A Machine Learning Approach for Displaying Query Results in Search Engines

Tunga Güngör[1,2]

[1] Boğaziçi University, Computer Engineering Department, Bebek, 34342 İstanbul, Turkey
[2] Visiting Professor at Universitat Politècnica de Catalunya, TALP Research Center, Barcelona, Spain
gungort@boun.edu.tr

Abstract. In this paper, we propose an approach that displays the results of a search engine query in a more effective way. Each web page retrieved by the search engine is subjected to a summarization process and the important content is extracted. The system consists of four stages. First, the hierarchical structures of documents are extracted. Then the lexical chains in documents are identified to build coherent summaries. The document structures and lexical chains are used to learn a summarization model by the next component. Finally, the summaries are formed and displayed to the user. Experiments on two datasets showed that the method significantly outperforms traditional search engines.

1 Introduction

A search engine is a web information retrieval system that, given a user query, outputs brief information about a number of documents that it thinks relevant to the query. By looking at the results displayed, the user tries to locate the relevant pages. The main drawback is the difficulty of determining the relevancy of a result from the short extracts. The work in [14] aims at increasing the relevancy by accompanying the text extracts by images. In addition to important text portions in a document, some images determined by segmenting the web page are also retrieved. Roussinov and Chen propose an approach that returns clusters of terms as query results [12]. A framework is proposed and its usefulness is tested with comprehensive experiments.

Related to summarization of web documents, White et al. describe a system that forms hierarchical summaries. The documents are analyzed using DOM (document object model) tree and their summaries are formed. A similar approach is used in [10] where a rule-based method is employed to obtain the document structures. Sentence weighting schemes were used for identification of important sentences [6,9,16]. In a study, the "table of content"-like structure of HTML documents was incorporated into summaries [1]. Yang and Wang developed fractal summarization method where generic summaries are created based on structures of documents [15]. These studies focus on general-purpose summaries, not tailored to particular user queries. There exist some studies on summarization of XML documents. In [13], query-based summarization is used for searching XML documents. In another study, a machine learning approach was proposed based on document structure and content [2]. The concept of lexical chains was also used for document summarization. Berker and Güngör used lexical chaining as a feature in summarization [3]. In another work, a lexical chain formation algorithm based on relaxation labeling was proposed [5]. The sentences were selected according to some heuristics.

In this paper, we propose an approach that displays the search results as long extracts. We build a system that creates a hierarchical summary for each document retrieved. The cohesion in the summaries is maintained by using lexical chains. The experiments on standard query sets showed that the methods significantly outperform the traditional search engines and the lexical chain component is an important factor.

2 Proposed Summarization Framework

The architecture of the summarization framework is shown in Fig. 1. The system is formed of four main components. The first component is structure extractor where the document structure is analyzed and converted into a hierarchical representation. The lexical chain builder processes the document content using WordNet and forms the lexical chains. Model builder employs a learning algorithm to learn a summarization model by using the structures and lexical chains. Finally, the summarizer forms the summaries of the documents in a hierarchical representation using the learned model.

