Summarizing Figures, Tables and Algorithms in Scientific Publications to Augment Search Results

Sumit Bhatia and Prasenjit Mitra
The Pennsylvania State University

Increasingly, special-purpose search engines are being built to enable the retrieval of document-elements like tables, figures, and algorithms[Bhatia et al. 2010; Liu et al. 2007; Hearst et al. 2007]. These search engines present a thumbnail view of document-elements, some document metadata such as the title of the papers and their authors, and the caption of the document-element. While some authors in some disciplines write carefully tailored captions, generally, the author of a document assumes that the caption will be read in the context of the text in the document. When the caption is presented out-of-context as in a document-element-search-engine result, it may not contain enough information to help the end-user understand what the content of the document-element is. Consequently, end-users examining document-element search results would want a short "synopsis" of this information presented along with the document- element. Having access to the synopsis allows the end-user to quickly understand the content of the document-element without having to download and read the entire document as examining the synopsis takes a shorter time than finding information about a document element by downloading, opening and reading the file. Furthermore, it may allow the end-user to examine more results than they would otherwise. In this paper, we present the first set of methods to extract this useful information (synopsis) related to document-elements automatically. We use Naïve Bayes and support vector machine classifiers to identify relevant sentences from the document text based on the similarity and the proximity of the sentences with the caption and the sentences in the document text that refer to the document-element. We compare the two classification methods and study the effects of different features used. We also investigate the problem of choosing the optimum synopsis-size that strikes a balance between the information content and the size of the generated synopses. A user study is also performed to measure how the synopses generated by our proposed method compare with other state-of-the-art approaches.

Categories and Subject Descriptors: H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval ; H.3.6 [**INFORMATION STORAGE AND RETRIEVAL**]: Library Automation; H.3.7 [**INFORMATION STORAGE AND RETRIEVAL**]: Digital Libraries; H.5.2 [**INFORMATION INTERFACES AND PRESENTATION**]: User Interfaces

General Terms: Algorithms, Experimentation.

Additional Key Words and Phrases: classification, document-element, summarization, synopses.

Author's address: Sumit Bhatia, Department of Computer Science and Engineering, The Pennsylvania State University, PA-16802, USA. E-mail:sumit@cse.psu.edu
Prasenjit Mitra, College of Information Sciences and Technology, The Pennsylvania State University, PA-16802, USA. E-mail:pmitra@ist.psu.edu

1. INTRODUCTION

Authors use a number of document-elements for a variety of purposes like reporting and summarizing experimental results (plots, tables), describing a process (flow charts) or presenting an algorithm (pseudo-code). A *document-element* is defined as an entity, separate from the running text of the document, that either augments or summarizes the information contained in the running text. Figures, tables and pseudo-codes for algorithms are the most commonly used document-elements in scientific literature and are sources of valuable information. For example, in biology, figures and their related textual descriptions account for as much as 50% of a whole paper [Futrelle 2004]. In Table I, we show the numbers of different document-elements present in all the papers published in last five years in some major Computer Science conferences[1]. From these examples it can be observed that document-elements constitute a substantial part of scientific literature. Oftentimes, the most important experimental results and ideas in an article are presented using non-textual document-elements.

Recently, significant efforts have been made to utilize and extract information present in document-elements. Science Direct[2] offers a "Figures/Tables" preview feature for many of the articles in its database. CiteSeerX[3], a major computer science digital library, has introduced a table search feature in addition to normal document search. *TableSeer* [Liu et al. 2007], a specialized search engine allows end users to search for tables in digital documents. We have proposed a search engine for finding algorithms in scientific articles [Bhatia et al. 2010]. Likewise, a specialized search engine for biology documents, *BioText Search Engine* [Hearst et al. 2007], offers end-users the capability to search for figures and tables in the documents.

A study by Sandusky and Tenopir [2008] on the usefulness of such document-element search engines reveals that while these specialized search engines help the users in identifying quickly the documents relevant to their information needs, users generally find it hard to completely understand the document-elements presented to them without examining the context in which they were used in the associated document. Demner-Fushman et al. [2009] also discuss the need to augment image captions in papers with related text so as to help readers understand the image in its right context. The special-purpose document-element search engines described above generally return a list of document-elements and a snippet constructed from the documents to provide this contextual information. In most cases, document-element captions and sentences in the document that mention the document-element are used as snippets. In our interaction with computer scientists, chemists, and environmental geo-scientists, we have observed that the end-user often wants to examine more information than is available in the snippets because he or she can not always interpret the information content of document-elements by examining just the snippets as illustrated by Figure 1, which shows a figure along with its associated caption. The end-user would find it hard to interpret results just by looking at a figure because the figure does not contain the full information

[1]These numbers were obtained by parsing methods as described in section 3.
[2]http://www.sciencedirect.com
[3]http://www.citeseerx.ist.psu.edu

Conference	No. of Papers	Figures		Tables		Algorithms	
		Total	Average	Total	Average	Total	Average
SIGIR	925	1990	2.15	1916	2.07	75	0.08
SIGMOD	608	4303	7.08	688	1.13	301	0.5
STOC	406	466	1.15	34	0.08	74	0.18
VLDB	538	5198	9.66	788	1.46	287	0.53
WWW	957	3735	3.9	1429	1.49	142	0.15

Table I. Distribution of different document-elements in different conferences in last five years (2005–2009).

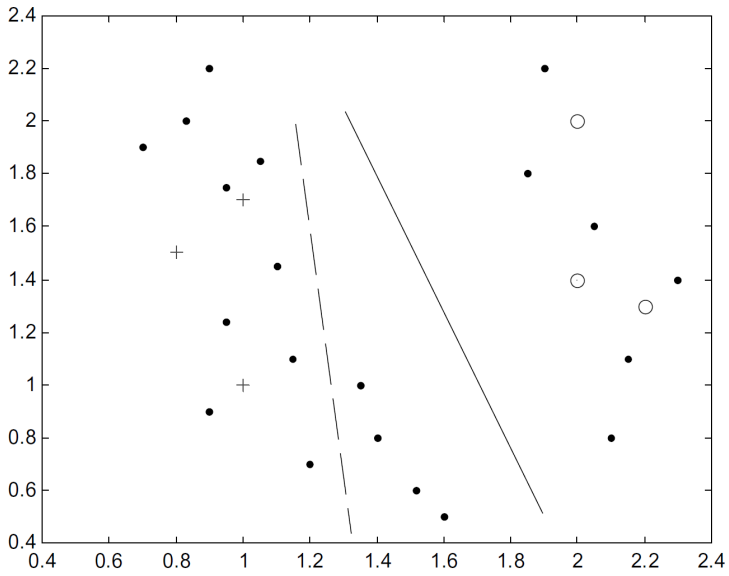


Fig. 3. Results comparison of TSVM and PTSVM.

