SmartNews: Towards content-sensitive ranking of comments

Marina Litvak
Sami Shamoon College of Engineering
Beer Sheva, Israel
marinal@sce.ac.il

Leon Matz
Sami Shamoon College of Engineering
Beer Sheva, Israel
leonm@sce.ac.il

Abstract

Various news sites exist today where internet users can read the most recent news and people's opinions about. However, usually these sites do not organize comments well and do not filter irrelevant content. Due to this limitation, readers who wonder about people's opinion regarding some specific topic, have to manually follow relevant comments, reading and filtering a lot of irrelevant text. In this work[1], we introduce a publicly available software implementing our approach, previously introduced in (Litvak and Matz, 2013), for retrieving and ranking the relevant comments for a given paragraph of news article and vice versa. We use Topic-Sensitive PageRank for ranking comments/paragraphs relevant for a user-specified paragraph/comment.

1 Introduction

Almost all modern news sites allow people to share their opinions by commenting a read article. However, usually comments are not organized well and appear under one long thread in chronological order. Some commenting systems include a rating component, but it is usually based on explicit feedback of users and does not relate to any specific content. In such conditions, the only way a reader can follow people's opinion about some *specific aspect* mentioned in the article is to scan manually a huge amount of comments.

Ranking comments on the web is a one of the central directions of IR in recent years (Dalal et al., 2012; Hsu et al., 2009). However, none of the works focused on the topic-sensitive ranking of comments. Since in many web domains like news different comments may refer to different aspects of the same article, resolving this problem is very important for structuring and better retrieval of user-contributed content.

In this paper we introduce an application for ranking comments in news websites relative to a given content[2]. The application provides ranked comments to the user-specified paragraph of a news item and, vice versa, ranked paragraphs that are relevant to a given comment. Our approach, that was previously introduced in (Litvak and Matz, 2013), is unsupervised and does not require training on an annotated data. It reduces the problem of topic-sensitive ranking of comments to the calculating of eigenvector centrality by adapting Topic Sensitive PageRank algorithm. The introduced application is implemented as a Chrome Extension to Yahoo! News site and is publicly available at author's homepage[3].

2 SmartNews

2.1 Problem Setting

We are given a set of comments $C_1, ..., C_m$ referring to an article describing some event and speaking on several related subjects. An article consists of a set of paragraphs $P_1, ..., P_n$ speaking on different related subjects. Our goal is, given paragraph P_i, to find a subset $C_{i_1}, ..., C_{i_r}$ of comments such that[4] (1) These are the most relevant to P_i comments that refer to topics described in P_i itself or comments about it; (2) The comments are ordered by the "relevancy" rank; (3) There are at most M comments.

Our method is based on eigenvector centrality

[1]This work was partially funded by the U.S. Department of Navy, Office of Naval Research.

[2]Here and further we refer to a paragraph as an independent text unit describing one of the article's aspects

[3]http://www.cs.bgu.ac.il/~litvakm/research/

[4]Here and further, we focus on comments ranking problem, while, generally, our method can be applied to the inverse problem – ranking paragraphs given a comment. Our plugin implements both directions.

The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations, pages 37–40,
Nagoya, Japan, 14-18 October 2013.

principle (PageRank, as its variant), that already has been successfully applied for ranking and extracting (Mihalcea and Tarau, 2004; Erkan and Radev, 2004) text units. Our approach consists of two main stages: (1) graph constructing and (2) computing the eigenvector centrality. Since this paper is focused on the *application* and not approach, the next two subsections briefly summarize both stages. The details can be found in (Litvak and Matz, 2013).

2.2 Graph Representation Model

In order to represent our textual data as a graph, we relay on the known factors influencing PageRank described in (Sobek, 2003). They are also enumerated and discussed in details in (Litvak and Matz, 2013). We organize comments as nodes in a graph (denoted by a **comments graph**), linked by edges weighted with text similarity score calculated between nodes. Formally speaking, we build a graph $G(E,V)$, where $N_i \in V$ stands for a comment C_i, and $e_k \in E$ between two nodes C_i and C_j stands for similarity relationship between texts of the two comments.[5] Each edge e_k is labeled by a weight w_k equal to the similarity score (we use cosine similarity (Salton et al., 1975)) between the linked text units. Edges with a weight lower then a pre-defined threshold are removed. By weighing links we diminish the influence of links between thematically unrelated text units and, conversely, increase the influence of links between strongly related ones. An example of resulted comments graph is demonstrated in Figure 1(a).

