# CSCI5510: Project Proposal
# TagExtra

Due on Friday, Oct 4, 2013, 23:59:59

**Lu, Yi 1155041049 ylu@cse.cuhk.edu.hk**
**Qian, Yinling 1155041291 ylqian@cse.cuhk.edu.hk**
**Chen Xixian 1155036600 xxchen@cse.cuhk.edu.hk**
**Zhang, Zhi 1155039997 cheungZeeCn@gmail.com**

## 1 Motivation

Online information is wildly increasing since the last decade. Providing meaningful information timely has become urgent needs for companies like Microsoft, Google, Facebook, etc. Naively searching all the data assuredly won't meet the demand. A basic solution would come into mind: category the information by topics or keywords. This encounters a problem: not every item has been given topics or keywords. Manually extracting the keywords may work in small datasets, but it will never be applicable for a search engine company. Thus, we need to design an algorithm which would work on disparate stack exchange sites to predict the tags (a.k.a. keywords, topics, summaries).

## 2 Most related topics

We learn a lot about Big Data Analystic in class, we may happily apply the methods listed below(not limited to),

- MapReduce and Frequent Itemsets;
- Locality Sensitive Hashing;
- Mining Data Streams;
- Dimensionality Reduction;
- Online Learning.

## 3 deliverables

We plan to participate in competition on the website: https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction. So we would deliver a runnable system and get a rank in top 20

## 4 tentative dataset

The dataset is very large, almost consisting of 6.75GB training data and 2.20GB testing data.

For training data, it has 6034195 items (questions). Every item includes Id, Title, Body and Tags. For testing data, it has 2013337 items. Every item only contains Id, Title and Body. Following are some other things about the dataset,

- **Id** - Unique identifier for each question
- **Title** - The question's title
- **Body** - The body of the question

- **Tags** - The tags associated with the question (all lowercase, should not contain tabs ' \t' or ampersands '&')

The questions are randomized and contain a mix of verbose text sites as well as sites related to math and programming. The number of questions from each site may vary, and no filtering has been performed on the questions (such as closed questions).

# 5 techniques/algorithms we will apply

## 5.1 Word Frequency Analysis

Much early work concerned the frequency of term usage in the text, but most of this work focused on defining keywords in relation to a single document. The idea of statistically analyzing the frequency of keyword usage within a document in relation to multiple other documents became more common. This technique, known as Term Frequency - Inverse Document Frequency or simply TF-IDF, weights a given term to determine how well the term describes an individual document within a corpus. It does this by weighting the term positively for the number of times the term occurs within the specific document, while also weighting the term negatively relative to the number of documents which contain the term. When the TF-IDF function is run against all terms in all documents in the document corpus, the words can be ranked by their scores. A higher TF-IDF score indicates that a word is both important to the document, as well as relatively uncommon across the document corpus. This is often interpreted to mean that the word is significant to the document, and could be used to accurately summarize the document. In this case, we simply regard the questions as multiple document/text. However, the title of question may tell more useful information than the body. As a result, we may weight more on the title.

## 5.2 Word Co-Occurrence Relationships

Word co-occurrence aims to find similarity between words or similarities of meaning among word patterns. The sentences in the question text are considered as a set of words; So we extract keywords from text uses word co-occurrence to build a co-occurrence matrix. Words are important to the text if they co-occur with other words more often in the text than they would if every instance of the word were randomly distributed. For a certain word $w_i$, this can be thought of as the ratio of the number of co-occurrences of words $w_i$, $w_j$ to the number of all other co-occurrences involving wi. Under the given assumptions, a high ratio would mean that the word $w_i$ is a likely keyword for the question text.

# 6 Measure Method

We use Mean F1 Score to evaluate the performance of our system. The F1 score, commonly used in information retrieval, measures accuracy using the statistics precision

$p$ and recall $r$. Precision is the ratio of true positives ($tp$) to all predicted positives ($tp + fp$). Recall is the ratio of true positives to all actual positives ($tp + fn$). The F1 score is given by:

$$p = \frac{tp}{tp + fp} \quad (1)$$

$$r = \frac{tp}{tp + fn} \quad (2)$$

$$F1 = \frac{2pr}{p + r} \quad (3)$$

The F1 metric weights recall and precision equally, and a good retrieval algorithm will maximize both precision and recall simultaneously. Thus moderately good performance on both will be favored over extremely good performance on one and poor performance on the other. We have a labeled test set, prepared by Kaggle.com. For this metric, each row of the prediction represents a list of items that are "correct" for that row. We compute the F1 metric for each row: The "true positives" are the intersection of the two lists, false positives are predicted items that aren't real, and false negatives are real items that aren't predicted. There is a leader board provided byKaggle.com. However, this leader board is calculated on approximately 30% of the test data. The final results will be based on the other 70%, so the final standings may be different. The final report deadline of this course is 6, Dec, which is earlier than the competition deadline. Therefore, we can only report the performance of our system in the 30% of the test data.