Fig. 1. A sample figure and its caption. Figure courtesy [Chen et al. 2003].

about what the lines and different points in it actually mean. Even though the associated caption and legend help in understanding the information presented in a figure, they hardly provide enough details to fully understand and interpret the figure.

As another use case, consider a chemist, who is searching for experimental results for the dissolution rate of kaolinite under certain conditions. The chemist wants to quickly find the results that were reported by earlier papers and compare his results with previously reported results. He would want to quickly search the digital library for documents, find the figures or tables that present the results, collect the results, extract the data and then analyze it. While searching in a large digital library that presents the chemist with a ranked list of documents along with the associated

figures and tables outlining experimental results, the chemist would like to avoid false positives. That is, the chemist would not want to open the documents that contain figures and tables reporting results that he or she does not exactly need. Can we provide this end-user with a basic understanding of a non-textual document element by summarizing the textual information describing the document element and presenting it succinctly to the end-user? If we can do so, the end-user would not have to examine a number of documents by opening them. Thus, such a summary would be of great use in reducing the work that needs to be done by the information-seeker. This additional information can be provided in a manner similar to what is used by state of the art modern search engines like MSN Bing[4]. When a user hovers over a search result, the synopsis can be shown in a pop-up box. Thus, this additional information can be provided without compromising the valuable real estate on the screen.

In this work, we show a way to automatically extract information related to document-elements from document text. We refer to this extracted information as a *synopsis*. Availability of a concise and relevant synopsis may help save the end-users' time when they are examining search results to find something that satisfies their information needs. In Figure 2, we show the synopsis generated by our method for the figure shown in Figure 1. The semantics of Figure 1 becomes much clearer upon reading the synopsis. Thus, our tool increases the degree of automation of information seeking and improves the productivity of end-users.

Extracting a synopsis for a document-element from a digital document involves filtering information related to the document-element from the rest of the document. Solving this problem accurately is easy if we understand the semantics of the text automatically. However, state-of-the-art techniques of natural language processing and statistical text processing still fall short in fully understanding the semantics of text documents. Additionally, good synopsis generation involves making a judgment call regarding the level of detail that may be useful to an end-user. If we generate a very large synopsis, it will be comprehensive, but the users' needs of finding information *quickly* will not be met. If we generate a very short synopsis, the user will not understand the document-elements clearly. We aim at striking a balance between these conflicting needs using automated synopsis-generation methods.

Our algorithm finds the reference text in the document, i.e., text that refers to document-elements, e.g., "In Figure 3, we show . . .", and then assigns scores to other sentences in the text depending upon their (a) similarity to the reference text and caption and (b) proximity to the reference text. A synopsis is then constructed out of these sentences based on a cost-benefit analysis that depends upon the marginal utility and marginal cost of including a sentence in a partially constructed synopsis.

Our work has the following key contributions.

—We propose a method for extracting document-element related information from digital documents automatically. We treat the problem as a special case of query-biased summarization where the document-element itself is the query. We adopt machine learning techniques and develop a novel feature set for identifying

[4]http://www.bing.com/

Fig. 3 illustrates the training results of TSVM and PTSVM on Tutorial dataset. The solid line is the final hyperplane found by PTSVM and the dashed line is the final hyperplane found by TSVM. As shown in Fig. 3, the wrong estimation for value of N is responsible for bad performance of TSVM. This problem is successfully avoided in PTSVM. We can also find out that the training time of PTSVM is much shorter than that of TSVM. This is mainly due to the fact that TSVM need to successively increase the value of C and calculation has to be done for every C value.

Fig. 2. Information extracted by our method for the figure described in Figure 1.

document-element related sentences.

—We propose a simple model for sentence selection that tries to strike a balance between the information content and the length of the synopsis. The top-ranked sentences selected by this model are finally included in the synopsis.

—We validate our proposed methods using empirical evaluation.

The rest of the paper is organized as follows. Section 2 provides an overview of the related work. Section 3 describes the methods used to identify relevant sentences from the documents and various features and classification methods used. Section 4 explains the sentence selection strategy for variable-length synopses generation. Section 5 describes the dataset used, experiments performed and discussion of results. Section 6 concludes the paper.

2. RELATED WORK

2.1 Work related to document-elements

The operational methodology and experimental results in a scientific paper are often elaborated using document-elements like figures and tables. There has been some work aimed at extracting this useful information from document-elements. Kataria et al. [2008] use image processing and Optical Character Recognition techniques for automatic extraction of data points and text blocks from 2-D plots. This extracted information can then be indexed and made available through a search interface to the end user. Liu et al. [2007] describe *TableSeer*, a search engine for tables in digital documents. They propose algorithms for automatic detection of tables in digital documents and table metadata extraction. A tailored vector-space model based ranking algorithm, *TableRank* is used to rank the search results. A specialized search engine for biology documents, *BioText Search Engine*, offers the capability to search for figures and tables [Hearst et al. 2007]. Similarly, the CiteSeerX digital library also offers a table search functionality[5]. However, none of these systems can actually *summarize* the information contained in a document-element. Often they do not provide enough textual information to help the end-user determine the relevance of a particular table or figure to her information needs.

Futrelle [1999] introduces the idea of diagram summarization and explores various related issues and problems. He advocates the use of the internal structure of the diagrams as well as the text in captions, running text and the diagrams themselves. The relationship between the text and the graphics is analyzed and

[5]http://citeseerx.ist.psu.edu/

the importance of captions and the reference text in diagram understanding is emphasized. Huang et al. [2005] utilize textual and graphical information in scientific charts to understand the underlying semantics. They do not, however, consider the important information present in the running text of a document. We also found previous work that uses figure captions as a guiding tool for extracting information about the figures. Guglielmo et al. [1996] report the use of figure captions for image retrieval systems. Passonneau et al. [1996] proposed a system that generates summaries of work flow diagrams but their approach is very specific in nature as it requires as input a set of flat attribute-value list representations describing all the information about the diagrams. No efforts, however, have been made to extract information related to document-elements from the document text. This information can greatly increase the understanding of document-elements without requiring the end-user to expend the time to read the entire document.

