QALink: Enriching Text Documents with Relevant Q&A Site Contents

Yixuan Tang
School of Computing, National University of Singapore
yixuan@comp.nus.edu.sg

Weilong Huang
School of Computing, National University of Singapore
weilong@comp.nus.edu.sg

Qi Liu
School of Computing, National University of Singapore
qiliu@comp.nus.edu.sg

Anthony K.H. Tung
School of Computing, National University of Singapore
atung@comp.nus.edu.sg

Xiaoli Wang
Software School, Xiamen University China
xlwang@xmu.edu.cn

Jisong Yang
SeSaMe Centre, National University of Singapore
idmyj@nus.edu.sg

Beibei Zhang*
Department of Computing, Hong Kong Polytechnic University
Hong Kong, China
beibei.zhang@connect.polyu.hk

ABSTRACT

With rapid development of Q&A sites such as Quora and StackExchange, high quality question-answer pairs have been produced by users. These Q&A contents cover a wide range of topics, and they are useful for users to resolve queries and obtain new knowledge. Meanwhile, when people are reading digital documents, they may encounter reading problems such as lack of background information and unclear illustration of concepts. We believe that Q&A sites offer high-quality contents which can serve as rich supplements to digital documents. In this paper, we devise a rigorous formulation of the novel text enrichment problem, and design an end-to-end system named **QALink** which assigns the most relevant Q&A contents to the corresponding section of the document. We first present a new segmentation approach to model each document with a hierarchical structure. Based on the hierarchy, queries are constructed to retrieve and rank related question-answer pairs. Both syntactical and semantic features are adopted in our system. The empirical evaluation results indicate that **QALink** is able to effectively enrich text documents with relevant Q&A contents to help people better understand the documents.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; *Document structure*;

KEYWORDS

text enrichment; Q&A sites; hierarchical text segmentation; probabilistic information retrieval; learning to rank

ACM Reference Format:
Yixuan Tang, Weilong Huang, Qi Liu, Anthony K.H. Tung, Xiaoli Wang, Jisong Yang, and Beibei Zhang. 2017. QALink: Enriching Text Documents with Relevant Q&A Site Contents. In *Proceedings of CIKM'17 , Singapore, Singapore, November 6–10, 2017,* 10 pages.
https://doi.org/10.1145/3132847.3132934

1 INTRODUCTION

Q&A sites such as Quora and StackExchange are gaining popularity in recent years. According to Quora[1], there are over 320,000 topics in various categories, such as education, sports and politics. The Q&A contents on those websites are contributed by their community users, who collaborate on editing questions and answers. This community-driven knowledge creation process has largely encouraged user interactions. High-quality question-answer pairs are produced by professional users and selected by their readers. Moreover, voting, ranking and reputation scoring schemes are commonly adopted by Q&A websites to promote useful information and to eliminate noises.

When people are reading digital documents (academic publications and magazines), they may fail to understand certain contents possibly due to the lack of background information, unclear presentation of key concepts, or lack of rigorous proof or illustrative examples in the document. On the other hand, readers may find the document interesting and they may want to read more related materials. In both scenarios, high-quality contents provided by Q&A sites can offer rich supplementary materials to enrich the document. Linking Q&A contents to the article not only satisfies the information need of readers, but also saves them time in searching for relevant information. The following is an example of a document that explain the use of SVM in stock market.

*Work done while the author was an intern at SeSaMe Cen- tre, National University of Singapore.

[1] https://www.quora.com/sitemap/topics

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132934

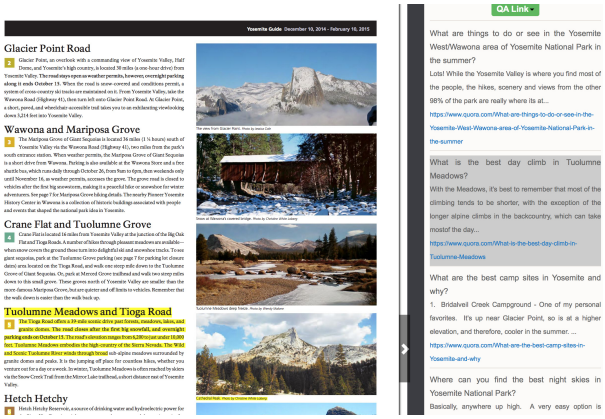

Figure 1: User interface of QALink

> We used the support vector machines (SVM) in a classification approach to 'beat the market'. Given the fundamental accounting and price information of stocks trading on the Australian Stock Exchange, we attempt to use SVM to identify stocks that are likely to outperform the market by having exceptional returns.

The author of the text assumes readers to have background knowledge about SVM. For those who are not familiar with SVM, Q&A contents such as 'What is a Support Vector Machine?'[2] and 'How to implement a SVM?'[3] are expected to be helpful since they resolve doubts for readers. Meanwhile, contents like 'why is ANN classification (regression) better than SVM?'[4] and 'How to predict stock market values?' [5] may also be useful for readers since they provide highly related extensive materials.

In this paper, we propose a method, denoted as **QALink**, to automatically enrich text documents with relevant Q&A contents. To ensure that supplementary materials are placed in close proximity to relevant portion of the main text, **QALink** first utilizes a novel algorithm to hierarchically segment the document and then retrieves relevant Q&A contents for each segment of the document. Retrieval and ranking of Q&A contents are done by submitting keyword queries that combine global and local information for each segment based on the segment hierarchy. Figure 1 shows the user interface of **QALink** 's enrichment result for a *Yosemite travel guide book*. The main document is present on the left, while the related Q&A contents retrieved are shown on the right. In order to avoid excessive reading burden, only a small number of most related contents are listed for each segment. Additionally, when a Q&A content is selected, the corresponding segment in the main document will be highlighted.

Main contributions of this work are:

- We address a novel problem of enriching text documents with Q&A contents and explain why they enhance people's reading experiences. The linked Q&A contents should be relevant and helpful for readers to understand the documents. To the best of our knowledge, no previous work has addressed this problem.
- We design an end to end system called **QALink** to provide supplementary Q&A contents for related parts of the document. A novel document segmentation algorithm is adopted. Unlike existing algorithms, it is able to segment a document hierarchically without requiring structural information. Based on the segmentation, **QALink** proceeds to retrieve and rank related Q&A contents using a measure called combinedBM25. The score tells the importance of each Q&A content with regards to the related text segment.
- We use datasets collected from Quora and StackExchange to evaluate the effectiveness of **QALink**. Both qualitative and quantitative evaluations demonstrate that our approach can better help readers understand the documents and provide more useful information compared to previous works.

