. Design of the software evaluation tool is demonstrated in Chapter 5 along with its architecture discussion, flexibility concerns and proof of functioning. Chapter 6 presents the evaluation and improvements of topic suggestion algorithm. Chapters 7 concludes and summarizes the work that was performed and narrows down the paths for future investigations and research.
Chapter 2

Theoretical background

2.1 Definition and classification of recommender systems

Recommender systems are intelligent applications, which assist users in information-seeking tasks, by suggesting those items (products, services, information) that best suit their needs and preferences [9, 10]. This definition implies that users need help with searching for specific items due to either lack of experience in the specific field, either being new to the content or format of the system or because of being overflown with information that could be gained by the application. Amazon [5] and Netflix [11] provide, probably, the most discussed recommender systems for related goods searching and movie recommendations.

More formal definition of recommender systems was introduced by Adomavicius and Tuzhilin [12]. They define two very large sets: C to be the set of all users and S to be the set of all possible items that can be recommended. Besides, they use an utility function $u$ to measure usefulness to a user $c$ of an item $s$. Based on this notions, authors give a formal definition of a recommender system (2.1):

$$\forall c \in C, s_c^* = \arg \max_{s \in S} u(c, s)$$  \hspace{1cm} (2.1)

Utility function is application dependent meaning that the function could be anything from movie rating to amount profit gained by a company.
Regarding of how recommendations are created, recommender systems fall in three groups [8]:

**Content-based recommender systems**

Content-based approach relies on the data that could be extracted from user’s profile preferences, for example, rating history. The recommender system uses this data to compare with information available for items (for example, movie genre) and shows the goods that are most correlated with the user’s preferences.

The key issue of this approach is that it is very hard to apply the knowledge gained from one data type to another.

**Collaborative filtering recommender systems**

This method does not rely on the properties of items to be recommended, but only on profiles of similar users and their preferences. The approach comes with several disadvantages, like, scalability issues (need to process more users), cold start problems (very hard to recommend anything until enough information about the user’s profile gathered and could be used for finding similar users), etc.

**Hybrid recommender systems**

The method combines content-based approach and collaborative filtering approach in various ways trying to compensate some of their drawbacks.

### 2.2 Related work

Herlocker, Konstan, Loren, etc. give the most comprehensive evaluation of collaborative filtering recommender systems currently available [13]. Shani G. and Gunawardana A. go beyond collaborative filtering and also focus on other systems types evaluation as well as reasoning on how to make reliable conclusions while comparing two recommendation algorithms [8].

Ciordas C. and Doumen J. describe an industry perspective on how a content-based recommendation system should be evaluated and argue that the evaluation should go beyond the accuracy property [14]. Bobadilla J, Hernando A, Ortega F., and Bernal J. carry out the same idea, but in an evaluation framework for collaborative
filtering systems [15]. Besides that they point out which properties and metrics correlate with each other and which should be investigated independently.

Our work focuses on the automation of the evaluation for content-based recommendation systems, but can easily be extended for collaborative filtering approach. It aims to create a flexible software tool that can easily plug new metrics and change data sets.

2.3 Evaluation types, properties and metrics

Three types of experiments that could be used for evaluation [8]:

**Offline experiment**

This method works by simulating user behavior on a pre-collected data set. The experiment relies on an assumption that the user behavior when the system will be deployed would be the same as during the simulation. It is widely used for comparing algorithms and tuning their parameters at low cost, because it does not require any interaction with a user.

**User studies**

A user study is performed by questioning a set of users about their interactions with a deployed recommender system. Gained information is analyzed using quantitative and/or qualitative methods. An example: how a user’s navigation is influenced by recommendations about news.

**Online evaluation**

This type works on a real life data that is collected when the system has been deployed and used by the system’s target users. Evaluation might take a lot of time depending on company’s deployment schedule.

In the experiments various properties of recommendation systems could be evaluated.

2.3.1 Prediction Accuracy

Prediction Accuracy is arguably the most used and discussed property in the literature. Its attraction is also a result of the property being easily evaluated
in an offline experiment. It relies on an assumption that a prediction engine is a core component of a recommendation system and more accurate predictions would result in a higher user satisfaction [8].

In literature prediction accuracy is usually divided in three categories depending on how the recommender system works:

**Ratings prediction accuracy** is used in applications where a system tries to predict a rating that a user would give to an item. For example, 5-star rating on Netflix.

Two metrics are most used for this category: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Both of them compare ratings predicted by the system with ratings given by a user. The only difference between them is that RMSE penalizes large rating differences harder.

**Usage prediction accuracy** is valuable when a system tries to predict if an item would be useful for a user instead of guessing the rating that the user would give to the item. For example, a system that suggest tags for a given text.

For evaluation of this property the result data set is divided in four groups: True-Positive (recommended by the system and used by the user), False-Negative (used by the user, but not recommended by the system), False-Positive (recommended by the system, but not used by the user) and True-Negative (not recommended by the system and not used by the user). Precision and Recall, the two arguably most utilized metrics, can be computed from the data in groups above. Precision shows a number of True-Positive items compared to all items recommended by the system. It can also be called an acceptance rate. Recall compares True-Positive items to all items used by the user. Precision and Recall could be combined into one metric that compares precision-recall curves to, for example, provide the best recommendation list size.

**Ranking measures** are used to ensure the best order of items in a list by comparing with a “true list” (reference ranking) or by considering utility of the items (utility-based ranking).
2.3.2 Coverage

Under coverage three different properties could be evaluated [8]:

**Item Space Coverage** - describes a proportion of items that could be recommended to a user. Sales diversity is arguably the most popular measure showing how unfairly distinctive the chosen items are.

**User Space Coverage** - proportion of user iterations with the system that can be handled by a recommendation algorithm.

**Cold start** - performance of the system for new users/items/data types. Especially sensible for collaborative-filtering systems, where enough users related information has to be collected prior to suggestions generation.

2.3.3 Confidence

Confidence can be defined by how certain the system is about its recommendations [8, 16]. This property is applicable to collaborative-filtering recommendation systems and can be shown to a user, when, for example, recommending a movie, to help her to make a decision. Arguably, the most common metric for measuring confidence is the probability that the predicted value is true or percentage of rating that lie in a selected interval. When the evaluation is performed, the algorithm chosen should be as close as possible to the selected interval. Besides that confidence could also be used, for example, for automatically assigning tags to text when the confidence value is above a specified threshold.

2.3.4 Novelty

Novel recommendations are recommendations for items that the user did not know about [8]. It can be easily measured in a user study, but some valuable information could be gained even in an offline experiment.

To measure novelty in the offline experiment, it is possible to divide the data set in two based on time and train the system on the first set. The application should increase the novelty score when a recommendation has not been seen prior to the
selected time and decrease the score when the suggestion was already presented in the first set.

### 2.3.5 Trust, Serendipity and Diversity

Trust is a property that shows how confident a user is in the recommendations provided by the application. As trust relies on personal opinion, it can only be measured in a user study.

Serendipity is a notion of how surprising to the user the successful recommendations are [8]. Usually, this property is evaluated in a user study, but if it is possible to create a distance measure that could distinguish two items (for example, actors or authors), than it could be evaluated in another experiment types.

Diversity is a measure that shows how disconnected the suggested items are [8, 17]. It could be evaluated by introducing a distance measure in an offline experiment.

### 2.3.6 Utility, Risk and Robustness

Utility is a property used for e-commerce web sites. It can easily be measured in an online evaluation comparing profits that the web site gained before and after the recommendation algorithm was deployed or updated.

Risk is an extension of utility and used when the recommendation carries a possible risk, for example, in predicting stock market dynamics. This is evaluated by adding utility variance.

Robustness is an ability of a recommender system to tolerate fake information. There are no recommendation systems that could avoid the effect of the fake data completely, but it is possible to compare the level of influence of such data. Can be evaluated by injecting fake information into the system.

### 2.3.7 Privacy, Adaptivity and Scalability

Privacy cannot be evaluated, but it can be analyzed. User trusts the application with his private information to get the best recommendation results possible. For
example, with collaborative-filtering approach items selected by other users could expose sensitive information.

Adaptivity is a property that emphasizes on how the system could adjust to rapid changes of data sources or of the information focus. This can be analyzed in an offline experiment measuring the amount of information about a user/item necessary to make recommendations.

Scalability is a notion of how the recommendation system adjusts to large data sets. This property is very sensitive for collaborative-filtering systems. Scalability is usually measured with a data set that dramatically changes in size.

### 2.4 Drawing conclusions

When comparing results of two recommendation systems’ evaluations, it is necessary to be sure that the evaluation results are correct and are not a result of luck. P-value is the most popular tool for significance testing and is an essential part of various statistical methods. P-value shows a probability that a coincidence was the reason for the obtained results [18].