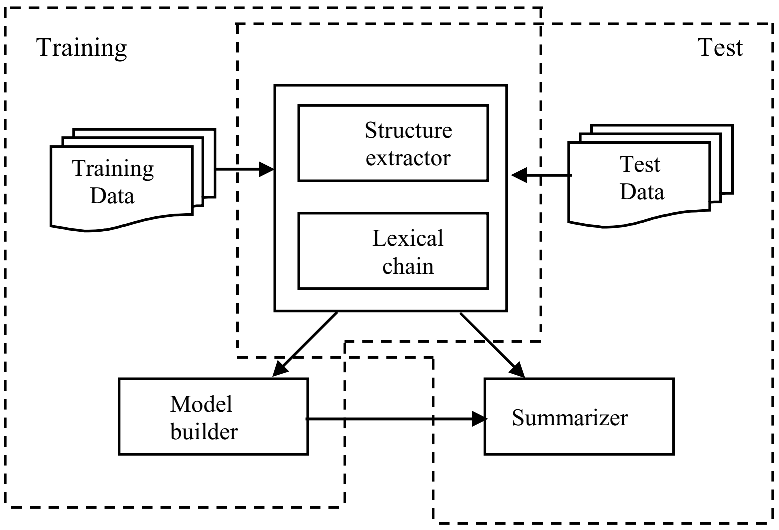
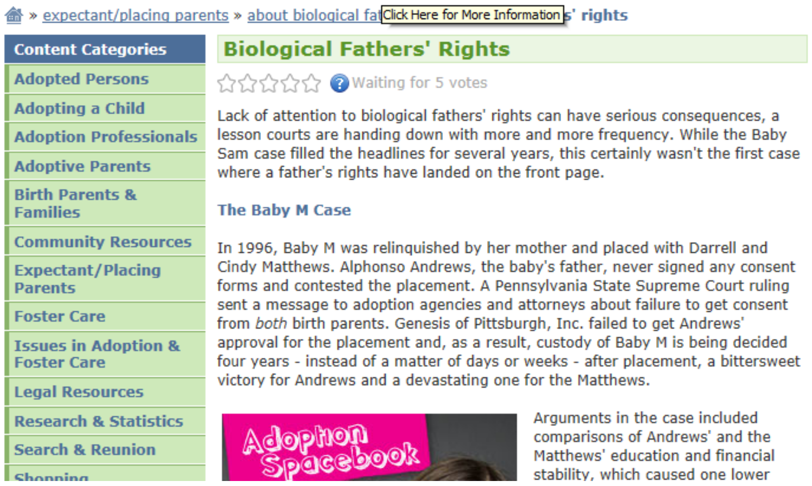


Fig. 1. System architecture

2.1 Extracting Structures of Documents

We simplify the problem of document structural processing by dividing the whole process into a number of consecutive steps. Fig. 2 shows an example web document that includes different types of parts. The first process is extracting the underlying content of the document. We parse a given web document and build its DOM tree using the Cobra open source toolkit [4]. We then remove the nodes that contain non-textual content by traversing the tree. After this process, we obtain a tree that includes only textual elements in the document and the hierarchical relations between them. The result of the simplification process for the document in Fig. 2 is shown in Fig. 3.



» expectant/placing parents » about biological fat Click Here for More Information s' rights

Content Categories
- Adopted Persons
- Adopting a Child
- Adoption Professionals
- Adoptive Parents
- Birth Parents & Families
- Community Resources
- Expectant/Placing Parents
- Foster Care
- Issues in Adoption & Foster Care
- Legal Resources
- Research & Statistics
- Search & Reunion

Biological Fathers' Rights

Waiting for 5 votes

Lack of attention to biological fathers' rights can have serious consequences, a lesson courts are handing down with more and more frequency. While the Baby Sam case filled the headlines for several years, this certainly wasn't the first case where a father's rights have landed on the front page.

The Baby M Case

In 1996, Baby M was relinquished by her mother and placed with Darrell and Cindy Matthews. Alphonso Andrews, the baby's father, never signed any consent forms and contested the placement. A Pennsylvania State Supreme Court ruling sent a message to adoption agencies and attorneys about failure to get consent from *both* birth parents. Genesis of Pittsburgh, Inc. failed to get Andrews' approval for the placement and, as a result, custody of Baby M is being decided four years - instead of a matter of days or weeks - after placement, a bittersweet victory for Andrews and a devastating one for the Matthews.

Adoption Spacebook

Arguments in the case included comparisons of Andrews' and the Matthews' education and financial stability, which caused one lower