We treat a paragraph as a query that must to influent the resulted ranks of comments. We add an additonal node (denoted by a **query node**) for the paragraph with respect to which the comments should be ranked. The query node is also linked to the comments nodes by similarity relations, with weighted edges directed from a query node to comment nodes. Adding weighed inbound links from the query node to thematically related comment nodes must increase their PageRank relative to other nodes. Here and further, we call the resulted graph **extended graph**. This stage is demonstrated in Figure 1(b).

The situation, where extended graph has groups of strongly connected nodes, mostly thematically irrelevant to a query node, is created when we have comments "talking" to each other and deviate from the main (query) topic. It is enough that only one node from a group will be linked to a query node for "grabbing" a query's rank to a group and, at each iteration, enlarging the PageRank of strongly connected nodes. In order to avoid (1) PageRank increasing in unrelated nodes linked with related ones in a closed system and (2) "leakage" of PageRank in a query node, we add outbound links from comment nodes to a query node. For uniform impact on all comment nodes, we give all edges the same weights of 1. Comment nodes that are strongly related to a query, will gain their PageRank back in each iteration due to a high weight assigned to inbound links from a query node, while irrelevant nodes will "loose" their PageRank irretrievably. The described update applied to a graph from Figure 1(b) will result in a new structure depicted in Figure 1(c).

In order to obtain similarity scores between nodes standing for text units, we calculate cosine similarity between vectors representing related texts, according to the Vector Space Model (Salton et al., 1975). Formally speaking, each text unit–paragraph or comment–is represented by a real vector V of size n, where $V[i]$ stands for *tf-idf* (Salton et al., 1975) of term i and n is a vocabulary size. Since we treat each text unit as a *document*, we adapt tf-idf to *tf-ipf* (term frequency inverse *paragraph* frequency) and *tf-icf* (term frequency inverse *comment* frequency) when applied on a paragraph or comment, respectively. The details and exact formulas can be retrieved from (Litvak and Matz, 2013).

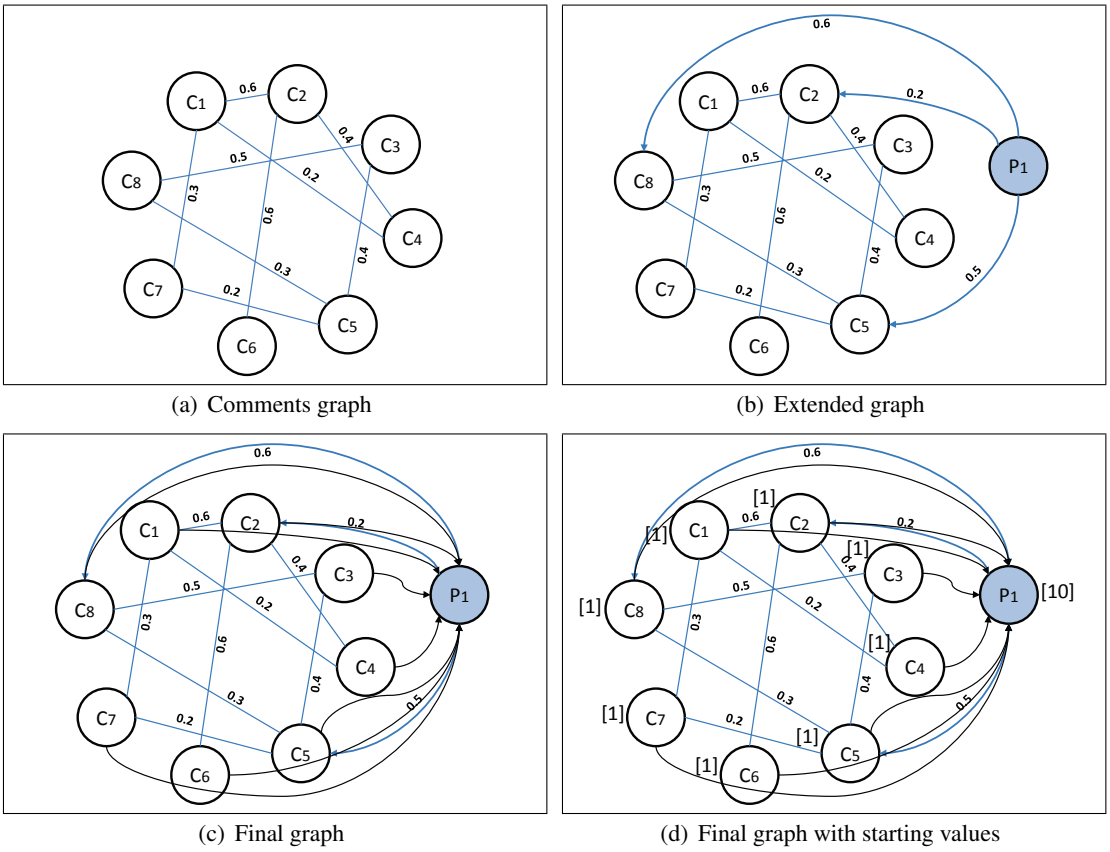
2.3 Computing the eigenvector centrality