The rest of the paper is organized as follows. Related work is reviewed and discussed in Section 2. Section 3 formally defines the problem. Section 4 presents each step of our system in detail. We provide our experimental analysis in Section 5 and the conclusion in Section 6.

2 RELATED WORK

Document Enrichment: Existing work on document enrichment can be roughly grouped into three categories, namely (1) enriching textbooks with images and videos, (2) enriching documents with Wikipedia contents, and (3) short documents enrichment.

The first group consists of works on enriching textbook sections with images and videos [2–4, 22]. In [2] and [3], the papers focus on finding images from the web that are relevant to textbook sections. They extract concept phrases from a given section and use those phrases to retrieve images from online search engines. As an image may be linked to different sections, they also propose assignment algorithms to ensure no image is repeatedly linked to multiple sections. [4] and [22] propose similar techniques to link relevant videos to textbook sections. However, those works are restricted to documents with explicit section structures. They model the document structure based on the original sections of a document which cannot guarantee the relevancy of contents from a global point of view. Besides, the effectiveness of their approach on the documents other than textbooks is not demonstrated.

The second group focuses on linking textual web contents like Wikipedia pages to keywords in documents. They seek to link Wikipedia entries to keywords in a text document to help users understand the meanings of the keywords[8, 17, 24]. The drawback is that the context of the keywords is not considered. As a result, those works are helpful for understanding the concepts instead of the whole document.

The last group addresses the problem of linking web contents to short documents [13, 15, 21, 31]. Some of them use Wikipedia pages to enrich a short text [13, 15]. Kang et al. focus on enriching Wikipedia pages using tweets [21] and Tsagkias et al. attempt to enrich online news with social media contents [31]. However, such effort cannot support voluminous documents. For a long document,

[2]https://www.quora.com/What-is-a-Support-Vector-Machine
[3]http://stackoverflow.com/questions/12141589/
[4]http://stackoverflow.com/questions/8326485/
[5]http://stackoverflow.com/questions/1989992/

web contents can not be effectively linked to a specific part of the document.

To the best of our knowledge, no existing work has solved the same problem as ours. We propose a hierarchy based approach which is able to capture context information. Our method supports all types of free text documents, ranging from short to long. It can also deal with documents with no explicit section structures.

Hierarchical Text Segmentation: Our work is also related to hierarchical text segmentation. The use of representing a document with hierarchical structure has been shown in existing work [32]. The hierarchy of a document can be obtained by running single-level text segmentation algorithms several times (e.g. [6, 10, 14, 18]) or by hierarchically segmenting the document directly [27, 28, 32].

For single level text segmentation algorithms, **HC**99 [10] and **HCWM** [14] are two representative works. However, such algorithms do not guarantee global optimum. For the latter approach, Yaari employs agglomerative clustering to produce the hierarchy [32]. The adapted clustering method, called **HAC**, is a greedy algorithm and also often converges to a local optimum. Slaney et al. use scale-space segmentation to compute the semantic path of the document using Latent Semantic Indexing [27]. However, it is difficult to construct the hierarchy with the output of the image segmentation algorithm. In [28], Song et al. propose an iterative algorithm to split a document into two at a place where the cohesion link is the weakest and then create the hierarchy using a binary tree. We denote this type of work as **BiSeg**. In addition, all the works above do not take semantically similar words into consideration. Our approach remedies this problem by adopting word embedding [29]. We also propose a novel hierarchical text segmentation algorithm that guarantee to achieve global optimum using dynamic programming.

3 PROBLEM DEFINITION

We formulate the problem as follows. Given a document D and a list of L Q&A contents $\{C_1, ..., C_L\}$. We aims to partition the document into a set of coherent text segments, i.e.$D = \{s_1....s_k\}$, and to retrieve top-K most relevant contents for each text segment s_i.

Each content C_i consists of a question Q_i, a topic set $T_i = \{t_{i,1}, ..., t_{i,|T_i|}\}$ and an answer set $A_i = \{a_{i,1}, ..., a_{i,|A_i|}\}$. Q_i contains the question title and the description. T_i contains crowdsourced topics, such as 'Politics' and 'Programming'. For simplicity, both topics and answers are concatenated into one sentence for each question.

The problem is challenging in the following aspects. First, the method should be able to handle various formats of documents with either explicit or implicit section structures, such as slides, web pages and books. Second, the language gaps between documents and Q&A contents should be tackled. Third, the retrieved contents should not only be relevant to the text segment but also stay relevant to the document from a global point of view.

4 METHODOLOGY

The overall system architecture of **QALink** is summarized in Figure 2. Given a document D, we first segment the document hierarchically into coherent text segments $D = \{s_1....s_k\}$. With this

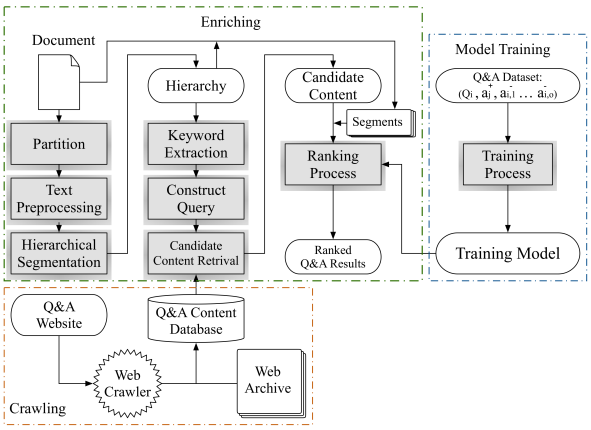

Figure 2: Overall System Architecture

approach, we can capture hierarchical relations instead of taking a flat view towards the document. Then, we extract keywords for each node in the hierarchy and generate a query based on a combination of the keywords along the path from the root node to the leaf node. This allows the query to contain both the local and global information. With the query, we can retrieve and rank relevant Q&A contents and attach them to the corresponding segment.