The simplest way to compute p-value is by performing a sign test [8, 18]. It computes the number of assessments where algorithm A outperforms \( n_A \) and underperforms \( n_B \) algorithm B leaving the equal values out of the consideration. Then based on these values two hypotheses are formed: one that states that algorithm A is better than algorithm B and the other that it does not. After that the significance level is calculated (2.2):

\[
p = (0.5)^n \sum_{i=n_A}^{n} \frac{n!}{i!(n-i)!},
\]

where \( n = n_A + n_B \). In the end, \( p \) is compared with an expected significance level. There are other tests for p-value calculation, but they make stronger assumptions on the value distribution in the set.

Size of the training and evaluation data sets linearly affect execution time of the quality assessment. A smaller data set size will result in a shorter execution time and the opposite holds for larger data sets. Aiming for the faster execution, it is still necessary to end up with reliable results. Standard deviation measurement
can help in this case. It guarantees that more than 99 percent of item would be in \((\pm 3 \times \text{StandardDeviation})\) interval. To compute the deviation a mean should be calculated (2.3):

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i,
\]

(2.3)

where \(n\) is the number of experiments and \(x_i\) is the result of \(i\)-th experiment. When the mean is available, it is easy to determine the standard deviation (2.4):

\[
\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2},
\]

(2.4)

2.5 Description of the algorithm used for topic suggestions

The recommendation algorithm in the consideration is used for generating topic suggestions for posts on Salesforce Chatter. A topic is a word or a set of words that best describe the content of a feed post. An example of a text and a recommendation window could be found on Figure 2.1.

![Figure 2.1: An example of topic suggestions](image)

The algorithm represents a content-based recommendation system. When a text is entered, it is analyzed with various text-mining libraries. The libraries extract terms that are looked up in an index of previously used topics. Topic index is unique for an organization. The organization encapsulates a collective goal for using the system. Each application might contain multiple organizations. If a
term matches with a topic in the index, the score of the term is summed with
the score of the topic in the index. Then weights are applied to the terms (for
example, an abbreviation has a higher weight than an adjective). All weights
are manually assigned from results of the online evaluation. After applying the
weights the items are sorted and the first three are presented to the user as topic
suggestions and the user can select only one item from the list. It is not possible
to assign more than 10 topics to one post and the topic name cannot be longer
than 99 characters.

2.6 Applicable properties

The properties that might be interesting for the algorithm evaluation are:

**Accuracy** - it was used by the company before and a combination of precision
and recall metrics could help in understanding how long the recommendation
list should be;

**Cold start** for the whole application - interesting from a perspective where the
algorithm cannot rely on the topic index and how it affects the accuracy
metrics;

**Diversity** - the algorithm does not have any notion of similarity and, thus, gen-
erates many similar topic suggestions. The way to avoid it is to create a
distance measure between two recommendations (matching parts of the sug-
gestions, synonyms, etc.) and evaluate the algorithm based on that.
Chapter 3

Research methods and methodologies

Methods and methodologies are essential parts of any research [19]. They provide all guidance necessary for a researcher to successfully complete his or her investigation. Anne Håkansson published a brief process necessary for figuring out the best suitable research methods and methodologies [20]. The paper presents a portal that could be used as a guidance tool for researchers (Figure 3.1).

The portal starts with basic research methods on top distinguishing between numerical and non-numerical research. It is followed by Philosophical Assumptions

Figure 3.1: The portal of research methods and methodologies [20]
that specifies the project’s point of observation. Research methods come after providing procedures for accomplishing the project’s tasks. It is succeeded by Research Strategy/Design, which narrows down organizational aspects of the research. Data Collection and Data Analysis follow it discussing how the data should be collected and analyzed based on the previous phases of the portal. Quality Assurance deals with validation and verification of the results with respect to validity, reliability, replicability and ethics. The portal ends with a Presentation phase, where results of the investigation are demonstrated and discussed.

3.1 Basic research methods

Quantitative and Qualitative methods represent two basic research method groups on top of the portal. The first performs experiments by measuring variables to accept or reject a hypothesis, while the second relies on understanding of behaviors and opinions. Both these research methods could be combined and used together one after another as a Triangulation method.

This work uses Quantitative research method group as it relies on measuring variables, which perfectly fits into a need of comparing and improving recommendation algorithms.

3.2 Philosophical assumptions

Philosophical assumption represents a starting point of the research and sets the projects’ perspective. Three of them are used in the portal:

- Posivism assumes that reality does not depend on tools or observers. A researcher prepares a hypothesis and performs tests to verify it;

- Realism also assumes the idea that reality is independent of the observer. The main difference from positivism is that the researcher observes the reality to find new facts instead of drawing a hypothesis in the beginning;

- Interpretivism relies on the idea that reality depends on observers’ opinion and should take it into account;
• Criticalism assumes that reality is a result of people activities.

Positivism perfectly fits into the needs of this project.

### 3.3 Research methods

Research methods represent procedures for reaching the investigation goals. Some of the most popular methods described in the portal are:

• Experimental Research where researcher tunes the variables to figure out the relations between them;

• With Descriptive Research the situation of phenomena is described without focus on causes or consequences of the situation;

• In Analytical Research the researcher operates with already existing data and knowledge to prove a hypothesis;

• Fundamental research is used to explore the fundamental concepts and to draw new theories about the laws of nature;

• Applied research method is used to explore specific problems while taking into account a limited number of circumstances;

• Conceptual research is used to produce new or to change developed concepts based on already existing knowledge;

• Empirical research is based on observation of real situations and experiences to state theories instead of making artificial pure experiments.

This work utilizes the Experimental Research method, as it is necessary to collect experimental data for testing hypotheses.

### 3.4 Research approaches

Research approaches are used for drawing conclusions. There are three approaches in the portal: Deductive, Inductive and Abductive. With Deductive approach the
researcher is supposed to state a hypothesis before performing the research. During the research a large amount of data is collected and hypothesis is verified based on quantitative research method. In opposite to Deductive, Inductive research method is used to state a hypothesis as a result of research. The amount of data should be small but sufficient for analysis of relations in phenomena. Abductive method combines the features of both deductive and inductive methods. It allows stating several hypotheses and selecting the best one as a result of the research.

This thesis uses Deductive approach due to the large amounts of data that should be collected and processed.

### 3.5 Research Strategies

Research strategies are guidelines for performing research. Here are some of the most popular ones: Experimental research, Ex post facto research, Surveys, Case study, Action research, etc. This work takes advantage of Experimental research due to its ability to provide cause-and-effect relationships between variables.

### 3.6 Data collection and data analysis methods

Data collection methods represent different ways to get data for research. Some of the methods in the portal are:

- The Experiment method accumulates data through experience;
- The Questionnaire method stands for collecting data in a form of opinions;
- Case study method is used to collect detailed data from a small number of participants;
- In the Observation method data is accumulated as a result of watching an event or an participants’ behavior;
- Language and Text method is used to collect data from documents studying the style of conversations.
Different methods can support the analysis of the gathered data: Statistics, Computational Mathematics (modeling, simulation), Coding (translation of interview into quantitative data), Analytic Induction and Grounded Theory (iterative methods, that alternate between collections and analyses), Narrative Analysis (concerns literary discussion and analysis).

This thesis uses the Experimental data collection method together with the Statistics as data analysis method for drawing reliable conclusions while comparing quality of recommendation algorithms.

### 3.7 Quality Assurance

Quality assurance for the quantitative research relies on validity, reliability, replicability and ethics. Validity guarantees that the instruments used during the investigation measure what was expected, reliability represents stability of measurements, replicability guarantees that another researcher can achieve the same results, while ethics questions moral principles of the research. The main ethics concern for the project is use of personal data. The work does not utilize, collect or expose personal information of any kind.
Chapter 4

Data sets

The two most difficult elements while evaluating a recommendation algorithm are: finding the correct data sets and modeling user behavior [8]. This chapter focuses on the data sets’ aspects of the investigation.

The most suitable data set should reflect end-user content expectations; it should also be as unbiased as possible by any recommendation algorithm or by theme of the text. This thesis investigates one recommendation algorithm and the rest of this chapter is focused on that algorithm.

4.1 Expected data

The recommendation algorithm gives topic suggestions for feed posts based on the content of the posts. A post consists of a text (varies in length, but is usually not longer than a couple of sentences), a user that created it, a timestamp and a set of associated topics and an invisible post identification number. An example of a feed post is presented on Figure 4.1. There are three possible sources of data for

![Figure 4.1: Example of a feed post](image)
the evaluation:

- Data available from internal deployments. This data is collected from two systems: the first one handles development process of all Salesforce products (tasks, bugs, work related questions and issues), while the second focuses on business aspects of the company. The data is highly biased by the current recommendation algorithm;

- Data from a short survey performed by one of the employees;

- Data available on the Internet.