For ranking and retrieving comments, we compute their eigenvector centrality by applying PageRank algorithm (Brin and Page, 1998) to an extended graph.

In order to influence node's rank by a query node for topic-sensitive retrieval, we relay on the known factors influenting PageRank score which are enumerated and described in (Litvak and Matz, 2013).

First, we give a high starting value to a query node before the iterative computation of PageRank begins. Adding outbound links from comment nodes to a query node (described in 2.2) helps to keep high PageRank in the query node through successive iterations. The final graph structure

[5] For the inverse problem, we represent a document as a graph of paragraphs (aka **paragraphs graph**) linked by a similarity relationship (Salton et al., 1997).



(a) Comments graph (b) Extended graph

(c) Final graph (d) Final graph with starting values

Figure 1: Graph representation: four steps.

including initial starting values is shown in Figure 1(d).

Second, in order to implement a theme-based retrieval, we adapt the idea of Topic Sensitive PageRank[6], where the thematically relevant comments get higher damping factor d. The final formula for ranking comments looks as follows.

$$PR(a) = E(a)d + (1-d) \sum_{x \in adj(a)} \frac{PR(x)w(a,x)}{\sum_{y \in adj(x)} w(y,x)},$$

where $E(a) = \frac{w(a,q)}{\sum_{i \in V} w(i,q)}$, q is a query node, $w(x,y)$ is a similarity score between nodes x and y.

We treat a PageRank score as a final rank of items. In a greedy manner, we extract and present at most M most ranked comments ordered by their rank to the end user. In our settings, $M = 5$.

3 Implementation Details

We implemented the introduced approach as a Chrome Extension (plugin) for the Yahoo! News[7] website. The plugin contains two sides: (1) **client** responsible for a data collecting and the results representation, and (2) **server** calculating ranks in a background.

Client performs the following: (1) collects the necessary textual and meta data and transfers it to the server, (2) visualizes the output (ranked comments and paragraphs, etc.) to the end user. The initial filtering of textual data is performed before transferring it to the server. The comments containing no words (considering synonyms) in common with the article are discarded. We used the following technologies for client's implementation: Javascript and jQuery Folder for scanning the article and collecting the relevant data, and JSON object as a data structure.

Server performs the following: (1) gets the textual data, (2) applies standard preprocessing including: HTML parsing, paragraph and sentence splitting, tokenization, stopwords removal, stemming, and synonyms resolving[8] for handling texts expressing the same issues with different vocab-

[6]Topic-Sensitive PageRank is a very intuitive choice in our setting, since we retrieve comments *with respect to* a given paragraph representing a topic an actual user is interested in.