4.1 Hierarchical Text Segmentation

Given a document D, we first divide it into N ($N > k$) partitions using sentences or paragraphs as the boundary, i.e. $D = \{p_1....p_N\}$. If no such boundaries are provided, the text is simply equally divided. Next, vector space model [12] is adopted to represent each partition. We perform standard procedures for text preprocessing, including tokenizetion, stop words removal and word stemming. Then, each partition is converted to a vector using TF-IDF [5] based term weighting. Former works [6, 10, 18] use cosine similarity to measure the similarity of two text vectors. Since each dimension of the text vector represent one word, such method results in losing connection of words with similar meanings but different forms. For example, 'car' and 'automobile' are considered as two different words and their connection would be ignored when calculating cosine similarity. In order to remedy this problem, a modified cosine similarity $SIM(p_i, p_j)$ of two partition vectors is defined based on word embedding [29]:

$$SIM(p_i, p_j) \propto \sum_{w_m \in p_i} \sum_{w_n \in p_j} tfidf(w_m, p_i) \cdot SIM(w_m, w_n) \cdot tfidf(w_n, p_j) \tag{1}$$

where $tfidf(w_m, p_i)$ is the TF-IDF score of word w_m in partition vector p_i; $SIM(w_m, w_n)$ is the cosine similarity of Word2Vec representations of words w_m and w_n. Word2Vec represents words using sparse vectors instead of one-hot encodings. Words with similar meaning are close to each other in the embedding space and thus have higher $SIM(w_m, w_n)$. After incorporating the Word2Vec similarity, we are able to capture the relations between semantically similar words.

Algorithm 1 Dynamic Programming

Require: The N partition vectors and cluster number K
Ensure: An optimal clustering of partitions into K clusters

```
1:  for k = 1 to K do
2:    for b = k to N do
3:      if k == 1 then
4:        f(b,k) ⇐ q(1,b);
5:      else
6:        f(b,k) ⇐ 0;
7:        for l = 2 to b do
8:          newValue ⇐ f(l−1,k−1) + q(l,b);
9:          if newValue > f(b,k) then
10:           f(b,k) ⇐ newValue;
11:           Boundary(b,k) = l
```

Next, in order to obtain the coherent text segments, similar partitions should be grouped together. Note that this problem can be regarded as a clustering problem. Given N partitions, $D = \{p_1 ... p_N\}$, K clusters $D = \{s_1 ... s_K\}$ are constructed. Each s_i is a set of continuous partitions in the original document and no two s_i and s_j are overlapped. In the remaining part of this paper, we use s_i to refer to a text segment and a cluster interchangeably. Hence, our problem is equivalent to find $K-1$ boundaries for N partitions. For example, given 4 partitions $\{p_1, p_2, p_3, p_4\}$, if we split between p_1 and p_2, p_2 and p_3, we get three text segments, i.e. $s_1 = \{p_1\}$, $s_2 = \{p_2\}$, $s_3 = \{p_3, p_4\}$.

Then, a new objective function is formulated to measure the quality of clustering. The objective function is motivated by k-means [1]. It is the sum of the qualities of K text segments.

$$f(\{s_1, ..., s_K\}) = \sum_{i=1}^{K} q(s_i) \tag{2}$$

To measure quality $q(s_i)$, similar to k-means, the centroid vector Z_i of s_i is defined as the average of the partition vectors. For example, given $s_3 = \{p_3, p_4\}$, $Z_3 = \frac{p_3+p_4}{2}$. Given Z_i, the quality is defined in Equation (3). We iteratively compute the similarity of partition vector and centroid vector using the modified cosine similarity defined in Equation (1) and sum them up to generate $q(s_i)$.

$$q(s_i) = \sum_{p_j \in s_i} SIM(p_j, Z_i) \tag{3}$$

Having fixed the choice of objective function, the problem of finding the boundaries for K text segments $\{s_1, ..., s_K\}$ can be optimally solved by a dynamic programming algorithm. The solution is motivated by computing space-bounded V-Optimal histograms [19]. Different from [19], we manipulate vectors instead of scalar values here. In order to find an optimal solution, we can divide the objective function into smaller parts. Let $f^*(b,k)$ denote the maximum quality of grouping first b partitions into k text segments. We have the following recursive relationship:

$$f^*(b,k) = \max_{1<l\leq b} \{f^*(l-1, k-1) + q(l,b)\}, \tag{4}$$

where $q(l,b)$ is computed as the quality of text segments consisting of partitions from l to b(both inclusive). By finding a best boundary l, the optimization problem of grouping first b partitions into k text segments is reduced to a sub-problem of grouping first $l-1$ partitions into $k-1$ clusters. Partitions from l to b form the last text segment, s_k. With this recursive equation, the problem can be solved in a bottom-up fashion using dynamic programming. The proof of global optimality can be found in Appendix A.

Algorithm 1 shows the algorithm for computing the optimal clustering. Line 3-4 compute the maximum quality when clustering all partitions into 1 text segment. Line 6-10 compute the maximum quality when grouping b partitions into $k(\geq 2)$ clusters using the calculated results on the optimal clustering of $k-1$ clusters. The value of $q(j,i)$ is calculated in advance so that we can retrieve it directly for Line 8. Line 10 updates the optimal grouping when higher quality value is found. Finding an optimal clustering of K clusters on N partitions takes $O(N^2K)$ time and $O(NK)$ space.

After dynamic programming, we can construct k text segments by backtracking using the values in array *Boundary*. Note that the array also contains the optimal clustering for grouping N partitions into v (for $1 \leqslant v \leqslant k-1$) text segments. Thus, a hierarchy of the document can be obtained by stacking those optimal clusterings. Thus, the leaf level would be the optimal clustering of $f(N,K)$, while the layer on top of it would be the optimal clustering of $f(N,K-1)$. Figure 3 shows a hierarchy of clustering generated from a real text document about data mining. In each level, a node is split into two. Thus, the cluster number increases one in every level. Compared to existing work described in Section 2, the proposed method gains two obvious benefits. Firstly, algorithm 1 guarantees that the result is a global optimal solution instead of local optimums. Moreover, the hierarchical clustering can be built by running algorithm 1 just once.

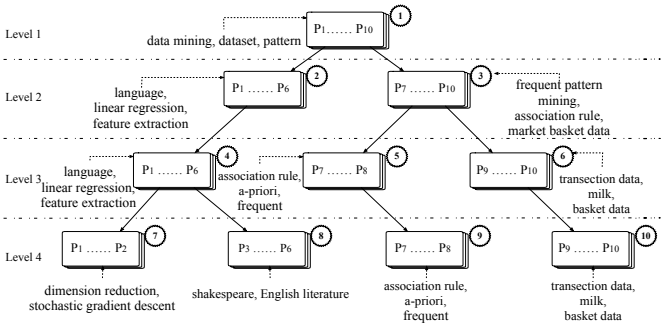

Figure 3: An example of text segmentation

4.2 Keyword Extraction and Query Generation

After building the hierarchy H, we summarize the document and retrieve relevant contents for each text segment. Towards this aim, keywords are first extracted and queries are then generated for each node in the hierarchy H. Note that each node in the hierarchy corresponds to a range of text in the document. For example, node 2 in Figure 3 contains partitions from 1 to 6. Same as for segmentation, TF-IDF based term weighting is not favourable for keyword extraction, since it fails to capture the connections between semantically

similar words. Also, it leads to low diversity for keywords extracted. Thus, we propose a graph based approach to extract keywords.