## 7 Related work

It is popular for users in Web 2.0 era to freely annotate online resources with tags. To ease the annotation process, it has been great interest in automatic tag suggestion.

FolkRank [7] and Matrix Factorization [15] are representative collaborative filtering methods for social tag suggestion.

Some researchers regarded social tag suggestion as a classification problem by considering each tag as a category label [14, 13, 10, 8, 4, 5]. Various classifiers such as Naive Bayes, kNN, SVM and neural networks have been explored to solve the social tag suggestion problem.

There are two issues emerging from the classification-based methods:

- The annotations provided by users are noisy, and the classification-based methods can not handle the issue well.
- The training cost and classification cost of many classification-based methods are usually in proportion to the number of classification labels. These methods may thus be inefficient for a real-world social tagging system, where hundreds of thousands of unique tags should be considered as classification labels.

Inspired by the popularity of latent topic models such as Latent Dirichlet Allocation (LDA) [1], various methods have been proposed to model tags using generative latent topic models. One intuitive approach is assuming that both tags and words are generated from the same set of latent topics. By representing both tags and descriptions as the distributions of latent topics, this approach suggests tags according to their likelihood given the description [9, 12]. Bundschus et al. [2] proposed a joint latent topic model of users, words and tags. Iwata et al. [6] proposed an LDA-based topic model, Content Relevance Model (CRM), which aimed at finding the content-related tags for suggestion.

It should also be noted that social tag suggestion is different from automatic keyphrase extraction [11]. Keyphrase extraction aims at selecting terms from the given document to represent the main topics of the document. On the contrary, in social tag suggestion, the suggested tags do not necessarily appear in the given resource description. We can thus regard social tag suggestion as a task of selecting appropriate tags from a controlled tag vocabulary for the given resource description.

## 8 Rough timeline

We will start doing our project after the proposal peer review. We think we can collect enough valuable suggestions from the peer review.

We set two Milestones in our project.

- Milestone 1: Finish feature extracting with the help of mapreduce [3]. The due is Oct 25,2013
- Milestone 2: Finish tag predicting with our own algorithm. The due is Nov 8, 2013.

The remaining time is for refining our system.

## References

[1] Blei, D. M., and Jordan, M. I. Modeling annotated data. In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval (2003), ACM, pp. 127–134.

[2] Bundschus, M., Yu, S., Tresp, V., Rettinger, A., Dejori, M., and Kriegel, H.-P. Hierarchical bayesian models for collaborative tagging systems. In Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on (2009), IEEE, pp. 728–733.

[3] Dean, J., and Ghemawat, S. Mapreduce: simplified data processing on large clusters. Communications of the ACM 51, 1 (2008), 107–113.

[4] Fujimura, S., Fujimura, K., and Okuda, H. Blogosonomy: Autotagging any text using bloggers' knowledge. In Web Intelligence, IEEE/WIC/ACM International Conference on (2007), IEEE, pp. 205–212.

[5] Heymann, P., Ramage, D., and Garcia-Molina, H. Social tag prediction. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (2008), ACM, pp. 531–538.

[6] Iwata, T., Yamada, T., and Ueda, N. Modeling social annotation data with content relevance using a topic model. In Advances in Neural Information Processing Systems (2009), pp. 835–843.

[7] Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., and Stumme, G. Tag recommendations in social bookmarking systems. Ai Communications 21, 4 (2008), 231–247.

[8] Katakis, I., Tsoumakas, G., and Vlahavas, I. Multilabel text classification for automated tag suggestion. In Proceedings of the ECML/PKDD (2008).

[9] Krestel, R., Fankhauser, P., and Nejdl, W. Latent dirichlet allocation for tag recommendation. In Proceedings of the third ACM conference on Recommender systems (2009), ACM, pp. 61–68.

[10] Lee, S. O. K., and Chun, A. H. W. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ann semantic structures. ACOS 7 (2007), 88–93.

[11] Liu, Z., Chen, X., Zheng, Y., and Sun, M. Automatic keyphrase extraction by bridging vocabulary gap. In Proceedings of the Fifteenth Conference on Computational Natural Language Learning (2011), Association for Computational Linguistics, pp. 135–144.

[12] Liu, Z., Li, P., Zheng, Y., and Sun, M. Clustering to find exemplar terms for keyphrase extraction. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1 (2009), Association for Computational Linguistics, pp. 257–266.

[13] Mishne, G. Autotag: a collaborative approach to automated tag assignment for weblog posts. In Proceedings of the 15th international conference on World Wide Web (2006), ACM, pp. 953–954.

[14] Ohkura, T., Kiyota, Y., and Nakagawa, H. Browsing system for weblog articles based on automated folksonomy. In Proceedings of the WWW 2006 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, at WWW (2006), vol. 2006.

[15] Rendle, S., Balby Marinho, L., Nanopoulos, A., and Schmidt-Thieme, L. Learning optimal ranking with tensor factorization for tag recommendation. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (2009), ACM, pp. 727–736.