Fig. 2. An example web document

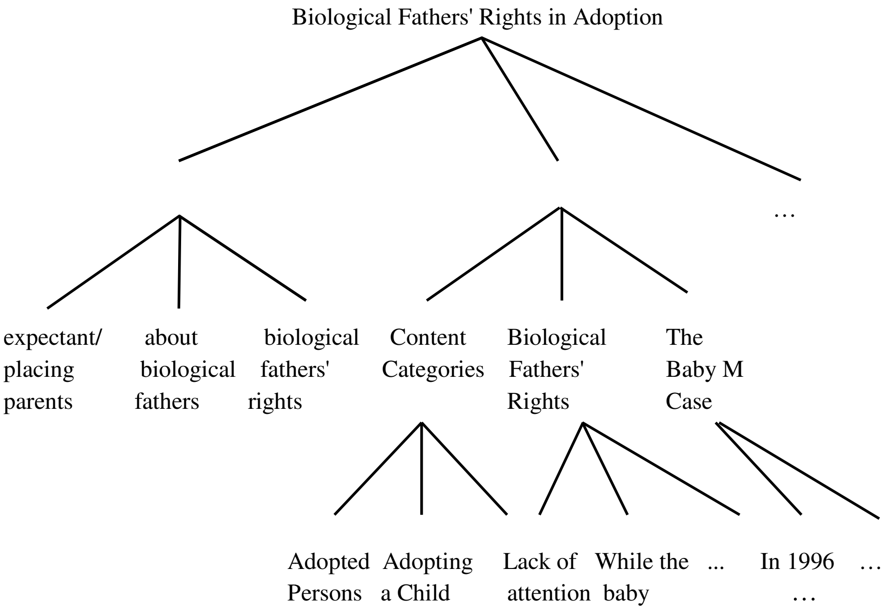


Fig. 3. Simplified DOM tree

The tree structure obtained does not correspond to the actual hierarchical structure. We identify the structure in three steps by using a learning approach. We consider the document structure as a hierarchy formed of sections where each section has a title (heading) and includes subsections. The first step is identification of headings which is a binary classification problem (heading or non-heading). For each textual unit in a document, we use the features shown in Table 1. The second step is determining the hierarchical relationships between the units. We first extract the parent-child relations between the heading units. This is a learning problem where the patterns of heading–subheading connections are learned. During training, we use the actual parent-child connections between headings as positive examples and other possible parent-child connections as negative examples. We use the same set of features shown in Table 1. In the last step, the non-heading units are attached to the heading units. For this purpose, we employ the same approach used for heading hierarchy.

Table 1. Features used in the machine learning algorithm

Heading	<h1> – <h6>
Emphasis	<b>, <i>, <u>, <em>, <strong>, <big>, <small>
Font	<font size>, <font color>, <font face>
Indentation	<center>
Anchor	<a>, <link>
Length	length of the unit (in characters)
Sentence count	number of sentences in the unit
Punctuation	punctuation mark at the end of the unit
Coordinate	x and y coordinates of the unit

3 Identification of Lexical Chains

A lexical chain is a set of terms that are related to each other in some context. We make use of lexical chains in addition to other criteria for summarization. We process all the documents and construct a set of lexical chains from the documents' contents. The idea is forming the longest and strongest chains so that the important relations between different parts of the documents can be captured. To determine the chains, we process the terms by using WordNet [11] to identify different term relations. We consider only the nouns. The documents are parsed using a part-of-speech tagger (http://alias-i.com/lingpipe). The relations we consider are the synonymy, hypernymy, and hyponymy relations. The words are processed and a word is placed in the lexical chain that holds the most number of words with a relation with the candidate word. After the lexical chains are formed, each chain is given a score as follows:

$$score(C) = \sum_{\substack{t_i, t_j \in C,\\ i<j}} tscore(t_i, t_j), \; where \tag{1}$$

$$tscore(t_i, t_j) = \begin{cases} value_s \;, & if\ t_i\ and\ t_j\ are\ synonyms \\ value_h \;, & if\ t_i\ is\ a\ hypernym/hyponym\ of\ t_j \\ value_{hh} \;, & if\ t_i's\ and\ t_j's\ hypernyms/hyponyms\ are\ the\ same \\ 0 \;, & otherwise \end{cases}$$

The chain score is calculated by summing up the scores of all pairs of terms in the chain. The score of terms t_i and t_j depends on the relation between them. We use a fixed value for each relation type. As the chains are scored, we select only the strongest chains for summarization. A lexical chain is accepted as a strong chain if its score is more than two standard deviations from the average of lexical chain scores.

4 Learning Feature Weights and Summarization of Documents

In this work, we aim at producing summaries that take into account the structure of web pages and that will be shown to the user as a result of a search query. We use the criteria shown in Table 2 for determining the salience of sentences in a document. For each feature, the table gives the feature name, the formula used to calculate the feature value for a sentence S, and the explanation of the parameters in the formula. The score values of the features are normalized to the range [0,1]. We learn the weight of each feature using a genetic algorithm. As the feature weights are learned, the score of a sentence can be calculated by the equation

$$score(S) = w_l score_l + w_{tf} score_{tf} + w_h score_h + w_q score_q + w_{lc} score_{lc} \quad (2)$$

where w_i denotes the weight of the corresponding feature. Given a document as a result of a query, the sentences in the document are weighted according to the learned feature weights. The summary of the document is formed using a fixed summary size. While forming the summary, the hierarchical structure of the document is preserved and each section is represented by that number of sentences that is proportional to the importance of the section (total score of the sentences in the section).