Given a node in hierarchy H, we model each word in the contents of this node as a vertex in V of graph $G = \{V, E\}$. To capture the tie strength of two words w_i and w_j, the tie value is calculated as:

$$TIE(w_i, w_j) = \lambda SIM(w_i, w_j) + (1-\lambda) log \frac{p(w_i, w_j; D)}{p(w_i; D) \cdot p(w_j; D)} \quad (5)$$

where $SIM(w_i, w_j)$ is the Word2Vec similarity of two words. $p(w_i, w_j; D)$ is the probability of words w_i and w_j appearing together in document D. It is estimated by the number of sentences containing both words divided by total number of sentences in D. Similarly, $p(w_i; D)$ and $p(w_j; D)$ are probabilities of words w_i and w_j appearing in document D. λ is a weight parameter. Equation (5) captures both word similarities and the closeness of two words in the document.

If $TIE(w_i, w_j)$ is larger than a threshold, we add an undirected edge between the two words in graph G. In order to extract most informative keywords from the graph, degree centrality is adopted. The keyword with the largest degree is first selected. Then, this keyword and its associated edges are removed from the graph in order to increase diversity of the selected keywords. We continue with this method until top-K keywords are selected. After extracting keywords for each node in the hierarchy H, the keywords form a hierarchical structure. It should be noted that because there is only one node split in each layer, we do not need to re-extract keywords for those nodes that are not split. Hence, the method is efficient enough to process multiple layers.

After extracting keywords, queries for each text segment s_i (leaf node in hierarchy) can be generated. In order to capture global information, we generate the query for each leaf node by combining the keywords along the path from the root node to leaf node. For example, in Figure 3, the query path of leaf node 8 is the set $\{1, 2, 4, 8\}$. The extracted keywords for node 8 are 'Shakespeare' and 'English literature'. Those keywords are relevant locally. However, it's not so related to the global view of 'data mining'. In this case, a union of keywords that belongs to nodes in the query path can easily overcome the problem. For node 8, we collect keywords of nodes $1, 2, 4, 8$ to form a query. Besides 'Shakespeare', it also contains words like 'data mining' and 'linear regression'. Based on the hierarchical structure, our method is able to capture global contexts and to support the query with background knowledge.

4.3 Candidate Contents Retrieval

After the query generation, each text segment s_i is associated with a query $R_i = \{r_1, ..., r_{|R_i|}\}$ where each r_j is a keyword. We first use the query to quickly retrieve candidate contents and then rank candidate contents according to their similarity to the corresponding text segment. Okapi BM25 [26] method is one of the state-of-the-art methods for ad-hoc retrieval. For a query R_i, the score of a candidate document d is calculated as:

$$BM25(R_i, d) = \sum_{r_j \in R_i} \frac{boost(r_j, R_i) \cdot idf(r_j) \cdot c(r_j, d) \cdot (\alpha + 1)}{c(r_j, d) + \alpha(1 - \beta + \beta \frac{len(d)}{avgl(d)})} \quad (6)$$

where a boost is set for each r_j in R_i, $idf(r_j)$ is the inverse document frequency, $c(r_j, d)$ is the frequency of r_j in document d, $len(d)$ is the length of d, $avgl(d)$ is the average length of d in the corpus and α, β are two parameters. BM25 can be computed efficiently on inverted index and it is employed to retrieve top-K Q&A contents from $\{C_1, ..., C_L\}$ as follows. For each C_j, the $CombinedBM25(R_i, C_j)$ with respect to query R_i is:

$$CombinedBM25(R_i, C_j) = \gamma_1 BM25(R_i, Q_j) + \gamma_2 BM25(R_i, T_j) + \frac{(1-\gamma_1-\gamma_2)}{|A_j|} \sum_{a_m \in A_j} BM25(R_i, a_m) \quad (7)$$

where BM25 scores on the question Q_j, the topic T_j and the answers A_j are combined with regards to the Q&A content C_j. The weights γ_1 and γ_2 are used to control the importance of the three factors. By calculating the $CombinedBM25(R_i, C_j)$, top-K related contents can be retrieved efficiently for each query R_i. Then, the retrieved contents are further ranked with regards to the segment text using the score function below:

$$score(R_i, C_j) = p(R_i|Q_j) + p(R_i|T_j) + \frac{1}{|A_j|} \sum_{a_m \in A_j} cos(vec(s_i), vec(a_m)) \quad (8)$$

where $p(R_i|Q_j)$ and $p(R_i|T_j)$ are defined as:

$$p(R_i|Q_j) = \prod_{r_n \in R_i} p(r_n|Q_j)$$
$$p(R_i|T_j) = \prod_{r_n \in R_i} p(r_n|T_j) \quad (9)$$

To estimate $p(r_n|Q_j)$ (similar definition can be applied to $p(r_n|T_j)$), we can regard the probability as a translational procedure [7]. For each word w_m in Q_j, it is translated to word r_n as follows:

$$p(r_n|Q_j) = \mu \sum_{w_m \in Q_j} p(r_n|w_m) p_{ml}(w_m|Q_j) + (1-\mu) p_{ml}(r_n|Q_j) \quad (10)$$

$p(r_n|w_m)$ is the translational probability and is simply calculated using Equation (5). Maximum likelihood (ML) estimator is used to estimate $p_{ml}(r_n|Q_j)$, i.e. $p_{ml}(r_n|Q_j) = \frac{c(r_n, Q_j)}{len(Q_j)}$ where $c(r_n, Q_j)$ is the count of r_n in Q_j, $len(Q_j)$ is the length of Q_j. Similar definition can be applied to $p_{ml}(w_m|Q_j)$. To avoid zero count and overfitting problems, we estimate $p_{ml}(r_n|Q_j)$ as follows [20]:

$$p_{ml}(r_n|Q_j) = \theta \frac{c(r_n, Q_j)}{len(Q_j)} + (1-\theta) \frac{c(r_n, Y)}{len(Y)} \quad (11)$$

Y is a reference corpus. The count of r_n in Y and the length of Y are used to smooth the ML estimation.