2.2 Summarization by Sentence Extraction

Automatic text summarization has been a long-standing and well-studied problem of interest to researchers in natural language processing, artificial intelligence and information retrieval [Mani and Maybury 1999]. Most summarization methods fall under two categories: Query Independent Summarization (QIS) and Query Dependent or Query-Biased Summarization (QBS). QIS techniques are static in nature and focus on producing generic summaries that act as surrogates for the original document [Luhn 1958; Kupiec et al. 1995]. QBS techniques on the other hand, are dynamic in nature and focus on producing query-specific summaries of the documents. QBS may therefore generate different summaries for the same document in response to different queries. This property is very useful in information retrieval and web search, because it provides a better means of gauging the relevance of search results [White et al. 2003]. Almost all modern web-based search engines display short text snippets along with the search results. These snippets are usually generated using *query-biased* summarization techniques. Selected portions of text are extracted based on their semantic closeness to the query. As identified by Tombros and Sanderson [1998], query-biased snippets alleviate the need to refer to the whole document text and help the user perform relevance judgments more quickly and accurately.

Sentence extraction has been one of the most popular techniques for automatic text summarization. It is useful for single and multi-document summarization as well as for query independent and query-biased summarization [Luhn 1958; Kupiec et al. 1995; Teufel and Moens 1997; Goldstein et al. 1999; Goldstein et al. 2000; Ko and Seo 2008; Metzler and Kanungo 2008]. Kupiec et al. [1995], were among the first to consider summarization as a statistical classification problem. They use a Naïve Bayes classifier for ranking sentences based on the probability of their being a part of the summary. Teufel and Moens [1997] have replicated their method and experimented with different datasets. Recently, Metzler and Kanungo [2008] have evaluated different machine-learning based sentence selection techniques for query-biased summarization. Using standard TREC test collections, they have evaluated Ranking SVMs, Support Vector Regression (SVR) and Gradient Boosted Decision Trees (GBDT) for the sentence-selection problem. Their results show that the effectiveness of machine learning approaches varies across collections with different

characteristics. All the above techniques, however, are related to either single or multi-document summarization. They are not concerned with document-elements.

In our previous work [Bhatia et al. 2009], we have described briefly the features that can be used for identifying sentences that summarize the content of a document-element and some results on a small dataset. In this work, we describe our approach in detail to enable reproduction, and provide additional experimental evaluation.

3. IDENTIFYING DOCUMENT-ELEMENT RELATED INFORMATION

In this section, we describe strategies to automatically identify information related to document-elements . We treat this problem as a classification task - each sentence is either relevant or non-relevant for a document-element. We describe the classification methods used and associated features used below.

3.1 Pre-processing

Text Extraction: A majority of the files in modern digital libraries are PDF files. All the files in our dataset are also in PDF format and thus, need to be converted to text format for further processing. We tried several tools available for PDF to text conversion (PDFBox[6], PDFTextStream[7], XPDF[8] and TET[9]) and found PDFTextStream to be the most suitable for our purpose. It performed best at preserving the sequence of text streams in the order they appeared in the document, especially for documents in double column format that are common in scientific literature. The text thus obtained is processed to extract document-element related information.

Document-Element Caption Parsing: Captions contain useful information cues that help in understanding the content of a document-element. A well-crafted caption explains the contents of a document-element well. Corio and Lapalme have studied a corpus of more than 400 documents and found that captions and figures complement each other and are incapable of transmitting the intended message completely when used in isolation from each other [Corio and Lapalme 1999]. Thus, extracting document-element captions is the first logical step in our algorithm. In order to deal with variations in the caption format across different domains and writing styles, we propose a grammar to distinguish and extract caption sentences from the rest of the sentences (see Figure 3).

The CAPTION non-terminal in this grammar has 4 sub-parts. DOC_EL_TYPE specifies the type of the document element, namely figure, table or algorithm. FIG_TYPE, TABLE_TYPE and ALGO_TYPE refer to the variations of the words "Figure", "Table" and "Algorithm" respectively, as they occur in the captions. The DOC_EL_TYPE non-terminal is followed by an integer that represents the document-element number. This number is used to track the corresponding elements and their reference sentences. The integer is followed by a DELIMITER that

[6]http://incubator.apache.org/pdfbox/
[7]http://snowtide.com/PDFTextStream
[8]http://www.foolabs.com/xpdf/about.html
[9]http://www.pdflib.com/products/tet/

⟨ CAPTION⟩	::=	⟨ DOC_EL_TYPE⟩ ⟨ Integer⟩ ⟨ DELIMITER⟩ ⟨ TEXT⟩
⟨ DOC_EL_TYPE⟩	::=	⟨ FIG_TYPE⟩ |⟨ TABLE_TYPE⟩ |⟨ ALGO_TYPE ⟩
⟨ FIG_TYPE⟩	::=	FIGURE|Figure|FIG.|Fig.
⟨ TABLE_TYPE⟩	::=	TABLE|Table
⟨ ALGO_TYPE⟩	::=	Algorithm|algorithm|Algo.|algo.
⟨ DELIMITER⟩	::=	: | .
⟨ TEXT⟩	::=	⟨ A String of Characters⟩

Fig. 3. A grammar for document-element captions.

can again be either ":" or ".". The final non-terminal TEXT gives a textual description of the element. Specifying a grammar enables us to follow a unified approach for dealing with different types of document-elements. Any new document-element can easily be handled by including rules for its DOC_EL_TYPE in the grammar.