The first two data sets do not satisfy the desired properties: the first is highly biased by the current recommendation algorithm as well as is highly specific and the second is too small (around 30 data items). This leaves data sets on the Internet for a consideration.

### 4.2 IEEE Xplore

IEEE Xplore is an online digital library sponsored by IEEE (Institute of Electrical and Electronics Engineers) [21]. The library contains all publications from different IEEE conferences. Access to the papers and articles is subscription-based, but abstracts and some additional information are available freely on the IEEE Xplore web site.

The average post length in Chatter is about the same size as a paper’s abstract and the author keywords are very close to a set of topics associated with a post. Even though content on IEEE Xplore is highly biased by scientific language and engineering aspects, it still makes one of the best possible data sources on the open Internet for the investigation.

IEEE Xplore lacks any kind of application programming interface (API) for accessing its data. This leaves one possible way for retrieving the data - parsing each page with a script.
4.2.1 IEEE Xplore structure

Each paper is associated with a separate web page that is identified by a unique number, and the number is a part of the page URL. A URL example is presented on Figure 4.2. 6678062 is a unique identifier. The logic behind how this number is generated is unknown, making it impossible to go through the data more efficient than incrementing the number by one. As a pay back for this approach, hitting a non-existing page is very common for the script (Figure 4.3).

Figure 4.2: Example of an IEEE Xplore URL

4.2.2 Parsing an article page

The script for retrieving data from an IEEE Xplore article page is written in Python. The script takes two integer values as an input for defining a range of unique article numbers for investigation. The logic behind the script does the following:
1. Article page URL is constructed and content of the page is requested (example of a page in the response presented on Figure 4.4);

![Article page example](image)

**Figure 4.4**: Article page example

2. Abstract retrieved for a HTML element with id = “article-page”;

3. Author keywords collected, all other types of keywords are ignored as only author keywords were manually assigned;

4. Date of conference is parsed (there are several date formats that are supported by the script: Day Month Year, Day-Day Month Year, Day Short-MonthName Year, Day-Day ShortMonthName, Year);

5. Collected data is serialized into a file as a JSON string.

The script is allowed to retrieve a new page only once a second to prevent IEEE Xplore from thinking that it performs a Denial of Service attack. Listing of the script is presented in Appendix A. Around 260 000 items were collected for future
experiments. None of those items contained any kind of personal information. The data would be fed to the topic suggestion algorithm and the list of author assigned keywords for each paper abstract would be used as a validation set.

4.3 Quora

Quora is a question-and-answer website, where questions are grouped into topics in a form of communities [22]. Content size on Quora varies from one sentence to a page of text. Each post may contain keywords that could be interpreted as topics from Salesforce.com point of view. Data on Quora covers a wide range of fields: from Computer Science to business, from baking to travelling.

Quora does not provide any kind of API for accessing its data. Even more, an authorization is required to access the most part of it, and the mechanism is not available for public use. Besides that, automated data retrieval becomes more complicated as there is no way to simply iterate over a unique identification number to be able to go through posts as all of them contain only a topic name and a post name in the URL.

But there is a way to collect enough data for the investigation without digging into the authorization protocol. Each Quora topic page (Figure 4.5) contains a set of previews of posts as well as a list of related topics.

The idea is to create a script that accumulates all the available topic pages and than retrieve all posts on those pages. Both scripts are written in Python.

4.3.1 Quora scripts

First script is responsible for collecting URLs to topic pages (Appendix B). It starts with a set of predefined topic URLs and executes until there are no pages it has not checked. On each page it parses “Related Topics” block by collecting URLs to the topic pages it has not worked on before. The script is allowed to retrieve a new page only once in a second to prevent Quora from thinking that it performs a Denial of Service attack. Around 41 000 topic pages were collected.

The second script takes care of data retrieval from posts. It takes a file with topic URLs as an input, collects posts’ URLs and parses data on those URLs. Each
Figure 4.5: Quora page for Computer Science topic

post (Figure 4.6) contains a header and a text. They are concatenated together and considered as a post body. Keywords are also preserved. Besides that a date of publication is also stored (it is not available on the post page, but is presented on the topic page).

The script (Appendix C) is allowed to retrieve a new page only once in a second to prevent Quora from thinking that it performs a Denial of Service attack. Around 175000 items were collected for future use. None of those items contained any kind of personal information. The data would be fed to the topic suggestion algorithm and the list of Quora keywords for each post would be used as a validation set.
Figure 4.6: Quora post in Computer Science topic
Chapter 5

Evaluation tool design

This chapter presents the design of the evaluation tool, how the tool is integrated into the current Salesforce.com test environment and shows examples of how an assessment of accuracy and diversity can be implemented.

5.1 Tool design

The core component of the tool is an evaluation framework. The framework design is inspired by MapReduce paper from Google [23]. Each evaluation consists of a number of independent experiments. Each experiment goes through the following five phases (Figure 5.1):

1. Set up phase. In this phase all the data that could be used during the entire experiment is generated. For example, to make experiments independent for the topic suggestion algorithm, it is necessary to create a new organization for each experiment, as topic index is unique per organization;

2. The Training phase covers the case when an algorithm should be taught based on a precollected data set before an evaluation could be performed. The phase is mostly applicable for evaluation of collaborative-filtering recommender systems;

3. The Evaluation phase is used for computing a selected metric using a data set that is different from the training set;
4. The Save results phase should preserve the metrics’ values computed in the evaluation phase to a storage space. The phase is also responsible for a correct data formatting. Normally, one experiment would produce one line of the output data;

5. The Clean up phase manages wiping the intermediate data from the system (topic index data, for example).

All phases described above are executed sequentially one after another and are together called an experiment workflow. The workflow is encapsulated in EvaluationWorker.java abstract class (Figure 5.2) and is marked final to prevent any future alteration. Each phase of the experiment workflow is presented as a separate abstract method that must be implemented based on needs of the selected algorithm and metrics. The Set up phase generates a SupportData object, which is passed to all other phases. The object consists of data that could be reused by different parts of the experiment workflow.

The Salesforce.com test environment takes care of running the evaluation. Each evaluation is a test located in a test class.
Any developer can run an evaluation for any arbitrary algorithm by implementing necessary parts of the abstract class and simulating a desired user behavior.

The rest of the chapter serves as an example on how an evaluation of the topic suggestion algorithm can be performed with the tool. It covers simulating expected user behavior as well as implementing accuracy and diversity metrics.

### 5.2 User behavior

User behavior for the evaluation of the topic suggestion algorithm is straightforward:

1. Create a feed post;
2. Assign a set of topics to the post (topics from older posts can also be used);
3. Repeat 1 and 2 until out of posts. Time flow should also be respected during the evaluation and the data sets should be sorted accordingly.

### 5.3 Accuracy evaluation

The accuracy evaluation of Topic Suggestion algorithm consists of two steps:
1. Generation of a training and evaluation data sets;

2. Running the evaluation test.

5.3.1 Data set parameters

The first step is necessary to preserve the training and evaluation data for comparison between different algorithms, as all of them should run on the same sets. For each experiment one data set file is generated (items are randomly selected and sorted). Each evaluation consists of 20 experiments to use sign test for drawing reliable results.

Training and evaluation data set sizes are equal and were selected experimentally based on standard deviation of the evaluation results. Bigger size requires longer execution time for a test. Additional 500 elements are punished by 10 minutes for each experiment. The experiment was performed on IEEE Xplore data set with sizes in range of 4000 and 6500 items and step of 500 items (Figure 5.3). With desired standard deviation equals to 0.3%, the size of 5000 items for the training and evaluation sets was found to be the optimal for both the precision and recall metrics used in accuracy evaluation.