In Equation (8), $cos(vec(s_i), vec(a_m))$ calculates the cosine similarity of the text embedding vectors $vec(s_i)$ and $vec(a_m)$, which encodes the segment and the answer in the vector space. This part of the function assigns higher scores to those semantically similar texts and lower score to those dissimilar texts. Note that if we view s_i as a "question" like the questions of Q&A contents, we should give answer a_m higher score if it can "answer" this question. Thus,

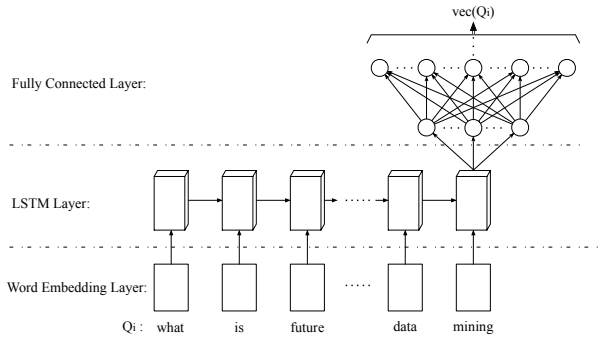

Figure 4: Architecture of neural network

we can construct a training dataset by pulling out questions Q_i and answers A_i from Q&A contents $\{C_1, ..., C_L\}$. The dataset is in form of $< Q_i, a_j^+, a_{i,1}^-, ...a_{i,o}^- >$. Q_i is the question of a content C_i and $a_j^+ \in A_i$. We only use answers with high up-votes (larger than 10 in the experiments) as the positive examples a_j^+. $\{a_{i,1}^-, ...a_{i,o}^-\}$ is a list of randomly selected negative examples. The cosine similarities of Q_i and a_j^+ should be higher than any of Q_i and $\{a_{i,1}^-, ...a_{i,o}^-\}$. Thus, the loss function is defined as follows:

$$Loss(< Q_i, a_j^+, a_{i,1}^-, ...a_{i,o}^- >) = max(0, U - cos(vec(Q_i), vec(a_j^+)) + margin) \quad (12)$$

where:

$$U = max(cos(vec(Q_i), vec(a_{i,1}^-)), ..., cos(vec(Q_i), vec(a_{i,o}^-))) \quad (13)$$

The loss function is a hinge loss. We take the largest cosine similarity between Q_i and answers in $\{a_{i,1}^-, ...a_{i,o}^-\}$ and compare it with the cosine similarity $cos(vec(Q_i), vec(a_j^+))$.

In order to learn the vector representations for questions and answers, a neural network is adopted, as shown in Figure 4. It consists a word embedding layer, a long short term memory (LSTM) layer and a fully connected (FC) layer. Firstly, each word of the given text, e.g. Q_i, is projected into a word embedding vector. Next, word embedding vectors are fed into LSTM units. Recurrent neural networks with LSTMs are state-of-art structures for many NLP tasks, such as machine translation and language modeling. LSTMs have shown a remarkable ability in capturing context information for long sequences. Then, the FC layer takes the output vector of the LSTM layer and simply computes output $O = acti(Wx)$.

For one question, Q_i and its related answers are fed into the network independently. The weights are shared for the question and answers. With the loss function, stochastic gradient descent and back propagation are used to train the network. To speed up the training process and to avoid gradient vanishing, we adopt leaky ReLU [23] as the activation function $acti()$. After training, the neural network can be used to calculate the vector representation of the input texts. In the example of Figure 4, the output is $vec(Q_i)$.

5 EXPERIMENTAL EVALUATION

5.1 Datasets

Two types of data are used in our experiments. First type contains three document datasets, and the second type includes two Q&A datasets which are used to enrich the documents.

Document Datasets: The document lengths of three datasets vary from short to long.

- The *Reuters*-RCV1 dataset collects the Reuters News stories during a year from August 20, 1996 to August 19, 1997. It contains about 810,000 Reuters News stories. The average length of the stories is 200 tokens. The dataset is available at Reuters Corpus@ NIST[6].
- The Wikipedia articles dataset is obtained from Wikipedia dump system[7]. We preprocess all the html pages and get 5 million articles with pure texts.
- The paper dataset is acquired by downloading published papers from ACM Digital Library[8] and arXiv[9]. It contains 2,198 articles of various domains ranging from subjects such as computer science, physics and social science.

As the articles in *Reuters*-RCV1 dataset are short, there is no explicit structure in it. The section structures of articles in Wikipedia and paper dataset can be extracted using HTML tags and section information. The details about the three datasets are illustrated in table 1:

Table 1: Details of document datasets

Dataset	Size	Avg words	Structured
Reuters-RCV1	$810K$	453	No
Wikipedia	$5,000K$	$12.4K$	Yes
Paper	2,198	$24.9K$	Yes

Q&A Datasets: Our enrichment process relies on two well structured datasets.

- Quora dataset: more than 2,040,000 web pages are collected through our web-based crawler. A standard is set to crawl only the Q&A pairs with more than two answers each of which is longer than 50 words. Based on our enrichment approach, questions, topics as well as answers are crawled. The Q&A pairs are stored in our database. It includes 147,000+ unique topics, 2,040,000+ distinct questions and 9,271,000+ distinct answers. Further details of crawling process will be discussed in the next section.
- The StackExchange dataset is periodically released to the public[10]. It contains all the questions and answers of StackExchange communities.

5.2 Data Collection

We collect our Quora dataset by using a crawler with proper etiquette. Our generic crawler structure clearly defines the workload

[6] http://trec.nist.gov/data/reuters/reuters.html
[7] https://dumps.wikimedia.org/
[8] http://dl.acm.org
[9] http://arxiv.org
[10] https://archive.org/details/stackexchange

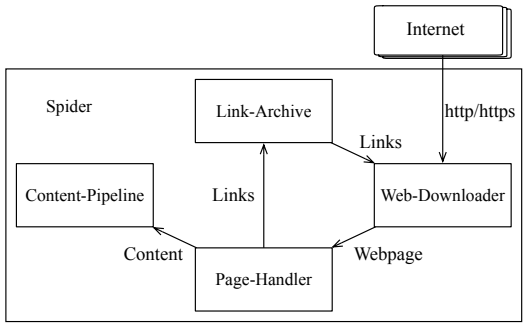

Figure 5: Architecture of Crawler

of each crawler unit. As shown in Figure 5, Web-Downloader is responsible to download the web pages from Internet according to links in the Link-Archive. Page-Handler extracts needed contents, discovers further requested pages and puts page links into the Link-Archive. Link-Archive is maintained as a queue structure. After the page contents are extracted, they are persisted in the database by the Content-Pipeline.