Sentence Segmentation: After extracting the caption sentences from the document text, we need to split the document text into its constituent sentences. Since our goal is to identify and extract sentences that are related to document-elements, accurate *sentence segmentation* is very important. However, the raw text obtained after caption removal contains a lot of unwanted information such as document title, authors' names and affiliations, section headings, table data etc. They are not related to document-elements and they might harm the sentence segmentation process. We use the following heuristics to remove this noise and clean up the document text:

(1) **Average Line Length:** The *length* of a line is defined as the number of words in the line. All the lines in the document text with length smaller than η times the average line length are removed from the document text. This method helps in filtering out the section headings, titles etc., that are generally shorter than the remaining lines [Kupiec et al. 1995]. In this paper, we chose η to be equal to 0.8. We also made sure that no line at the end of a paragraph was removed. Also note that this pre-processing is done *after* caption extraction so that the caption is not discarded as noisy data.

(2) **Word Density:** The document text contains a lot of sparse lines corresponding to table data, equations, authors' names and affiliations etc. Generally, when converting from PDF to text, formatting of table text etc. is lost and the mathematical symbols in equations are not properly converted to text form. Hence, these need to be removed. In order to identify these sparse lines, we use a *word density* measure that is defined as follows:

$$dw_l = \frac{L}{L+S} \tag{1}$$

Here,
dw_l is the word density of line l,
L is the length of line l in words,
S is the number of spaces in line l.

Note that the word density of a normal text line is greater than 0.5, because in this case $S = L - 1$. So we filter out only those lines from the document having word density less than 0.5.

The cleaned up text is then fed to a sentence segmenter. It splits the document text into its constituent sentences and yields the sentence set $\mathcal{S}$.

Reference Sentence Parsing: Although captions provide some details about the element of interest, often they do not contain enough information to allow the reader to understand the information in the document element. Elzer et al. [2005], study the role of figure captions in understanding figures. They conclude that though captions help in understanding a figure's intended message, they alone are insufficient to help a reader to fully understand and interpret a given figure. In order to get a complete understanding of the content and context of a document element under consideration, we have to also analyze the running text in the document [Futrelle 1999]. Assuming good writing style, we hope to find at least one explicit reference to a particular document-element in the running text and this reference sentence can reveal useful information about the element. To identify reference sentences, we use a grammar similar to that used for caption parsing. However, there is a small difference. In the reference sentence, the delimiter will not be present in most cases and the integer will tell us to which element this sentence is referring.

In order to evaluate the performance of the above mentioned pre-processing steps, we used a set of 17 different Computer Science papers that had 192 document-elements and 258 associated reference sentences. The method described above for caption extraction was able to identify 177 caption sentences out of which 173 were correct and 4 were false positives (reference sentences identified as captions). Overall, the method achieved a precision of 97.74% and recall of 90.10%. The method for extracting reference sentences achieved a recall of 86.43% with 100% precision. However, we do note that the performance of these methods may differ depending upon the field of study as different fields use different formatting standards and the rules described above may need to be modified per the writing conventions of the field under study. The main objective of this work is to study the problem of synopsis generation and identifying captions or references given a general document is a separate problem that is out of scope of the present work. In this paper, we focused only on Computer Science publications and the methods described above worked very well for these papers.

3.2 Feature Extraction

As outlined in the previous section, complete understanding of document-elements critically depends on the content as well as the context in which they are used in the document. Therefore, we try to extract features for each sentence that can capture how well a sentence describes the content and the contextual information of a document-element.

3.2.1 *Content based Features*

(1) **Similarity with Caption (CapSYM):**
This feature utilizes information cues present in the caption. It is a score

assigned to each sentence based on its similarity with the caption. After removal of stopwords from the caption sentence and stemming using Porter's Algorithm [Porter 1980], the resulting keywords form a *"query"* that provides cues about the information contained in the document-element. This query is then used to assign *Similarity Scores* to all sentences in the document based on their similarity to the query. We adapt Okapi BM25 [Robertson et al. 1995; Manning et al. 2008] as our similarity measure, since it has been proved to be very effective in a wide variety of IR tasks. It is defined as follows:
If q is the generated query then the BM25 score of sentence s in document $\mathcal{D}$ is computed as:

$$BM25(q,s) = \sum_{t \in q} \left\{ \log \frac{N}{sf_t} \times \frac{(k_1+1)tf_{ts}}{k_1((1-b)+b \times (\frac{l_s}{l_{av}}))+tf_{ts}} \times \frac{(k_3+1)tf_{tq}}{k_3+tf_{tq}} \right\} \quad (2)$$

where:
N is the total number of sentences in the document,
sf_t is the sentence frequency, i.e., the number of sentences that contain the term t,
tf_{ts} is the frequency of term t in sentence s,
tf_{tq} is the frequency of term t in query q,
l_s is the length of sentence s,
l_{av} is the average length of sentences in D,
k_1, k_3 and b are constants which are set to 2, 2 and .75 respectively. These values have empirically been found to perform well [Manning et al. 2008].
In the above equation, the term $\log \frac{N}{sf_t}$ on the right hand side represents the *Inverse Sentence Frequency*. It is analogous in function to inverse document frequency (IDF) as used in information retrieval and deemphasizes common terms. The second term represents the frequency of each query term t in sentence s, normalized by sentence length and scaled by k_1. A value of $k_1 = 0$ refers to using a binary model where all the terms present in a sentence are given an equal weight. On the other hand, a large k_1 corresponds to using raw term frequency. Likewise, the third term scales the term weights by the frequency of terms in the query. The parameter $b(0 \leq b \leq 1)$ controls the amount of sentence length normalization, with $b = 1$ for full length normalization and $b = 0$ for no normalization at all. After computing the scores for all the sentences, the top 20 sentences with the highest scores are selected and assigned a feature value of 1. All other sentences are assigned a feature value of 0.

(2) **Similarity with Reference Sentence (RefSYM)**
Like captions, the reference sentences also contain important cues providing information about the document-elements. For all the reference sentences of a document-element, we compute their similarity scores with all the other sentences as described above. The top 20 highest scoring sentences are assigned a feature value of 1 while all other sentences get a feature value 0.