![Figure 5.3: Standard deviation for precision over different data set sizes](image-url)
5.3.2 Accuracy evaluation test

The accuracy evaluation test (Appendix D) spans 20 tasks (one task per experiment) that are consumed by three threads. Three threads is an optimal value for a typical Salesforce.com developer machine. Each task takes the size of the data sets, input and output file paths together with some additional parameters as arguments. When a thread consumes a task, it follows the experiment workflow:

1. In the Set up phase the organization identifier is created together with user information. This is necessary for creating a feed post. It is not possible to save a topic to Topic Index without a feed post associated with the topic. Also, data set provider is initialized and the training/evaluation data is loaded into memory;

2. In the Training phase 5000 feed posts together with assigned topics are added to the system;

3. The Evaluation phase is used for computing the precision and recall metrics by comparing suggested topics with the expected ones from the evaluation data set;

4. The Save results phase is responsible for saving the output data in the correct format (Figure 5.4). Every output line consists of (from left to right):

   (a) Id. It is a unique experiment identifier used for sorting the results when comparing two algorithms with a sign test. Starts with 0;

   (b) Precision score. Represents a precision metric result with 1 as maximum value;

   (c) Recall score. Represents a recall metric result with 1 as maximum value;

   (d) Data set size. Shows the evaluation and training data set sizes;

   (e) Time taken for the experiment to be completed.

   |   | 0.1167 | 0.0715 | 5000 | 00:44:35 |
   | 0 | 0.1187 | 0.0726 | 5000 | 00:44:59 |
   | 2 | 0.1232 | 0.0757 | 5000 | 00:49:56 |
   | 4 | 0.1122 | 0.0692 | 5000 | 00:46:33 |
   | 3 | 0.1125 | 0.0695 | 5000 | 00:47:14 |

Figure 5.4: Example of a correct output lines for accuracy property
5. In the Clean up phase all posts and topics are removed from the system.

To verify that the evaluation tool works, the topic suggestion algorithm was tested with the topic index on and off (which could be considered as a cold start case). The expectation was that the algorithm with topic index on would outperform the one with no index in accuracy metrics (Figures 5.5(a), 5.5(b), 5.5(c), 5.5(d)).

\[ \text{Figure 5.5: Accuracy scores with and without topic index} \]

It is visible on the graphs that the algorithm with the topic index is better than the one without.

5.4 Diversity evaluation

The process of diversity evaluation is similar to accuracy with two exceptions: the implementation of the evaluation phase in the experiment workflow and the output data format (Appendix E). The phase computes a diversity score, which
represents the number of similar topic suggestions. Two suggestions are similar if one of the words in the first suggestion exists in the second. Stop words [24] and words with a length that is less than 3 characters are ignored. If two topics are considered similar, the one with lower score is removed.

The outcome of the diversity evaluation is presented in a text file, where each line corresponds to one experiment (Figure 5.6).

All output elements are the same as for accuracy evaluation with two alterations: the second element from the left represents the diversity score, while the third is 1/(diversity score) and might be used for significance testing while comparing two algorithms based on diversity property (the value will be equal to one in case of zero diversity score).
Chapter 6

Topic suggestion algorithm evaluation and recommended improvements

The topic suggestion algorithm is evaluated in this chapter aiming to prepare a set of improvements that can increase the quality of generated recommendations. The evaluation is performed based on two properties: accuracy and diversity. The diversity is assessed by a distance measure that is calculated for each two generated recommendations, while accuracy is graded based on two metrics:

1. Precision (acceptance rate) - number of accepted suggestions divided by total number of recommendations;

2. The recall compares accepted items from the recommended list to all items used by the user.

6.1 Accuracy evaluation

In the accuracy evaluation section several parameters and weights were studied: number of generated suggestions, threshold, adjective and topic index weights.
6.1.1 Number of generated suggestion

The current topic suggestion algorithm generates three recommendations for each run. This value was selected based on an assumption that a list of three items looks appealing to the customer in the user interface. It is well known that higher recall results in lower precision as well as the opposite [13, 25, 26]. Thus, an experiment exercising different number of topic recommendations (between 1 and 5) was performed.

Figures 6.1(a), 6.1(b), 6.1(c), 6.1(d) present how precision and recall change with different number of generated topic suggestions for the IEEE Xplore and Quora data sets. Horizontal axes reflect median values.

**Figure 6.1:** Effect of different number of suggestions on accuracy metrics

The correlation between precision and recall for both data sets based on median values is illustrated on Figure 6.2. The figure shows that the precision decreases when the recall grows. This is statistically significant for most of the test cases.
and the desired p-value is 0.01 (with an exception of precision for Quora data set between 4 and 5 recommendation items). The target user can benefit from switching to 2 topic suggestions for a feed post, which would increase precision. This is considered to be the most valuable metric at Salesforce.com.

![Figure 6.2: Precision/Recall for IEEE and Quora data sets](image)

6.1.2 Topic threshold

Before topic suggestions are presented to the user, they are filtered. It happens by comparing their scores to a threshold parameter. This section looks into how the parameter could be larger in order to achieve higher suggestions quality. An experiment exercising different threshold values was designed. The values were in a range between 10 and 16 (current value is 12.5) with a step of 0.5. The experiment was performed on Quora and IEEE Xplore data sets aiming to find a better threshold weight that either improves both the precision and recall metrics or increases one of them without impacting the other.

Figure 6.3(b) plots median values for all results on the Quora data set. The recall constantly decreases with the threshold value being increased. The change is statistically significant (desired p-value equals 0.01), proving value 10 to achieve the highest recall score. Sign test fails to justify one of the threshold values being better than the other for precision in a range between 10 and 12. If the threshold is higher than 12, it gets a lower precision score and is overperformed by smaller
values. The algorithm might benefit from switching the parameter to a lower value than the current one both in precision and recall. Tests suggest it to be equal to 10, because any considered value that is below 12.5 performs the same from precision point of view and 10 outperforms all the others for the recall metric.

![Median results](image)

(a) Median precision and recall on IEEE data  
(b) Median precision and recall on Quora data

**Figure 6.3:** Effect of Topic threshold parameter on accuracy metrics

Evaluation results of the topic threshold parameter on the IEEE Xplore data set do not resemble the same outcome: there is no statistically significant difference between any of the parameter values that were investigated (Figure 6.3(a)). It might be due a fact that average post length on Quora is shorter that an abstract on IEEE Xplore and the libraries used for terms extraction are less confident with assigning scores.

The overall recommendation is to try to lower the parameter as the average Salesforce Chatter post is shorter than an abstract of a scientific paper and to observe the results in an online experiment.

### 6.1.3 Topic Index

The topic index parameter is responsible for specifying how likely an entity to be a good topic suggestion if a user already utilized it for previous feed posts. Evaluation of different topic index weights (9, 10, 11, 12, 13, 14, 15) did not show any improvements compared to the already selected one (12) in any of the accuracy metric for the IEEE Xplore data (Figure 6.4(a)). Experiments on the Quora set showed different results: they suggested that the parameter should be increased to 15 and this would result in improvement for all accuracy metrics.
Increasing the parameter should positively affect the quality of the algorithm, because the Quora data set better represents the target content of the system.

![images](Figure 6.4(b)).

6.1.4 Adjective parameter

Adjective parameter is one of many annotation type weights used by the topic suggestion algorithm. It represents how important adjectives are for the topic names generation.

To understand how the parameter affects accuracy metrics an experiment was performed: two values lower and higher than the currently used were selected (0.1, 0.4; 1.0, 1.3; 0.7 - current) and accuracy was measured on Quora data set. Recall grows with the adjective parameter, but precision is statistically highest with desired p-value equals 0.01, when the parameter equals 1.0 (Figures 6.5(a), 6.5(b)). When the experiment was performed on IEEE Xplore data set, 0.7 is the lowest optimal value with desired significance level. With a higher value for the adjective parameter (1.0), probability that the result is gained by luck started to drop, but did not reach the desired p-value. Making the adjective weight higher might improve accuracy, but it should be tested in production.

![images](a) Median precision and recall on IEEE data  (b) Median precision and recall on Quora data

**Figure 6.4:** Effect of Topic Index parameter on accuracy metrics
6.2 Diversity metric

Diversity shows how unsimilar recommendations are. The simplest way to understand how close in meaning two topic suggestions are to introduce a basic distance measure: two topics are similar if at least one word in a topic name exists in another topic name. A metric built on this principle gives around 2000 matches for a 5000 evaluation set size. Barry Smyth, Maurice Coyle and Peter Briggs state that higher diversity leads to a faster search [8]. This can also suggest the following hypothesis: better diversity will result in a better accuracy. To verify the hypothesis the topic suggestion algorithm was altered to filter out similar recommendations based on the distance measure described above and accuracy metrics were compared.