5 Experiments and Results

To evaluate the proposed approach, we identified 30 queries from the TREC (Text Retrieval Conference) Robust Retrieval Tracks for the years 2004 and 2005. Each query was given to the Google search engine and the top 20 documents retrieved for each query were collected. Thus, we compiled a corpus formed of 600 documents. The corpus was divided into training and test sets with a 80%-20% ratio.

For structure extraction, we used SVM-Light which is an efficient algorithm for binary classification [8]. The results are given in Table 3. Accuracy is measured by dividing the number of correctly identified parent-child relationships to the total number of parent-child relationships. The first row of the table shows the performance of the proposed method. This figure is computed by considering each pair of nodes independent of the others. A stronger success criterion is counting a connection to a node as a success only if the node is in the correct position in the tree. The accuracy under this criterion is shown in the second row, which indicates that the method identified the correct path in most of the cases. The third result gives the accuracy when only the heading units are considered. That is, the last step of the method explained in Section 2 was not performed.

Table 2. Sentence features used in the summarization algorithm

Feature	Feature equation and parameter explanations
Sentence location	$score_l(S) = \frac{maxs}{sdepth} * \frac{1}{spos}$ *maxs*: maximum score that a sentence can get *sdepth*: section depth *spos*: location of the sentence within the section
Term frequency-inverse document frequency (tf-idf)	$score_{tf}(S) = \sum_{t_i \in S} tf(t_i) * log \frac{dnum}{dnum_{t_i}}$ $tf(t_i)$: term frequency of term t_i in the document *dnum*: number of documents in the corpus $dnum_{ti}$: number of documents that t_i occurs
Heading	$score_h(S) = \sum_{t_i \in S} hterm(t_i)$ $hterm(t_i)$: 1 or 0 depending on whether t_i, respectively, occurs or does not occur in the corresponding heading.
Query	$score_q(S) = \sum_{t_i \in S} qterm(t_i)$ $qterm(t_i)$: 1 or 0 depending on whether t_i, respectively, occurs or does not occur in the query
Lexical chain	$score_{lc}(S) = \sum_{t_i \in S, \; t_i \in strong\ chains} tf(t_i)$ $tf(t_i)$: term frequency of t_i in the document

We use the document structures identified by the structure extractor component in the summarization process. As lexical chains are formed, we use genetic algorithm to learn the weights of the features. 50 documents selected randomly from the corpus were summarized manually using a fixed summary length. The feature weights were allowed to be in the range of 0-15. After training, we obtained the feature weights as w_l=5, w_{rf}=7, w_h=8, w_q=12, and w_{lc}=11. This shows that query terms are important in determining the summary sentences. The lexical chain concept is also an important tool for summarization. This is probably due to the combining effect of lexical chains in the sense that they build a connection between related parts of a document and it is preferable to include such parts in the summary to obtain coherent summaries.

Table 3. Results of structure extractor

	Accuracy
Document structure	76.47
Document structure (full path)	68.41
Sectional structure	78.11

As the feature weights were determined, we formed the summaries of all the documents in the corpus. For evaluation, instead of using a manually prepared summarization data set, we used the relevance prediction method [9] adapted to a search engine setting. In this method, a summary is compared with the original document. If the user evaluates both of them as relevant or irrelevant to the search query, then we consider the summary as a successful summary.

The evaluation was performed by two evaluators. For a query, the evaluator was given the query terms, a short description of the query, and a guide that shows which documents are relevant results. The evaluator is shown first the summaries of the 20 documents retrieved by the search engine for the query in random order and then the original documents in random order. The user is asked to mark each document or summary displayed as relevant or irrelevant for the query. The results are given in Table 4. We use precision, recall and f-measure for the evaluation as shown below:

$$precision = \frac{|D_{rel} \cap S_{rel}|}{|S_{rel}|} \tag{3}$$

$$recall = \frac{|D_{rel} \cap S_{rel}|}{|D_{rel}|} \tag{4}$$

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{5}$$

where D_{rel} and S_{rel} denote, respectively, the set of documents and the set of summaries relevant for the query.