(3) **Cue Words and Phrases (CP)** There are certain cue words and phrases that are used frequently by authors while describing a document-element. For

accuraci	describ	illustr	origin	run
achiev	detail	improv	outperform	scenario
actual	determin	increas	output	schema
addition	differ	infer	paramet	scheme
aggreg	discuss	inform	partition	score
algorithm	distanc	input	percentag	slope
analysi	distribut	instanc	perform	show
approxim	docum	interest	plot	shown
assign	error	label	point	signific
averag	estim	larg	position	significantli
baselin	evalu	larger	precision	similar
case	execut	length	predic	size
collect	exist	level	previou	small
column	expect	line	problem	state
compar	experiment	list	procedur	step
comparison	fact	maximum	process	structur
comput	featur	mean	produc	system
concept	figur	measur	rang	tabl
consist	final	method	rank	techniqu
constraint	focu	metric	rate	test
content	frequenc	minimum	row	threshold
correl	frequent	model	record	time
cost	good	note	relat	total
curv	graph	number	repres	valu
data	hierarchi	observ	requir	vari
dataset	high	obtain	result	variabl
defin	higher	oper	return	x-axis
depict	highlight	optim	rule	y-axis

Table II. Cue words used in our experiments.

example, certain verbs are used typically to describe the purpose of document-elements ("shows", "describes", "illustrates" etc.). Likewise, there are many document-element specific words that can be used to identify related sentences ("distribution", "points" etc. for figures; "row", "column" for tables). A list of 140 such words (listed in Table II, after stemming) was created by manual inspection of 200 document-elements. These 200 document-elements were *different* from the ones we used for experiments in Section 5. A sentence in which one or more cue words/phrases were present was assigned a feature value of 1. All other sentences were assigned a feature value of 0.

3.2.2 *Context based features.* The features described above consider only the *content similarity* between sentences in a document and the document-element. They assume all sentences to be equally important. This assumption, however, is not true as explained in Figure 4. Generally, when a document-element is referenced in a sentence in the running text, the sentences near it also relate to the *document-element* and are *"contextually"* more important than the other sentences. These

"nearby" sentences provide the "context" in which a document-element is being described or used. We use the following features to identify and capture these contextually important sentences:

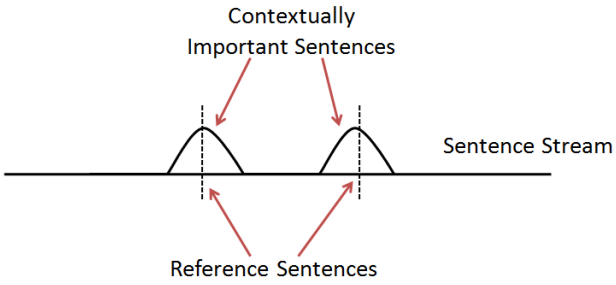


Fig. 4. Contextually Important Sentences. Our assumption is that sentences near the reference sentence of a document-element are more important than the sentences farther away from the reference sentence. Further, the importance of sentences decreases with their distance from the reference sentence.

(1) **IfReference Sentence (IfRefSent)**
It is a binary feature with a value of 1 if a sentence is a reference sentence for the document-element. Otherwise, it has value 0.

(2) **Paragraph Location (IsInSamePara)**
It is again a binary feature and has a value 1 if a sentence belongs to the same paragraph as the reference sentence. Otherwise, the value is 0.

(3) **Proximity** This feature captures the fact that a sentence closer to the reference sentence has a higher probability of being related to the document-element than a sentence located far away from the reference sentence. The first ten sentences on either side of a reference sentence are assigned a feature value of 1. All other sentences are assigned a feature value of 0.

3.3 Classification

In this sub-section we briefly describe the classification methods used for identifying document-element related sentences.

3.3.1 *Naïve-Bayes Classifier.* Naïve-Bayes classifiers have been previously used successfully to extract sentences for document summarization [Kupiec et al. 1995; Teufel and Moens 1997]. This method is simple, fast and can be easily adapted for use in modern digital libraries having millions of documents. It is defined as follows: Let the set of sentences that are related to the document-element d be $\mathcal{S}_d$ and let $\mathcal{S}$ be the set of all sentences in the document $\mathcal{D}$. Given the features $F_1, F_2, ..., F_n$ for sentence $s \in \mathcal{S}$, we use Bayes' rule to compute the probability that s also belongs to $\mathcal{S}_d$, as follows:

$$P(s \in \mathcal{S}_d \mid F_1, F_2, ..., F_n) = \frac{P(F_1, F_2, ..., F_n \mid s \in \mathcal{S}_d) P(s \in \mathcal{S}_d)}{P(F_1, F_2, ..., F_n)} \quad (3)$$

Assuming independent features, the above equation can be written as:

$$P(s \in \mathcal{S}_d \mid F_1, F_2, ..., F_n) = \frac{\prod_{i=1}^{n} P(F_i \mid s \in \mathcal{S}_d) P(s \in \mathcal{S}_d)}{\prod_{i=1}^{n} P(F_i)} \tag{4}$$

The probabilities $P(F_i \mid s \in \mathcal{S}_d)$ and $P(F_i)$ are not known *a priori* but they can be estimated by counting occurrences in the training set. This gives a simple Bayesian classification function that assigns a probability score to each sentence in the document. The top-scoring sentences can be identified as related to document-elements. The scores for all the sentences in the document are normalized in the range [0–1]. Note that $P(s \in \mathcal{S}_d)$ is the same for all sentences in the document and is therefore a constant. Since we are interested in the relative values of sentence scores and not the absolute values, this constant may be ignored.

3.3.2 *Support Vector Machines.* Support Vector Machines (SVMs) are a class of supervised learning algorithms that have been successfully used for a wide variety of classification problems [Bishop 2006]. In our problem, sentences in a document that are related to a document-element constitute one class (labeled positive) and all the other sentences constitute the other class (labeled negative).

For our problem, however, we can not directly use standard SVMs because in general, there are very few sentences in a document that are related to a document-element, i.e., the proportion of data points in the two classes is unbalanced. In such problems, it has been proposed to use different penalty parameters for different classes in the basic SVM formulation [Osuna et al. 1997]. Thus, the basic SVM problem can now be formulated as:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C_+ \sum_{y_i=1} \xi_i + C_- \sum_{y_i=-1} \xi_i \tag{5}$$

such that,

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \tag{6}$$

and,

$$\xi_i \geq 0, i = 1, \ldots, l. \tag{7}$$

Here, x_i is the feature vector of the data point under consideration and w is the weight vector that along with b determines the separating hyperplane between the two classes. C_+ and C_- are the parameters that determine the misclassification penalty associated with positive and negative examples. As indicated above, in the present problem, we have a lot more non-relevant sentences than relevant sentences. This imbalance can be accounted for by choosing C_+ and C_- so that the ratio of C_+ to C_- is greater than one.