Figures 6.6(a), 6.6(b), 6.6(c), 6.6(d) illustrate the outcome of the experiment. The currently used solution (Current line) outperforms the solution with a better diversity in almost every experiment (Better diversity line). To understand the reasons of it the topic suggestion algorithm was altered in a different way: similarities were filtered out after three recommendations are generated (Better local diversity line). With this new approach the precision metric reports statistically significant improvements of the algorithm (with a respect to desired p-value equals 0.01), while the recall metric scores shrink. It happens due to how accuracy measures are calculated. Assertion of the recommendations is done based on string equality, but there are plenty of cases when expected result only partly matches the actual outcome. For example, there are two topic suggestions: “US Supreme
Figure 6.6: Accuracy metric scores for the algorithm with different diversities

Court” and “Supreme Court”. They are similar based on the used distance measure and the one with a lower score would be filtered out. If an expected topic was “Supreme Court”, the recommended “US Supreme Court” would be found irrelevant. The situation described is very common and solving it might be a core element in future work.
Chapter 7

Conclusions and future work

7.1 Conclusions

We have built a tool for evaluation of recommendation systems. The tool is designed to provide flexibility by adopting different evaluation metrics, various user behaviors and accommodating any type of recommendation system. Due to its simplicity, the tool does not bind a user only to the accuracy evaluation property, but allows him to measure any property he wants.

The tool is an evaluation framework that is built on top of Salesforce.com test environment inheriting parts of user interface and other capabilities. The framework pictures the quality evaluation as a set of experiments. Each experiment consists of the following developer-defined stages: setup, training, evaluation, save results and clean up. Inspiration for such a division comes from the MapReduce framework introduced by Google, which is widely used due to its flexibility and simplicity.

The work also investigates quality of a recommendation algorithm for topic suggestions used by Salesforce.com. It experiments with several different weights showing how accuracy of the algorithm could be improved just by triggering them. All the experiments were performed on big data sets collected from IEEE Xplore and Quora. Besides accuracy, the diversity property was examined by introducing a simple distance measure for reflecting similarity between suggestions. Adjusting the algorithm for higher diversity showed how accuracy scores might be affected.
by providing less similar recommendations in a set. These evaluations narrowed down a list of possible directions for future work.

7.2 Future work

There are several directions for the future work. First, when omitting similar recommendation and comparing the outcome with the expected set, it is impossible to automatically see similarities between an expected topic suggestion and a generated one. It might be useful to add either an additional set of similar expected results or by employing an external library or service for finding analogies. For topic suggestion algorithm it might be an online service for finding synonyms - thesaurus.com [27].

Second, we investigated less than half of the weights used for topic recommendations and we did not look into any other properties besides accuracy and diversity. It might be useful to run experiments with other weights and to employ other evaluation properties, like novelty, serendipity, adaptivity and ranking measures for accuracy.

Third, some weights might be changed automatically during the evaluation, which is very useful for selecting the right value before production deployment and online assessment. Unfortunately, most of the weighs cannot be changed in that way and the application has to be restarted every time a user wants to change a value. We believe that it could be done automatically by adding new features to the test environment, which would allow updating weights in configuration files and restarting the application without any human attention.

Forth, we used two data sets for the evaluation. The one collected from Quora best reflects the expected content of the system and contains about 175 000 items. Gathering more items is much harder than collecting the existing set: it requires authorization on Quora web site. The process is not documented and there are no APIs provided for performing authorization and accessing the data. It makes the process of protocol reverse engineering very complicated and time consuming, but the data is extremely valuable. Besides that, there might be commercially available data sets that even better represent the expected content of the system.
Fifth, some algorithms rely heavily on time while generating recommendations. For these algorithms it might be useful to have a module that simulates different time behaviors. For example, assume an algorithm that uses two weights for topic index. The first weight is for topic items assigned no later than half a year ago and the second is for those that were assigned earlier. If the difference between those weights is significant for the algorithm, it is necessary to simulate a user behavior that takes it into account. The module might also contain a predefined set of behaviors: sequential with a fixed time step, sequential with a random step, exponential, etc. It may also have some templates for simulating concurrent behavior. This will reflect how multiple users working with the same system and how it affects realtime recommendations.
Appendix A

Script for retrieving data from IEEE Xplore

To run the script execute in terminal:

```bash
python script_name left_border right_border destination_file
```

Script:

```python
import datetime
import time
import os
import sys
import getopt
import urllib2
from BeautifulSoup import BeautifulSoup
from dateutil import parser
import json

__author__ = 'dkazarin'

def unix_time (dt):
    epoch = datetime.datetime.utcfromtimestamp(0)
    delta = dt - epoch
    return delta.total_seconds()

def unix_time_millis (dt):
    return unix_time (dt) * 1000.0

def extract_keywords(soup):
    keywords_list = soup.find("div", {"id": "abstractKeywords"}) .
        find("p", text="AUTHOR KEYWORDS").parent.parent.findChildren('ul')
    keywords = []
```
for keyword in keywords_list:
    keywords.append(keyword.li.a.text)
return keywords

def extract_abstract(soup):
    abstract = soup.find("div", {"id": "article-page"}).find("div", {"class": "article"}).p.text
return abstract

def parse_author_line_with_multiple_names(author_line):
    authors = author_line.split(';')
    return [x.strip() for x in authors]

def extract_authors(soup):
    authors_list = soup.find("div", {"id": "abstractAuthors"}).find("div", {"class": "art-authors"}).findChildren('a')
    authors = []
    for author in authors_list:
        author_line = author.text
        if ';' not in author_line:
            authors.append(author_line)
        else:
            authors.append(parse_author_line_with_multiple_names(author_line))
    return authors

def extract_conference_date(page, soup):
    if "Date of Conference:" in page:
        date = soup.find("div", {"id": "article-page"})
        .find("div", {"class": "article-ftr"})
        .find("b", text="Date of Conference:").parent.parent.text[19:]
        if len(date) > 50:
            date = soup.find("div", {"id": "article-page"})
            .find("div", {"class": "article-ftr"})
            .find("b", text="Date of Conference:").parent.parent.contents[6]
        if '-' in date:
            date = date.split('-')[1]
        else:
            date = soup.find("div", {"id": "article-page"})
            .find("div", {"class": "article-ftr"})
            .find("div", {"class": "article-info cf"})
            .find("bt", text="Date of Publication:").parent.parent.dd.text
        if '-' in date:
            date = date.split('-')[1]
    date = date.replace("Jan.", "January")
    date = date.replace("Feb.", "February")
    date = date.replace("Mar.", "March")
    date = date.replace("Apr.", "April")
    date = date.replace("Jun.", "June")
date = date.replace("Jul.", "July")
date = date.replace("Aug.", "August")
date = date.replace("Sept.", "September")
date = date.replace("Oct.", "October")
date = date.replace("Nov.", "November")
date = date.replace("Dec.", "December")

return parser.parse(date)

def main():
    # parse command line options
    try:
        opts, args = getopt.getopt(sys.argv[1:], "h", ["help"])
    except getopt.error, msg:
        print msg
        print "for help use --help"
        sys.exit(2)
    # process options
    for o, a in opts:
        if o in ("-h", "--help"):
            print __doc__
            sys.exit(0)
    # process arguments
    left = int(args[0])
    right = int(args[1])

    # article_id = '6224392'
    number_of_articles = 0
    number_of_errors = 0

    # open file for writing
    if not os.path.isfile(args[2]):
        f = open(args[2], "w")
        f.write("{"dataset": ["n"
    else:
        f = open(args[2], "a")

    for article_id in range(left, right):
        print '--------\nWorking on {0} ({1} left)...'.format(str(article_id), \n            str(right - article_id - 1))
        time.sleep(1)
        url = 'http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=' + \n            str(article_id)
        try:
            response = urllib2.urlopen(url, timeout=5)

            if response.code != 200:
                continue
            page = response.read()
            if 'Page Not Found' in page or 'AUTHOR KEYWORDS' not in page:
                continue
            soup = BeautifulSoup(page)
# get abstract
abstract = extract_abstract(soup)

# get keywords
keywords = extract_keywords(soup)

# get authors
authors = extract_authors(soup)

# get dates
date = extract_conference_date(page, soup)

number_of_articles += 1

print 'id: {4}
keywords: {1}
authors: {2}
publication date: {3}'.format(abstract, keywords, authors, date, article_id)

data = {
'article_id': article_id, 'abstract': abstract, 'keywords': keywords, 'authors': authors, 'date': long(unix_time_millis(date))
}
serialized_article = json.dumps(data)

f.write(serialized_article + ',
')
f.flush()
soup.reset()
response.close()

except BaseException as e:
    print 'Article id= ' + str(article_id) + ' Error message = ' + e.message
    number_of_errors += 1