Quora does not define fixed topic structures for its questions. Moreover, and users can add any topic to categorize their questions. This makes it difficult to crawl informative contents according to topics provided in Quora Sitemap. Thus, we apply a new method to crawl the contents. We first put links of 100 popular questions that are evenly distributed in 20 most popular topics into the Link-Archive. New question links can be additionally obtained from the "Related Questions" section in each of question pages. Since answers have been listed in terms of their popularity, we crawl the answers according to their original sequence. Quora only presents limited number of answers and remaining answers are embedded in Ajax calls. Thus, we employ the Selenium downloader and simulate the scrolling and clicking actions of users in order to load and crawl the whole contents that are conformed to our crawling standard. BloomFilter is utilized to filter out duplicated contents in this circumstance.

5.3 Parameter Setting

We choose the optimal parameters based on the result of a validation set of 10% of the data. For text segmentation, the value of k varies according to the length of document. For keyword extraction, the weight λ is set to 0.4 in equation (5). The threshold for $TIE(w_i, w_j)$ is set to be 0.6. For equation (7) in retrieving Q&A contents, we use $\gamma_1 = 0.4$ and $\gamma_2 = 0.2$. And for the training of the neural network via SGD, $o = 8$ negative examples are selected for each question.The margin for the loss function (12) is set to be 0.35. And a learning rate of 0.1 is chosen. Besides otherwise stated, the same set of parameters is used in all experiments.

5.4 Text Segmentation Quantitative Analysis

In our approach, the proposed V-Optimal dynamic programming algorithm denoted as **DP** is used to construct the hierarchy of segments. The algorithm is compared with four state-of-the-art hierarchical text segmentation algorithms: **HC99** [10], **HCWM** [14], **HAC** [32], and **BiSeg** [28]. Our algorithm and those four algorithms are all domain independent segmentation algorithms which do not require building a segmentation model beforehand.

Table 2: On E_{WD}

	DP	HC99	HCWM	HAC	BiSeg
Wikipedia	**0.36**	0.48	0.45	0.57	0.56
Paper	**0.36**	0.46	0.42	0.52	0.52
Choi	**0.13**	0.20	0.18	0.32	0.34

Table 3: On E_{P_k}

	DP	HC99	HCWM	HAC	BiSeg
Wikipedia	**0.38**	0.46	0.44	0.52	0.50
Paper	**0.31**	0.37	0.34	0.43	0.45
Choi	**0.10**	0.15	0.14	0.29	0.28

Two commonly used evaluation metrics are adopted: E_{P_k} and E_{WD} proposed in [9]. The E_{P_k} is the mean of Beeferman errors calculated over the segmentation result in each level of the hierarchy [6]; while the E_{WD} is calculated using a sliding window according to the measure proposed in [25]. These two error rates are calculated as below:

$$E_{P_k} = \frac{1}{|J|} \sum_i c_i P_k(J_i, H_i)$$

$$E_{WD} = \frac{1}{|J|} \sum_i c_i WD(J_i, H_i)$$

where

$$J_i = \{b_j : b_j \in J \wedge j \leq i\}$$

The set J contains reference segment boundaries, i.e. the ground truth, while H contains algorithm output boundaries. Both J and H are in a ranked order. J_i is a subset of J with elements which rank before element with subscript i. Here, c_i is the predefined weight. In order to build the reference segment boundary set J, we extrct segment information on the two structured datasets: Wikipedia and Paper. We utilize the HTML tag information such as "h1" and "h2" in the Wikipedia dataset, and the heading title information like "abstract" and "introduction" in the Paper dataset, to construct the set J. Choi's dataset which is widely used in text segment evaluation [14][10] is also used. To evaluate, we plot out the total errors by accumulating the error on all the levels in the hierarchy. The comparison results on three datasets are demonstrated in Tables 2 & 3.

From the tables, we can observe that the DP algorithm achieves the best performance comparing to the existing methods. In particular, the DP algorithm significantly outperforms two classical hierarchical text segmentation methods: HAC and BiSeg. Those methods use greedy strategy to split the texts and cannot guarantee an optimal solution. Although HC99 and HCWM are better than HAC and BiSeg, they are still outperformed by our approach. These two methods iteratively run state-of-the-art linear algorithms and return the best segmentation they can find. Such methods are time-consuming while the optimality cannot be guaranteed. We think

one of the most important reasons that our approach reaches the best result is that it captures word relations by incorporating semantic word embeddings. Especially when the texts are short and the corresponding word vectors are very sparse, one-hot encoding for words which are applied in other algorithms tends to lose important information about word relations.

5.5 Qualitative Analysis on Results

This section provides an illustrative example of the complementary Q&A contents retrieved by our system. we show the top-5 retrieved Q&A contents for the following two text segments taken from the beginning of a paper [16].

> **Abstract:** We used the Support Vector Machines in a classification approach to 'beat the market'. Given the fundamental accounting and price information of stocks trading on the Australian Stock Exchange, we attempt to use SVM to identify stocks that are likely to outperform the market by having exceptional returns. The equally weighted portfolio formed by the stocks selected by SVM has a total return of 208% over a five years period, significantly outperformed the benchmark of 71%. We have also given a new perspective with a class sensitivity trade-off, whereby the output of SVM is interpreted as a probability measure and ranked, such that the stocks selected can be fixed to the top 25%.
>
> **Introduction:** Investors are usually faced with an enormous amount of stocks in the market. A crucial part of the their decision process is the selection of stocks to invest in. In a data-mining perspective, the problem of stock-selection aims to identify stocks with potential to outperform the market (i.e. exhibit exceptional returns) in the following year. Given the database of stock prices and indicators, it is a prediction problem that involves discovering useful patterns or relationship in the data, and applying that information to classify stocks.

After hierarchical text segmentation, our algorithm successfully clusters the text into two segments, i.e. abstract and introduction. The corresponding top 5 retrieved results are presented in Table 4, where the Q&A contents are ranked according to ranking scores.

For abstract, results (1), (2) and (4) are queries about applying SVM to stock prediction, which is exact the focus of the abstract paragraph. They provide more explanations and examples for the main document. Results (3) and (5) mainly focus on SVM, which provide useful background information about this paragraph. These top 5 results not only capture most of the important concepts mentioned in the abstract, but also provide questions that really help the readers to understand the text.