Support Vector Machines predict the class labels of test points on the basis of the learned model. However, in addition to the class labels, we are also interested in knowing about the relative importance of individual sentences. Among all the sentences that are related to a document-element, some might be more important than others. This information is essential for producing a ranked list of relevant sentences and for automatically generating dynamic length synopses as described in the next section. In order to obtain probability estimates from the SVM output,

we use the method based on pairwise coupling as described by Wu et al., [Wu et al. 2003].

We used the LIBSVM library for support vector machines in this paper [Chang and Lin 2001]. It provides options to control the ratio of C_+ and C_- as well as implements the pairwise coupling method for computing the probability estimates.

4. SENTENCE SELECTION - DETERMINING OPTIMAL SYNOPSIS SIZE

After identifying the document-element related sentences, we need to decide how many and which sentences to include in the synopsis that will be presented to the user. Both of our classification methods also provide us with a score that is a measure of relative importance of individual sentences and can be used for selecting the most relevant sentences. Two possible approaches towards addressing this problem are 1) always select a fixed number of sentences, or 2) use a global score threshold for all document-elements. However, both these approaches have their own shortcomings [Metzler and Kanungo 2008]. Always returning a fixed number of sentences is a rigid strategy and fails to adapt to cases when there are fewer or more relevant sentences than the fixed number chosen. Similarly, choosing a fixed global score threshold that will work for all cases is difficult.

Carbonell et al. [1998] describe Maximum Marginal Relevance (MMR) as a criterion for selecting sentences for query-biased summarization. Their approach combines query-relevance and information novelty, and tries to minimize the redundancy in the final set of selected sentences. For a complete document like a paper, there are many sentences that convey the same information. For example, sentences in the abstract, introduction, conclusion etc., almost always have similar information content. However, for a document-element, we get only a small set of sentences that are actually related to it. Therefore, it is unlikely that such a small set of candidate sentences will introduce redundancy. However, we observed in our preliminary experiments that presenting all such relevant sentences to the user may have a detrimental effect on the *desirability* and *user-friendliness* of the synopsis due to the efforts involved in reading a longer synopsis. A longer synopsis might be comprehensive, but it may also contain some unrelated or marginally unrelated information. Moreover, it requires more time to read and understand a longer synopsis, thereby defeating the whole purpose of making search results more user-friendly. Therefore we seek to determine an optimum synopsis size that balances the trade-off between *information content* and *length* of the synopsis.

In general, the sentence selection problem can be framed as follows: let U_k be the *Utility* measure of sentence s_k that tells us whether it is useful to select the sentence or not. It is defined as:

$$U_k = g(k) - f(k) \tag{8}$$

Here, $g(k)$ is a function that favors the selection of s_k and $f(k)$ is another function opposing the selection of s_k. Sentences for which utility > 0 are included in the final set. Note that when $g(k)$ is the similarity between s_k and the query and $f(k)$ measures the redundancy of s_k, U_k becomes same as Maximum Marginal Relevance.

We now define the Utility measure more rigorously. Let the score of the k^{th} sentence be $score_k$ and let all sentences be ranked in decreasing order of their

scores so that $i < j$ implies $score_i \geq score_j$. We define the *Utility* measure U_k as:

$$U_k = score_k - (1 - exp^{-\lambda(k-1)}) \tag{9}$$

We include a sentence in the synopsis if and only if its utility is greater than zero. The above function is chosen so as to satisfy the following properties:

—The utility of a sentence is determined by two competing factors – (a) The relevance of the sentence to the document-element that is measured by the score of the sentence; (b) The penalty incurred by having an additional sentence s_k in the synopsis. The *Penalty Parameter* λ controls the penalty for including additional sentences and thus, determines the length of the synopses.

—$f(1) = 0$, this is important because no penalty should be incurred for including the first sentence. It ensures that we will never have an empty synopsis.

—$f(k)$ is a monotonically increasing function that assigns lower penalty to initial sentences and gradually increases the amount of penalty as more and more sentences are added to the synopses. If $\lambda = 0$, no penalty is incurred while adding a sentence and we will have the whole document as the synopsis. On the other hand, if λ is very high, we will have very short synopses.

The final set of selected sentences is arranged in the order in which they appear in the document. Non-consecutive sentences are separated by ellipsis (. . .) to indicate discontinuity and maintain readability and cohesiveness of the synopsis.

5. EXPERIMENTS AND RESULTS

In this section, we evaluate the effectiveness of the proposed methods for extracting document-element related information. We compare the two classification methods (Naïve Bayes and SVM) and study the effects of different features used. We also evaluate the proposed method for automatic sentence selection and study how variations in the penalty parameter λ affect the generated synopses. A user study is also performed to measure how the synopses generated by our proposed method compare with other state-of-the-art approaches.

5.1 Data Description

For our experiments, we selected 290 different document-elements from 152 different Computer Science publications covering varied topics such as operating systems, data mining, information retrieval, theoretical computer science and databases. The average length of each document is 314.66 sentences. The dataset consists of 163 figures, 78 tables and 49 algorithms. For each document-element, we asked two judges ($J1$ and $J2$) to manually identify the relevant sentences from the associated document. Judge $J1$ was a first year graduate student in Computer Science and judge $J2$ was a senior year undergraduate (honors) student in Computer Science. Note that users who evaluate our system must have some expertise in the area such that they can read and understand an academic paper clearly – a prerequisite for evaluating the generated synopses. Given this requirement, evaluation of our system using a system like the Amazon Mechanical Turk or using a general recruitment of students across the university were not feasible. Inevitably,

	Figure	Table	Algorithm	All
No. of document-elements	163	78	49	290
Caption length (no. of sentences)	1.28 ± 0.80	1.44 ± 0.81	1.00 ± 0.00	1.28 ± 0.75
Average number of reference sentences per document-element	1.42 ± 0.89	1.69 ± 1.27	2.14 ± 1.42	1.62 ± 1.16
Average number of relevant sentences (synopsis length)	9.02 ± 5.88	7.67 ± 4.32	8.00 ± 3.05	8.49 ± 5.13

Table III. Characteristics of our test dataset.

this limits the size of the evaluation study we can perform. Moreover, in order to avoid biased evaluations, we selected our evaluators such that both $J1$ and $J2$ were not associated with the project. $J1$ and $J2$ provided judgments for 140 and 150 document-elements respectively. For each document-element, they were asked to identify a set of sentences from the associated document that could describe the content of the document-element. We treat all such sentences identified by the human judges as *relevant* to the document-element and all the remaining sentences in the document as *irrelevant* to the document-element. Thus, for each document-element the sentences identified as relevant by the human evaluator were assigned a label 1 and all other sentences in the associated document were assigned a label -1. Table III summarizes the characteristics of the dataset thus created. We note from the table that the number of sentences that are relevant to a document-element is much less than the average number of sentences in a document (314.66 sentences).