# f.write('}"

f.close()
Appendix B

Script for retrieving topic URLs from Quora

To run the script execute in terminal:

```python
python script_name
```

Script:

```python
from BeautifulSoup import BeautifulSoup
import httplib2
import time
__author__ = 'dkazarin'

def main():
    file_name = 'links
    base_url = 'http://www.quora.com'
    categories = [
        base_url + '/UI-UX-Design-Patterns', base_url +
        '/Computer-Science', base_url + '/Business',
        base_url + '/Medicine-and-Healthcare', base_url +
        '/Law', base_url + '/Product-Management',
        base_url + '/Sales-Management-1', base_url +
        '/Marketing', base_url + '/Leadership',
        base_url + '/Finance']
    checked_categories = []

    # copy all categories to a temp list
    categories_for_iteration = categories[:]

    while len(categories_for_iteration) != 0:
        for category in categories_for_iteration:
            if category in checked_categories:
                categories.remove(category)
```
continue

resp, content = httplib2.Http('.cache').request(category)
if resp.status != 200:
    continue

try:
    soup = BeautifulSoup(content)
    block = soup.find('div', attrs={'class': 'row section related_topics'})
    links = block.findAll('li', attrs={'class': 'related_topic_item'})
    for link in links:
        url = base_url + link.find('a')['href']
        if url not in checked_categories:
            categories.append(url)

    checked_categories.append(category)
    with open(file_name, 'a') as f1:
        f1.write(category + '\n')
except Exception:
    continue
    categories.remove(category)

time.sleep(1)

categories_for_iteration = categories[:]
print str(len(categories_for_iteration)) + ' items on the list'

if __name__ == '__main__':
    main()
Appendix C

Script for retrieving posts data from Quora

To run the script execute in terminal:

```
python script_name left_border right_border destimation_file
```

```python
from BeautifulSoup import BeautifulSoup
import datetime
import httplib2
from dateutil import parser
import json
import time
__author__ = 'dkazarin'

def unix_time(dt):
    epoch = datetime.datetime.utcfromtimestamp(0)
    delta = dt - epoch
    return delta.total_seconds()

def unix_time_millis(dt):
    return unix_time(dt) * 1000.0

def main():
    file_name = 'links'
    id_counter = 0

    with open(file_name, 'r') as f1:
        links = f1.readlines()
```
for link in links:
    if '\n' in link:
        link = link.replace('\n', '')

resp, content = httplib2.Http().request(link)
if resp.status != 200:
    continue

soup = BeautifulSoup(content)
posts = soup.findAll('a', attrs={'class': 'question_link'})
for post in posts:
    post_link = 'http://www.quora.com' + post['href']
    date = post.parent.parent.parent.parent.parent.parent.parent.find(
        attrs={'class': 'timestamp'})
    if date is None:
        date = post.parent.parent.parent.parent.parent.parent.parent.parent.find(
            attrs={'class': 'timestamp'})
        if date is None:
            continue
    try:
        date_value = parser.parse(date.text)
    except ValueError:
        continue
    resp, content = httplib2.Http().request(post_link)
    if resp.status != 200:
        continue

local_soap = BeautifulSoup(content)
topic_tags = local_soap.findAll('div', attrs={'class': 'topic_list_item'})
topics = []
is_skipping = False
for tag in topic_tags:
    topics.append(tag.text)
    if len(tag.text) >= 99:
        is_skipping = True
if len(topics) == 0 or len(topics) >= 10 or is_skipping:
    continue

post_text = local_soap.find('div', attrs={'class': 'question_text_icons'})
    .parent.text.replace('&nbsp;', '')
try:
    post_text = post_text + '\n' + local_soap.find('div', attrs={
        'class': 'question_details_text inline_editor_content'}).text
except AttributeError:
    continue

id_counter += 1

with open('data', 'a') as f1:
    new_data = {'abstract': post_text, 'keywords': topics,
'date': long(unix_time_millis(date_value)),
'article_id': id_counter
f1.write(json.dumps(new_data) + '\n')

time.sleep(1)

if __name__ == '__main__':
    main()
Appendix D

Accuracy evaluation

```java
public class PrecisionAndRecallEvaluationTest extends BaseEvaluationTest {
    private final String corePath;
    private final String home = "\home/dzakarin";

    // this is an optimal value for a machine with 12 core CPU.
    private final int numberOfThreads = 3;

    public PrecisionAndRecallEvaluationTest(String name) {
        super(name);

        corePath = Paths.get("").toAbsolutePath().getParent().toString();
    }

    /**
     * Applies n times the topic suggestion algorithm to a data set (line by line)
     * and compares the output with the expected values. Based on the comparison
     * computes precision and recall and saves the computed values in a file.
     *
     * @throws Exception
     */
    public void testPrecisionAndRecall() throws Exception {
        int dataSetSize = 5000;
        int sampleSize = 20;

        int numberOfSuggestions = 3;
        String outputFile = home + "\Downloads/Accuracy/Quora/Aj1.3.txt";

        // directory must exist before running this test
        String folderPath = corePath + "/sbi/test/func/java/src/sbi/quality/evaluation/data/ieeexplore/reuse/";
        String[] paths = generateNFilePaths(folderPath, 20);

        assertEquals(paths.length, sampleSize);

        ExecutorService executorService = Executors.newFixedThreadPool(numberOfThreads);
    }
}
```
// test runs sampleSize times for each data set size
runNExperiment(executorService, sampleSize, numberOfSuggestions,
dataSetSize, outputFile, paths);

executorService.shutdown();
extectorService.awaitTermination(BasicTestContext.getTotalTimeout(), TimeUnit.SECONDS);

} /* *
* Generates paths to files with numbered names in a folder
* @param folderPath path to the folder with files
* @param numberOfFiles number of file paths to generate */
public static String[] generateNFilePaths(String folderPath,
int numberOfFiles) {
String[] result = new String[numberOfFiles];
for (int i = 0; i < numberOfFiles; i++) {
result[i] = folderPath + String.valueOf(i);
}

return result;
}

private void runNExperiment(ExecutorService executorService,
int sampleSize,
int numberOfSuggestions, int currentDatasetSize, String outputFile,
String[] paths) throws IOException {
for (int currentSampleId = 0; currentSampleId < sampleSize; currentSampleId++) {
FileDataWriter dataWriter = new FileDataWriter(outputFile);
Runnable worker = new AccuracyEvaluationWorker(String.valueOf(currentSampleId),
FileInMemoryDataProvider.class, paths[currentSampleId], dataWriter,
currentDatasetSize, numberOfSuggestions);
executorService.execute(worker);
}
}

public class AccuracyEvaluationWorker extends EvaluationWorker {
private FeedItemTestingUtil feedItemTestingUtil;
private TopicSuggester topicSuggester;
private TopicSystem topicSystem;
private final int dataSetSize;
private final int numberOfSuggestions;
private final String id;
private DataProviderBase dataReader;
private final Class<? extends DataProviderBase> dataProvider;
private final String dataSource;
private final IDataWriter dataWriter;
private long truePositive;
private long falsePositive;
private long falseNegative;
protected static final Logger logger = SFDCLog.open(AccuracyEvaluationWorker.class);

/**
* Creates an instance of AccuracyEvaluationWorker with default
* MiscTopicSuggesterWeights, but the topicScoreThresholdParam
* @param id
* @param dataProvider
* class that represents a data provider
* @param dataSource
* path to data that is understandable by dataProvider
* @param dataWriter
* class that used as a data writer
* @param dataSetSize
* size of training and evaluation data sets
* @param topicScoreThresholdParam
* parameter value
*/
public AccuracyEvaluationWorker(String id, Class<? extends DataProviderBase> dataProvider,
        String dataSource, IDataWriter dataWriter, int dataSetSize,
        int numberOfSuggestions, float topicScoreThresholdParam) {
    this.dataSetSize = dataSetSize;
    this.numberOfSuggestions = numberOfSuggestions;
    this.dataProvider = dataProvider;
    this.dataSource = dataSource;
    this.dataWriter = dataWriter;
    this.id = id;

    this.topicSuggester = new TopicSuggester(Config.getAppConfig() .getField(MiscTopicSuggesterWeightsConfig.existingTopicsWeightParam),
            Config.getAppConfig().getField(MiscTopicSuggesterWeightsConfig.allUpperCaseTopicsWeightParam),
            Config.getAppConfig().getField(MiscTopicSuggesterWeightsConfig.topicIndexScaleFactorParam),
            topicScoreThresholdParam);
    this.topicSystem = ProviderFactory.get().get(TopicSystem.class);
}