For the introduction part, the top 5 contents are quite different from those for abstract. The results focus on the topic of stock market and machine learning. This is closely related to the information mentioned in this segment. Results (2) (4) talk about stock market, while results (1) (3) mention applying machine learning and data mining techniques to stock data. It is worth noticing that though stock prediction is not mentioned in this segment, Q&A content like (5) is still retrieved. It shows the global relavency of our results.

Overall, the illustratative example intuitively demonstrates that our system is able to retrieve relevant Q&A content to a segment to help readers understand the document and inspire new thoughts.

Table 4: Top five retrieved Q&A contents

Abstract:
1. Andrew Ng: Is SVM good for developing model to stock market forecasting? SVM is indeed one of the methods to predict implied volatility of a stock, that's used as an ...
2. Which one of two techniques: ANN or SVM performs better for stock price prediction? Every action has an equal and opposite reaction – and every trade made to exploit ...
3. What is SVM? Can you explain the algorithm in detail and its application? Hard margin SVM: You're given a set of points labelled as Red and Blue. Let's say the ...
4. I want to implement a SVM based classifier for predicting stocks using data of micro blogging sites like Stocktwits. Where can I get the datasets? How can I implement ...
5. What is the best way to implement an SVM using Hadoop? Thought it might be worthwhile to put all the proposed solutions in context to see when one might us ...
Introduction:
1. What are some good, stock market data mining resources? The average day-over-day percentage price change of the S&P 500 is a descriptive statistic. All you need is ...
2. Who determines the price of a stock in a Stock Exchange? It is determined by supply and demand forces. That is not to say that the market price is the correct ...
3. How do financial companies use machine learning? As a scientist working with ML and knowing nothing about finance, I once tried to understand why mod ...
4. What is market capitalization and how does it help in judging the viability of investment in a company? Market capitalization is an important data point for investors ...
5. What is the best neural network architecture for stock market predictions? RNNs tend to connect/biased to previous information/states which when you think about it ...

5.6 Quantitative Analysis on Results

The difficulty of analyzing the results of our system has been experienced in similar works, such as attaching images to textbooks[2], enriching Wikipedia pages using tweets[21]. There is no standard benchmarks for us to compute metrics such as precision and recall. Thus, we choose to adopt a similar approach used in [2], i.e. to conduct a user study to evaluate the results of our system. The user study is performed on the survey website Surveymonkey[11].

To conduct user study, we randomly sample 600 documents respectively from the *Reuters*-RCV1, Wikipedia and Paper datasets. The documents are hierarchically segmented and candidate contents are retrieved for each segment using different algorithms. For each query, we provide the text segment, the link to complete

[11] http://www.surveymonkey.com

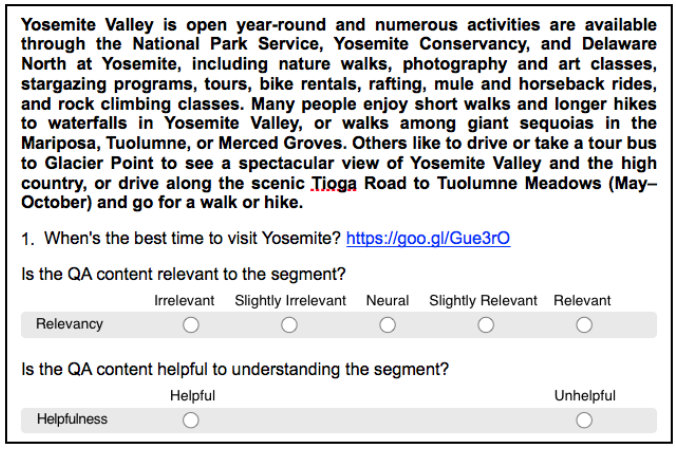

Figure 6: Survey on one query

original document, the top 5 related Q&A contents retrieved and their associated web links. Figure 6 illustrates a sample set of survey questions for a content retrieved.

Annotators are asked to read the text segment, check the original document, and then answer whether the results are relevant or not (scale from 0 to 4) and helpful or not in understanding the segment. A minimum time for a set of questions for one text segment is specified. It's calculated by the length of document and the average reading speed. Decisions made within the minimum time is not considered. Each query set is answered by 5 annotators. Before the experiments, annotators are required to answer a set of 30 pre-defined similar questions on relevancy and helpfulness. Only those achieve accuracy higher than 70% can proceed to join the user study.

Standard metrics are used in our experiments to compare the result generated by different algorithms. We employ NDCG [11] to measure relevancy since it can measure ranking performance on range relevancy. And helpfulness index proposed by [2] is adopted here to quantize helpfulness.

In order to validate the effectiveness of our approach, QALink is compared with the methods proposed in [2], [3] and various retrieval and ranking methods. For [2] and [3], we implement their methods to construct concept graph and generate queries for retrieval. Vector space model (Vector) and BM25 are used as two baseline methods. Standard parameters are employed to implement BM25, i.e. $\alpha = 1.2$ and $\beta = 0.75$. Translational modeling (TM) approach is solely using translational modeling without incorporating LSTM semantic ranking. We modify Equation (8) to use translational modeling on answer terms also. LTR [30] uses learning to rank to rank retrieved candidate contents. Ranklib[12] is used to implement the ListNet which is a representative learning to rank model. A number of features such as BM25 score, translational modeling score, popularity of Q&A pairs are used to train LTR. When building a dataset for ListNet, we adopt a similar approach as we mentioned in Section 4.3. We split the dataset into training and test datasets. After getting a satisfying accuracies on both training and test dataset, we use the trained model to perform the ranking and generate survey papers. QALink is our overall methods proposed in this paper. For fairness, we use our dynamic programming algorithm for text segmentation in all experiments.

[12]https://people.cs.umass.edu/vdang/ranklib.html

Table 5: NDCG for various methods.

Method	Reuter	Wiki	Paper
Vector (baseline)	0.493	0.534	0.512
BM25 (baseline)	0.562	0.581	0.575
TM	0.591	0.618	0.605
Image	0.601	0.620	0.610
DMI	0.610	0.615	0.627
LTR	0.625	0.659	0.654
QALink	**0.661**	**0.702**	**0.688**

Table 6: Helpfulness index on various methods.

Method	Reuter	Wiki	Paper
Vector(baseline)	0.571	0.563	0.582
BM25(baseline)	0.618	0.613	0.619
TM	0.652	0.667	0.689
Image	0.712	0.754	0.731
DMI	0.758	0.775	0.749
LTR	0.792	0.783	0.763
QALink	**0.837**	**0.839**	**0.813**

Table 5 and Table 6 present the results. As expected, all the methods outperforms the two baseline methods. After incorporating translational modeling ranking, the results improve upon baseline methods. Methods in [2] and [3] are able to beat translational modelling. By including answer semantic information using LSTM, our method outperforms the other algorithms.