5.2 Inter-annotator Agreement

In order to study the agreement between the two judges, we randomly selected 50 document-elements from the set evaluated by each evaluator and asked the other evaluator to provide his judgments for these document-elements. In this way, we obtained a set of 100 document-elements for which the synopses were provided by both $J1$ and $J2$. This set was used to study the agreement between the two evaluators. The average length of the synopses is 8.25 sentences for $J1$ and 7.38 sentences for $J2$. We used following two metrics to study the agreement between the two judges.

(1) **Jaccard Coefficient:** It is used to measure the similarity between two given sets and is defined as the ratio of the size of the intersection and the size of the union of the two sets. In our case, the two sets under consideration are the sets of relevant sentences as identified by each judge and the Jaccard Coefficient is computed as follows:

$$\frac{\text{Total number of sentences considered relevant by both J1 and J2}}{\text{Total number of sentences considered relevant by either of J1 or J2}}$$

The average Jaccard Coefficient between $J1$ and $J2$ was found to be 0.6438.

(2) **Kappa Coefficient (κ):** The Kappa Coefficient to measure the inter-annotator

		$J2$	
		Relevant	Non-Relevant
$J1$	Relevant	5.67	2.58
	Non-Relevant	1.71	353.26

Table IV. Confusion Matrix for judgments provided by two judges $J1$ and $J2$.

agreement is defined as follows:

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \tag{10}$$

Here, $Pr(a)$ is the observed agreement between the two annotators and $Pr(e)$ is the probability of the two annotators agreeing by chance. For our case, the average Kappa Coefficient between the two annotators was found to be 0.7427.

For this particular task, we note that the number of sentences in a document that are related to a document-element is much less than the number of sentences that are not related to the document-element. As mentioned in Table III, average synopsis length (i.e., number of relevant sentences) is just 8.46 sentences per document-element whereas a document contains a few hundred sentences. Hence, for a given document-element even if the two judges pick completely different sets of sentences to be included in the synopsis, the fraction of sentences for which their judgments agree will still be very high due to the large number of *irrelevant* sentences. Hence in order to get a better understanding of the inter-annotator agreement, we compute the confusion matrix between $J1$ and $J2$ (Table IV). The confusion matrix shown in Table IV is the average matrix for 100 document-elements used for studying inter-annotator agreement. From the confusion matrix, we observe that on an average there are 5.67 sentences per document-element that both the annotators agree on being relevant. Furthermore, on an average, there are 2.58 sentences per document-element that $J1$ considers as relevant but $J2$ considers as irrelevant. On the other hand, there are 1.71 sentences that $J2$ considers to be relevant but $J1$ considers them to be irrelevant. The number of sentences that both the annotators consider as irrelevant is very high (353.26 sentences per document-element).

5.3 Relevant Sentence Identification

The aim of this experiment is to evaluate how well the proposed methods are able to identify the document-element related sentences. We use a Naïve Bayes classifier and an SVM for computing probability estimates . Both methods assign a score to each sentence. The score is a measure of the sentence's relevance to the query. If the model learned is reasonable, then the sentences that are more relevant are assigned a higher score. Thus, each method gives us a ranked list of relevant sentences for each document-element. As discussed by Kanungo and Metzler [Metzler and Kanungo 2008], an appropriate evaluation measure for the sentence selection task is R-precision. For a given <document-element,document> pair, the R-precision is defined as the precision at rank R, where R is the total number of relevant sentences in the document. This measure is more appropriate than using precision at a fixed value because for different <document-element, document> pairs, the value of R

	Naïve Bayes	**SVM**
Figures	0.7424	0.7172
Tables	0.6867	0.6680
Algorithms	0.6190	0.6143
All	0.7110	0.6507

Table V. *R*-precision achieved by Naïve Bayes and SVM classifiers for the sentence extraction task. We report results for the whole dataset as well as for individual document-element types. The differences between the two classifiers were not found to be statistically significant at the 95% confidence interval using one-way analysis of variance (ANOVA) test. However, for each classifier, the difference between precision values for figures and algorithms were found to be statistically significant at 95% confidence interval using one-way ANOVA followed by a multiple comparison test using bonferroni compensation.

is different and ideally, we want to return only these R relevant sentences.

All the features that we use are categorical in nature. Categorical features are useful for Naïve Bayes classifiers. For our SVM, we transform the categorical features into numeric data as described in the LibSVM guide [Chang et al. 2009]. We use 5-folds cross validation for evaluation. For each validation, both SVM and Naïive Bayes classifiers were trained on the training set and then the performance of the learned classifier was evaluated using the test set.

Table V reports the R-precision values achieved by the two methods averaged over five validations. We report the results for the whole dataset as well as for each document-element type individually. R-Precision measures how many sentences out of the total R relevant sentences were present in the top R sentences. Note that here R is different for different document-elements. We see that the performance of both the methods is very similar for the sentence extraction task. We also note that the Naïve Bayes classifier outperforms the SVM classifier by small amounts, however, the differences between the two classifiers were not found to be statistically significant using a one-way analysis of variance (ANOVA) test at 95% confidence interval. Analysing the results by each document-element type, we observe that both the classifiers perform best for figures followed by tables and algorithms, in that order. In order to assess the statistical significance of these differences, we performed an unbalanced one-way ANOVA test followed by multiple comparison test using Bonferroni correction. For both the methods, differences between figures and algorithms were found to be statistically significant at the 95% confidence interval. From Table III, we observe that all the algorithms in the dataset have a single sentence caption. In fact, a majority of these captions were either function names (e.g. *Algorithm 3: SuffixFilter(x, y, Hmax, d)*) or algorithm names (e.g. *Algorithm 1 Link-Training Algorithm*). On the other hand, captions for figures and tables were comparatively longer and contained more information that could be utilized by our algorithm.