/**
* Creates an instance of AccuracyEvaluationWorker with default MiscTopicSuggesterWeights,
* but the existingTopicWeights
* @param id
* @param dataProvider
* class that represents a data provider
* @param dataSource
* path to data that is understandable by dataProvider
* @param dataWriter
* class that used as a data writer
* @param dataSetSize
* size of training and evaluation data sets
* @param topicScoreThresholdParam
* parameter value
*/
public AccuracyEvaluationWorker(String id, Class<? extends DataProviderBase> dataProvider,
        String dataSource, IDataWriter dataWriter, int dataSetSize,
        float indexWeights, int numberOfSuggestions) {
    this.dataSetSize = dataSetSize;
    this.numberOfSuggestions = numberOfSuggestions;
    this.dataProvider = dataProvider;
    this.dataSource = dataSource;
    this.dataWriter = dataWriter;
    this.id = id;

    this.topicSuggester = new TopicSuggester(indexWeights,
        Config.getAppConfig().get(SectionEnum.MiscTopicSuggesterWeights)
        .getField(MiscTopicSuggesterWeightsConfig.allUpperCaseTopicsWeightParam),
        Config.getAppConfig().get(SectionEnum.MiscTopicSuggesterWeights)
        .getField(MiscTopicSuggesterWeightsConfig.topicIndexScaleFactorParam),
        Config.getAppConfig().get(SectionEnum.MiscTopicSuggesterWeights)
        .getField(MiscTopicSuggesterWeightsConfig.topicScoreThresholdParam));
    this.topicSystem = ProviderFactory.get().get(TopicSystem.class);
}

/**
 * Creates an instance of AccuracyEvaluationWorker with default MiscTopicSuggesterWeights
 *
 * @param id
 * @param dataProvider
class that represents a data provider
 * @param dataSource
path to data that is understandable by dataProvider
 * @param dataWriter
class that used as a data writer
 * @param dataSetSize
size of training and evaluation data sets
 * @param numberOfSuggestions
number of suggestions that (link TopicSuggester) will generate
*/

public AccuracyEvaluationWorker(String id, Class<? extends DataProviderBase> dataProvider,
        String dataSource, IDataWriter dataWriter, int dataSetSize,
        int numberOfSuggestions) {
    this(id, dataProvider, dataSource, dataWriter, dataSetSize, numberOfSuggestions,
        Config.getAppConfig().get(SectionEnum.MiscTopicSuggesterWeights)
        .getField(MiscTopicSuggesterWeightsConfig.topicScoreThresholdParam));
}

@Override
public SupportData setUpEvaluation() {
    // each run - create new org
    UserContext.getProvider().release();

    String orgName = UUID.randomUUID().toString();
    String orgId;
    try {
        orgId = establishNewOrg(orgName);
    }
feedItemTestingUtil = new FeedItemTestingUtil(new OrgTestingUtil(orgId));
)
catch (Exception ex) {
    throw new RuntimeException(ex.getMessage());
}

UserInfo user = UserContext.get().getUserInfo();
String parentId = user.getUserId();

try {
    this.dataReader = getDataProviderInstance(this.dataProvider, this.dataSource);
} catch (Exception x) {
    throw new RuntimeException(x);
}

QualityEvaliationUtil qualityEvaliationUtil = new QualityEvaliationUtil(dataReader, dataWriter);

return new AccuracySupportData(orgId, user, parentId, qualityEvaliationUtil);

@Override
public void train(SupportData supportData) {
    AccuracySupportData accuracySupportData = (AccuracySupportData)supportData;
    int numberOfItems = 0;
    while (numberOfItems < dataSetSize) {
        logProgress(numberOfItems, dataSetSize, "Training progress:");
        try {
            Quadruple<Integer, String, List<String>, Date> evaluationLine =
                getNextDataTuple(accuracySupportData.getQualityEvaliationUtil());
            String postId = addPost(evaluationLine.getSecond(),
                accuracySupportData.getUser(),
                accuracySupportData.getParentId());
            saveTopics(postId, evaluationLine.getThird(), evaluationLine.getFourth(),
                accuracySupportData.getOrgId());
        } catch (Exception ex) {
            logger.warning(ex.getMessage());
            // log the error and continue, there will be more lines
            continue;
        }
        numberOfItems++;
    }
}

@Override
public void evaluate(SupportData supportData) {
    AccuracySupportData accuracySupportData = (AccuracySupportData)supportData;
    int numberOfItems = 0;
    while (numberOfItems < dataSetSize) {
        logProgress(numberOfItems, dataSetSize, "Evaluation progress:");
try {
    Quadruple<Integer, String, List<String>, Date> evaluationLine = 
        getNextDataTuple(accuracySupportData.getQualityEvaluationUtil());

    String postId = addPost(evaluationLine.getSecond(),
        accuracySupportData.getUser(),
        accuracySupportData.getParentId());

    // make suggestions
    List<TopicSuggestion> suggestions = topicSuggester
        .suggestTopics(evaluationLine
            .getSecond(), numberOfSuggestions,
            Collections.<String>emptyList());

    Collection<String> matchedRecommendations =
        foundMatchedWithIgnoredCase(suggestions,
        evaluationLine.getThird());

    int recommendedAndUsed = saveTopics(postId, matchedRecommendations,
        evaluationLine.getFourth(),
        accuracySupportData.getOrgId());

    int recommendedAndNotUsed = suggestions.size() - recommendedAndUsed;
    int notRecommendedButUsed = evaluationLine.getThird().size() -
        recommendedAndUsed;
    truePositive += recommendedAndUsed;
    falsePositive += recommendedAndNotUsed;
    falseNegative += notRecommendedButUsed;
} catch (Exception ex) {
    logger.warning(ex.getMessage());
    // log the error and continue, there will be more lines
    continue;
}

numberOfItems++;
}

@Override
public void saveData(SupportData supportData) {
    AccuracySupportData accuracySupportData = (AccuracySupportData)supportData;
    // compute precision and recall
    double precision = (double)truePositive / (double)(truePositive + falsePositive);
    double recall = (double)truePositive / (double)(truePositive + falseNegative);
    double totalSeconds = getExecutionTime() / 1000;
    int hoursTaken = (int)(totalSeconds / 3600);
    int minutesTaken = (int)((totalSeconds - 3600 * hoursTaken) / 60);
    int secondsTaken = (int)(totalSeconds - 3600 * hoursTaken - 60 * minutesTaken);

    String[] result = new String[] { id, String.format("%.4f", precision),
        String.format("%.4f", recall),
        String.format("%d", dataSetSize),
        String.format("%02d:%02d:%02d", hoursTaken,
minutesTaken, secondsTaken);
}

String resultLine = prepareResults(result);
try {
    // save the result in a file
    (accuracySupportData.getQualityEvaluationUtil()).writeResultLine(resultLine);
} catch (Exception ex) {
    throw new RuntimeException(ex.getMessage());
}

@Override
public void clean(SupportData supportData) {
    try {
        feedItemTestingUtil.deleteAllRecords();
        CTopicIndexTestUtils.clearTopicIndex();
    } catch (Exception ex) {
        logger.warning(ex.getMessage());
    }

    System.gc();
}

/**<p>
 * Compares two sets and returns a set of items found in both sets
 * <p>
 * @param suggestions
 * @param expectations
 * @return
 * */
protected Collection<String> foundMatchedWithIgnoredCase(Collection<TopicSuggestion> suggestions,
    Collection<String> expectations) {
    ArrayList<String> matchedRecommendations = new ArrayList<String>();
    // compare the suggestions with expectations
    for (TopicSuggestion suggestion : suggestions) {
        for (String expectedRecommendation : expectations) {
            if (expectedRecommendation.equalsIgnoreCase(suggestion.getName())) {
                matchedRecommendations.add(expectedRecommendation);
            }
        }
    }

    return matchedRecommendations;
}

/**<p>
 * Creates a data provider instance using reflection
 * */
private DataProviderBase getDataProviderInstance(Class<? extends DataProviderBase> type,
    String dataSource) throws Exception {
    return type.getConstructor(String.class).newInstance(dataSource);
}
protected int saveTopics(String postId, Collection<String> topicNames, Date postCreationDate, String orgId)
throws SQLException {
    ArrayList<String> networkIds = new ArrayList<String>();
    ArrayList<String> postBodies = new ArrayList<String>();
    ArrayList<String> topicIds = new ArrayList<String>();
    ArrayList<Date> dates = new ArrayList<Date>();

    List<TopicCreationResult> topicCreationResults = topicSystem.
    assignTopicsByName(postId, Networks.DEFAULT_NETWORK_ID, topicNames, true);

    for (TopicCreationResult topicResult : topicCreationResults) {
        networkIds.add(Networks.DEFAULT_NETWORK_ID);
        postBodies.add(postId);
        topicIds.add(topicResult.id());
        dates.add(postCreationDate);
    }

    int numberOfElements = topicCreationResults.size();