[7]http://news.yahoo.com/

[8]with Synonym Map http://lucene.apache.org/core/old_versioned_docs/versions/2_9_1/api/all/org/apache/lucene/index/memory/SynonymMap.html

ulary, (3) builds VSM representation, then (4) builds graph representation, (5) calculates ranks of comments given a specified paragraph as a query or vice versa, (6) converts the processed data into Json object, and (7) transfers it to the client. We used the following technologies for server's implementation: Java EE, Tomcat server, Spring Environment.

In order to apply a Topic-Sensitive PageRank for a specific paragraph[9] we identify the actual paragraph a user is interested in by sending the position of the user's mouse to the server.

Figure 2 demonstrates the infrastructure of the plugin including interconnection between client (front end) and server (back end) sides.

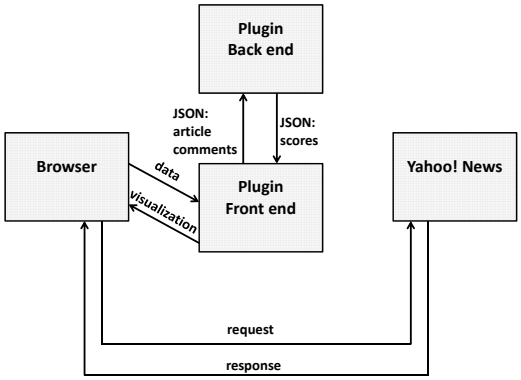


Figure 2: Application infrastructure.

The computational complexity of our approach depends on graph construction time, that is quadratic in number of comments/paragraphs in a given article. In practice, it takes about two seconds to perform precomputation–graph construction and ranks calculation on all article-related data–when user opens an article page, and then the results for any user-specified paragraph/comment are provided immediately.

4 Conclusions

The examples of article texts and the most ranked comments, per paragraph, can be seen in `http://goo.gl/7idNw`. It can be seen that the comments are very related to the paragraphs content and, moreover, they relates the **subject** of a paragraph as well as a **discussion** and **opinions** it arises, beyond the text overlapping. Such performance is provided by a recursive nature of PageRank, where the relationships between comments are iteratively elaborated. Unlike this approach, ranking comments by a (text) similarity to a given paragraph would not retrieve related comments with a different vocabulary.

The plugin implementing our approach is publicly available from `http://goo.gl/To4Rd`.[10] In future, we intend to evaluate our system by comparing it to the other state-of-the-art ranking techniques.

Acknowledgments

Authors thank project students: M. Magaziner, A. Shpilgerman and S. Pinsky for implementing the introduced approach, and I. Vinokur for a technical support of the software. Especial thanks to Dr. Amin Mantrach from Yahoo! Labs, Barcelona, for very constructive and helpful comments.

REFERENCES

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117.

Dalal, O., Sengemedu, S. H., and Sanyal, S. (2012). Multi-objective ranking of comments on web. In *Proceedings of the 21st international conference on World Wide Web*, pages 419–428.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Hsu, C.-F., Khabiri, E., and Caverlee, J. (2009). Ranking comments on the social web. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 04*, pages 90–97.

Litvak, M. and Matz, L. (2013). Smartnews: Bringing order into comments chaos. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, KDIR '13.

Mihalcea, R. and Tarau, P. (2004). Textrank – bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Salton, G., Singhal, A., Mitra, M., and Buckley, C. (1997). Automatic text structuring and summarization. *Information Processing and Management*, 33(2):193–207.

Salton, G., Yang, C., and Wong, A. (1975). A vector-space model for information retrieval. *Communications of the ACM*, 18.

Sobek, M. (2003). A Survey of Google's PageRank. `http://pr.efactory.de/`.

[9]The original idea of a Topic-Sensitive PageRank was to calculate PageRank for several topics simultaneously, but we don't need to do that until a user is interested in all paragraphs of a given article.

[10]Unzip the archive, press "Load unpacked extension" in "Developer mode" of chrome "Extensions" tool, and choose the unzipped plugin folder.