The first row in the table shows the performance of the method, where we obtain about 80% success rate. The second row is the performance of the Google search engine. We see that the outputs produced by the proposed system are significantly better than the outputs produced by a traditional search engine. This is due to the fact that when the user is given a long summary that shows the document structure and the important contents of the document, it becomes easier to determine the relevancy of the corresponding page. Thus we can conclude that the proposed approach yields an effective way in displaying the query results for the users.

Table 4. Results of the summarization system

	precision	recall	F-measure
Proposed method	80.76	78.17	79.44
Search engine	63.57	58.24	60.79

6 Conclusions

In this paper, we built a framework for displaying web pages retrieved as a result of a search query. The system makes use of the document structures and the lexical chains extracted from the documents. The contents of web pages are summarized according to the learned model by preserving the sectional layouts of the pages. The experiments on two query datasets and a corpus of documents compiled from the results of the queries showed that document structures can be extracted with 76%

accuracy. In the summarization experiments, we obtained nearly 80% success rates. A comparison with a state-of-the-art search engine has shown that the method significantly outperforms the performance of current search engines.

As a future work, we plan to improve the summarizer component by including new features that can determine the saliency of sentences more effectively. Some semantic features that take into account dependencies between sentences can be used. The methods used in the proposed framework such as structural analysis and lexical chain identification can also be utilized in other related areas. Another future work can be making use of these methods in multi-document summarization or text categorization.

References

1. Alam, H., Kumar, A., Nakamura, M., Rahman, A.F.R., Tarnikova, Y., Wilcox, C.: Structured and Unstructured Document Summarization: Design of a Commercial Summarizer Using Lexical Chains. In: Proc. of the 7th International Conference on Document Analysis and Recognition, pp. 1147–1150 (2003)
2. Amini, M.R., Tombros, A., Usunier, N., Lalmas, M.: Learning Based Summarisation of XML Documents. Journal of Information Retrieval 10(3), 233–255 (2007)
3. Berker, M., Güngör, T.: Using Genetic Algorithms with Lexical Chains for Automatic Text Summarization. In: Proc. of the 4th International Conference on Agents and Artificial Intelligence (ICAART), Vilamoura, Portugal, pp. 595–600 (2012)
4. Cobra: Java HTML Renderer & Parser (2010), `http://lobobrowser.org/cobra.jsp`
5. Gonzàlez, E., Fuentes, M.: A New Lexical Chain Algorithm Used for Automatic Summarization. In: Proc. of the 12th International Congress of the Catalan Association of Artificial Intelligence (CCIA) (2009)
6. Guo, Y., Stylios, G.: An Intelligent Summarisation System Based on Cognitive Psychology. Information Sciences 174(1-2), 1–36 (2005)
7. Hobson, S.P., Dorr, B.J., Monz, C., Schwartz, R.: Task-based Evaluation of Text Summarisation Using Relevance Prediction. Information Processing and Management 43(6), 1482–1499 (2007)
8. Joachims, T.: Advances in Kernel Methods: Support Vector Learning. MIT (1999)
9. Otterbacher, J., Radev, D., Kareem, O.: News to Go: Hierarchical Text Summarisation for Mobile Devices. In: Proc. of 29th Annual ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 589–596 (2006)
10. Pembe, F.C., Güngör, T.: Structure-Preserving and Query-Biased Document Summarisation for Web Searching. Online Information Review 33(4) (2009)
11. Princeton University, About WordNet (2010), `http://wordnet.princeton.edu`
12. Roussinov, D.G., Chen, H.: Information Navigation on the Web by Clustering and Summarizing Query Results. Information Processing and Management 37 (2001)
13. Szlavik, Z., Tombros, A., Lalmas, M.: Investigating the Use of Summarisation for Interactive XML Retrieval. In: Proc. of ACM Symposium on Applied Computing (2006)
14. Xue, X.-B., Zhou, Z.-H.: Improving Web Search Using Image Snippets. ACM Transactions on Internet Technology 8(4) (2008)
15. Yang, C.C., Wang, F.L.: Hierarchical Summarization of Large Documents. Journal of American Society for Information Science and Technology 59(6), 887–902 (2008)
16. Yeh, J.Y., Ke, H.R., Yang, W.P., Meng, I.H.: Text Summarisation Using a Trainable Summariser and Latent Semantic Analysis. Information Processing and Management 41(1), 75–95 (2005)