    // add matched suggestions to the topic index
    topicSuggester.feedbackRecord(orgId, networkIds.toArray(new String[numberOfElements]),
    postBodies.toArray(new String[numberOfElements]),
    topicIds.toArray(new String[numberOfElements]),
    dates.toArray(new Date[numberOfElements]), true);

    return numberOfElements;
}

protected String addPost(String data, UserInfo user, String parentId)
throws Exception {
    String[] posts = feedItemTestingUtil.addPosts(new String[] { parentId },
    new String[] { data },
    new FeedItemTypeEnum[] { FeedItemTypeEnum.ANNOUNCEMENT_POST },
    new String[] { data },
    new FeedItemTypeEnum[] { FeedItemTypeEnum.ANNOUNCEMENT_POST },
    new String[] { data },
    new FeedItemTypeEnum[] { FeedItemTypeEnum.ANNOUNCEMENT_POST });
user);

    return posts[0];
}

/**
* Ensures the validity of the line returned. If the line is invalid, next one is
* requested
*
* @param util
* @return a valid line
* @throws IOException
*/
protected Quadruple<Integer, String, List<String>, Date> getNextDataTuple(QualityEvaliationUtil util) throws IOException {
    String nextLine = null;
    Quadruple<Integer, String, List<String>, Date> nextTuple = null;

    boolean isLooping = true;
    while (isLooping) {
        nextLine = util.getNextDataLine();
        if (nextLine == null) throw new IOException();
        nextTuple = GsonConverter.fromJson(nextLine);
        // number of topics shouldn’t > 10
        if (nextTuple.getThird().size() > 10) continue;
        // each topic length shouldn’t > 99
        boolean isLong = false;
        for (String expectation : nextTuple.getThird()) {
            if (expectation.length() > 99) {
                isLong = true;
                break;
            }
        }
        if (isLong) continue;
        isLooping = false;
    }

    return nextTuple;
}

protected TopicSuggester getTopicSuggester() {
    return topicSuggester;
}

protected int getDataSetSize() {
    return dataSetSize;
}

protected int getNumberOfSuggestions() {
return numberOfSuggestions;
}

protected String getId() {
  return id;
}
}
Appendix E

Diversity evaluation

```java
public class DiversityEvaluationTest extends BaseEvaluationTest {
    private final String corePath;
    private final String home = "~/home/dkazarin";
    // this is an optimal value for a machine with 12 core CPU.
    private final int numberOfThreads = 3;

    public DiversityEvaluationTest(String name) {
        super(name);

        corePath = Paths.get("").toAbsolutePath().getParent().toString();
    }

    /**
     * This test computes a diversity index. Smaller the value (second column in
     * the output file) - higher the diversity. Third column is (1/Index). Each
     * time one word contains in 2 suggestions for the same suggestTopics call,
     * + 1 is added to the index.
     */
    public void testDiversity() throws Exception {
        int dataSetSize = 50;
        int sampleSize = 3;

        int numberOfSuggestions = 3;
        String outputFile = home +
                            "~/Downloads/Accuracy/Quora/DefaultDiversity.txt";

        // directory must exist before running this test
        String folderPath = corePath +
                            "~/sbi/test/func/java/src/sbi/quality/evaluation/data/quora/reuse/";

        String[] paths = PrecisionAndRecallEvaluationTest
                         .generateNFilePaths(folderPath, 3);

        assertEquals(paths.length, sampleSize);

        ExecutorService executorService = Executors
                                              .newFixedThreadPool(numberOfThreads);
    }
}
```
// test runs sampleSize times for each data set size
runNExperiment(executorService, sampleSize, numberOfSuggestions,
dataSetSize, outputFile, paths);

executorService.shutdown();
executorService.awaitTermination(BasicTestContext.getTotalTimeout(),
TimeUnit.SECONDS);
}

private void runNExperiment(ExecutorService executorService, int sampleSize,
int numberOfSuggestions, int currentDatasetSize, String outputFile,
String[] paths) throws IOException {
for (int currentSampleId = 0; currentSampleId < sampleSize;
currentSampleId++) {
    FileDataWriter dataWriter = new FileDataWriter(outputFile);
    Runnable worker = new DiverisyEvaluationWorker(String
        .valueOf(currentSampleId), FileInMemoryDataProvider.class,
        paths[currentSampleId], dataWriter, currentDatasetSize,
        numberOfSuggestions);
    executorService.execute(worker);
}
}

public class DiverisyEvaluationWorker extends AccuracyEvaluationWorker {
    private long diversityIndex;

    /**
     * Creates a new instance of DiverisyEvaluationWorker
     * @param id
     * evaluation id
     * @param dataProvider
     * class that represents a data provider
     * @param dataSource
     * path to data that is understandable by dataProvider
     * @param dataWriter
     * class that used as a data writer
     * @param datasetSize
     * size of training and evaluation data sets
     * @param numberOfSuggestions
     * number of suggestions that (link TopicSuggester) will generate
     */
    public DiverisyEvaluationWorker(String id,
    Class<? extends DataProviderBase> dataProvider, String dataSource,
    IDataWriter dataWriter, int datasetSize, int numberOfSuggestions) {
        super(id, dataProvider, dataSource, dataWriter, datasetSize,
        numberOfSuggestions);
    }

    @Override
    public void evaluate(SupportData supportData) {
        AccuracySupportData accuracySupportData = (AccuracySupportData)supportData;
        int numberOfItems = 0;
    }
while (numberOfItems < getDataSetSize()) {
    logProgress(numberOfItems, getDataSetSize(), "Evaluation progress:");
    try {
        Quadruple<Integer, String, List<String>, Date> evaluationLine =
            getNextDataTuple(accuracySupportData
                .getQualityEvaluationUtil());

        String postId = addPost(evaluationLine.getSecond(),
            accuracySupportData.getUser(),
            accuracySupportData.getParentId());

        // make suggestions
        List<TopicSuggestion> suggestions = getTopicSuggester()
            .suggestTopics(evaluationLine.getSecond(),
                getNumberOfSuggestions(), Collections.<String> emptyList());

        Collection<String> matchedRecommendations =
            foundMatchedWithIgnoredCase(suggestions,
                evaluationLine.getThird());

        saveTopics(postId, matchedRecommendations, evaluationLine.getFourth(),
            accuracySupportData.getOrgId());

        int similaritiesInSuggestions = countSimilarities(suggestions);
        diversityIndex += similaritiesInSuggestions;
    } catch (Exception ex) {
        logger.warning(ex.getMessage());
        // log the error and continue, there will be more lines
        continue;
    }
    numberOfItems++;
}

@Override
public void saveData(SupportData supportData) {
    AccuracySupportData accuracySupportData = (AccuracySupportData) supportData;
    // compute precision and recall
    double totalSeconds = getExecutionTime() / 1000;
    int hoursTaken = (int)(totalSeconds / 3600);
    int minutesTaken = (int)((totalSeconds - 3600 * hoursTaken) / 60);
    int secondsTaken = (int)(totalSeconds - 3600 * hoursTaken - 60 * minutesTaken);

    String[] result = new String[] {getId(), String.format("%d", diversityIndex),
        String.format("%.7f", diversityIndex == 0 ? 1 : (1 / (double)diversityIndex)),
        String.format("%d", getDataSetSize()),
        String.format("%02d:%02d:%02d", hoursTaken, minutesTaken, secondsTaken) };

    String resultLine = prepareResults(result);
    try {
        // save the result in a file
(accuracySupportData.getQualityEvaluationUtil())
    .writeResultLine(resultLine);
} catch (Exception ex) {
    throw new RuntimeException(ex.getMessage());
}

/**
 * Computes a number of similar suggestions in a list
 *
 * @return number of similar suggestions
 */
private int countSimilarities(List<TopicSuggestion> suggestions) {
    int similarities = 0;
    ArrayList<String> stop = new ArrayList<String>(
        Arrays.asList(TopicSuggester.stopWords));
    for (int i = 0; i < suggestions.size(); i++) {
        String suggestionName = suggestions.get(i).getName();
        for (int j = i + 1; j < suggestions.size(); j++) {
            String suggestionToCheck = suggestions.get(j).getName();
            if (suggestionToCheck.contains(suggestionName)) {
                similarities++;
            } else {
                String[] words = suggestionName.split(" ");
                if (words.length == 1) {
                    continue;
                } else {
                    for (String word : words) {
                        if (!stop.contains(word) && word.length() > 3
                            && suggestionToCheck.contains(word)) {
                            similarities++;
                        }
                    }
                }
            }
        }
    }
    return similarities;
}
Bibliography


[26] R.A. Turrin (General Chair) M. Torrens (General Chair) P. Resnick (Program Chair) M. Zanker (Program Chair) P. Cremonesi, Y. Koren. Performance of recommender algorithms on top-n recommendation tasks. 2010.