From the experiments, we can conclude that **QALink** is able to capture contexts of documents and achieves global relevancy compared to [2] and [3] by utilizing the hierarchy structures. By ultilizing semantic information from main document and Q&A contents, our method can retrieve and rank the results better.

6 CONCLUSION

In this paper, we have proposed an interesting problem of enriching text documents with Q&A contents in order to enhance people's reading experiences. The linked Q&A contents should be relevant and helpful for readers to understand the documents. To maintain global relevancy and capture the hierarchical properties of documents, a novel hierarchical text segmentation based approach has been employed. Based on the hierarchies, we have extracted keywords using a graph based approach and hierarchically generated queries. BM25 has been used to quickly retrieve a candidate content list and we have combined translational modeling and neural embedding to rank the result list. Experiments have shown the effectiveness of our approach.

Our work opens up a new direction on enriching documents with relevant Q&A contents. The supplementary contents can help

readers understand the documents and provide useful new information about the documents. In future works, we will we will continue to enhance each step of the QALink framework. Taking query generation for example, we will look for a proper weights to balance global context and local keywords. Moreover, we will investigate how to enrich documents with other social media data such as Twitter and Facebook.

A PROOF OF GLOBAL OPTIMALITY FOR EQUATION 4

Lemma A.1. *The recursive formula $f^*(b,k)$ returns the global optimal solution for the text segmentation problem defined in section 4.1.*

Proof. To prove by contradiction. Assume that there is another segmentation $f'(b,k)$ achieves better global score, i.e.$f'(b,k) > f^*(b,k)$. Let l' be the start position of last segment of $f'(b,k)$, then

$$\begin{aligned} f'(b,k) &= f'(l'-1,k-1) + q(l',b) \\ &> \max_{1<l\leq b}\{f^*(l-1,k-1)+q(l,b)\} \\ &\geqslant f^*(l'-1,k-1)+q(l',b) \\ &\Rightarrow f'(l'-1,k-1) > f^*(l'-1,k-1) \end{aligned} \tag{14}$$

which contradicts the fact that $f^*(l'-1,k-1)$ is the optimal solution to cluster $l'-1$ partitions into $k-1$ segments. □

ACKNOWLEDGMENTS

This work was supported by the Singapore NRF under its IRC@SG Funding Initiative and administered by the IDMPO, at the SeSaMe Centre, under the project of Readpeer.

Xiaoli Wang was supported by the Education and Research Project of Young Teacher of Fujian, China under Grant No: JAT160003 and by NSFC under Grant No: 61702432

REFERENCES

[1] Hartigan John A and Wong Manchek A. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society* (1979), 100–108.
[2] Rakesh Agrawal, Sreenivas Gollapudi, Anitha Kannan, and Krishnaram Kenthapadi. 2011. Enriching Textbooks with Images. In *CIKM*. 1847–1856.
[3] Rakesh Agrawal, Sreenivas Gollapudi, Anitha Kannan, and Krishnaram Kenthapadi. 2012. Data Mining for Improving Textbooks. *SIGKDD Explor. Newsl.* (2012), 7–19.
[4] Rakesh Agrawal, Sreenivas Gollapudi, Anitha Kannan, and Krishnaram Kenthapadi. 2014. Similarity Search using Concept Graphs. In *CIKM*.
[5] Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing and Management* (2003), 45–65.
[6] Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical Models for Text Segmentation. *Mach. Learn.* (1999), 177–210.
[7] Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 222–229.
[8] Marc Bron, Bouke Huurnink, and Maarten de Rijke. 2011. Linking Archives Using Document Enrichment and Term Selection. In *TPDL*.
[9] Lucien Carroll. 2010. Evaluating Hierarchical Discourse Segmentation. *ACL* (2010).
[10] Freddy Y. Y. Choi. 2000. Advances in Domain Independent Linear Text Segmentation. In *NAACL*.
[11] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *SIGIR*.
[12] Inderjit S. Dhillon and Dharmendra S. Modha. 2001. Concept Decompositions for Large Sparse Text Data Using Clustering. *Mach. Learn.* (2001), 143–175.
[13] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *CIKM*. 1625–1628.
[14] Johanna Moore Freddy Choi, Peter Wiemer-Hastings. 2001. Latent semantic analysis for text segmentation. In *EMNLP*. 109–117.
[15] Evgeniy Gabrilovich and Shaul Markovitch. 2006. Overcoming the Brittleness Bottleneck Using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. In *AAAI*. 1301–1306.
[16] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).
[17] Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective Entity Linking in Web Text: A Graph-based Method. In *SIGIR*. 765–774.
[18] Marti A. Hearst. 1997. TextTiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.* (1997), 33–64.
[19] H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, and Torsten Suel. 1998. Optimal Histograms with Quality Guarantees. In *VLDB*. 275–286.
[20] Frederick Jelinek. 1980. Interpolated estimation of Markov source parameters from sparse data. *Pattern recognition in practice* (1980).
[21] Wei Kang, Anthony K. H. Tung, Wei Chen, Xinyu Li, and Qiyue Song. 2014. Trendspedia: An Internet observatory for analyzing and visualizing the evolving web. In *ICDE*.
[22] Marios Kokkodis, Anitha Kannan, and Krishnaram Kenthapadi. 2014. Assigning Videos to Textbooks at Appropriate Granularity. In *ACM LAS*.
[23] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*.
[24] Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking Documents to Encyclopedic Knowledge. In *CIKM*. 233–242.
[25] Lev Pevzner and Marti A. Hearst. 2002. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Comput. Linguist.* (2002), 19–36.
[26] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond.*
[27] Malcolm Slaney and Dulce Ponceleon. 2001. Hierarchical segmentation: finding changes in a text signal. In *SIAM Text Mining Workshop*. 6–13.
[28] Fei Song, William M. Darling, Adnan Duric, and Fred W. Kroon. 2011. An Iterative Approach to Text Segmentation. In *ECIR*. 629–640.
[29] Mikolov Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.
[30] Andrew Trotman. 2005. Learning to rank. *Information Retrieval* (2005), 359–381.
[31] Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. 2011. Linking Online News and Social Media. In *WSDM*.
[32] Yaakov Yaari. 1997. Segmentation of expository texts by hierarchical agglomerative clustering. *RANLP* (1997).