In Figure 5, we report precision at different ranks ($P@N$, $N = \{1, 2, 3, 4, 5\}$) for both the classifiers. Precision at N measures how many of the relevant sentences were present in the top N sentences. We observe that both the methods achieved high precision values at top 5 ranks indicating that both the methods are able to

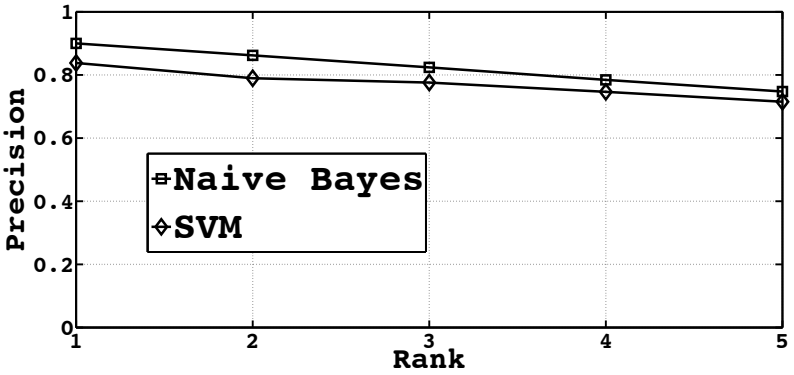
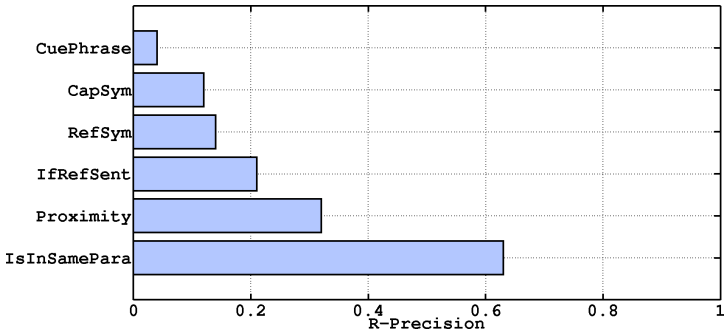
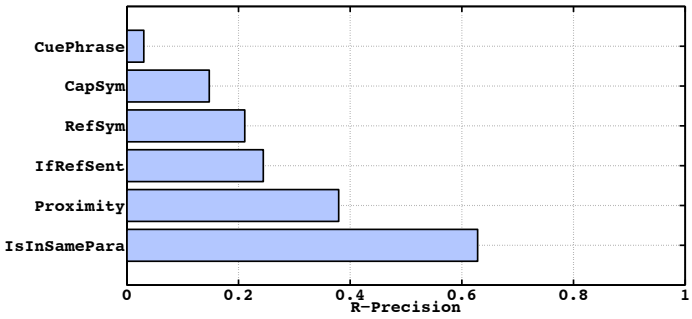


Fig. 5. Precision achieved by Naive Bayes and SVM classifier at different ranks.



(a) Naïve-Bayes



(b) SVM

Fig. 6. Performance of individual features for : (a) Naïve Bayes and (b) SVM.

learn reasonable models and the scores assigned on the basis of learned models are good indicators of the relevance of sentences to document-elements.

Next, in order to understand the relative importance of the different features we examine the performance of each individual feature for the sentence extraction task. The results for both the Naïve Bayes and SVM methods are summarized in

Figure 6. We note that the cue-phrase feature performs the worst for our problem even though it has been used successfully in sentence extraction for generic document summarization [Kupiec et al. 1995; Teufel and Moens 1997]. Moreover, we also observe that in general, the performance of the different context-based features is better than that of the content-based features. This observation substantiates our hypothesis that contextual information is essential for a proper understanding of the document-elements. The context based features help us identify regions in the document text that are important with respect to the document-element. The content-based features provide additional useful information that helps us to determine which sentences are actually related to the document-elements.

It is also interesting to note how well the human judges perform at this task and compare the performance of our proposed automated method with the performance of human judges. For this evaluation, we used the same set of 100 document-elements described in subsection 5.1. First, using the sentences marked by $J1$ as the "gold standard", we compute precision of the relevance judgment provided by $J2$. Then, we compute the R-precision of each synopsis as produced by the proposed method [10]. Next, we treat the sentences marked by $J2$ as the gold standard and compute the precision of the relevance judgments provided by $J1$ and the proposed method. Table VI summarizes the results. Using the relevant sentences provided by $J1$ as a gold standard, $J2$ achieves a precision of 79.90% whereas the automated methods achieve an R-precision of 73.79%. The difference in performance of the human judge and our proposed method was found to be statistically significant using a one-way ANOVA test at 95% confidence interval. When we treat the relevance judgment of $J2$ as the gold standard, the precision for $J2$ is 78.21% whereas our proposed method achieves an R-precision of 74.65%. The difference, however, was not found to be statistically significant. Note that we report the precision for relevance judgments provided by the human judges and R-precision for the proposed method. In our proposed framework, the number of top scoring sentences that should be retained in the synopsis is not known in advance. Using R-precision means that the synopsis produced by our proposed methods are of the same length as that of the gold standard synopsis. This is a reasonable approximation as the average synopsis length for the two human evaluators is also very similar (subsection 5.2). From Table VI we observe that the precision values achieved by our proposed method are quite comparable with the values obtained by the human judges, which can be considered as a reasonable upper bound for the given task. We do however note that the levels of agreement may vary depending upon the dataset as well as the human annotators.

5.4 Sentence Subset Selection

After identifying relevant sentences from the documents, we need to select a subset of top-ranking sentences that should be included in the final synopsis to be presented to the user. In this sub-section, we evaluate our proposed sentence selection strategy for this purpose. We propose a model for sentence selection trying to strike a balance between the information content and the conciseness of the generated synopses.

[10] We use the Naïve Bayes classifier for this part as its performance was found to be better than that of the SVM.

Gold Standard	$J1$	$J2$	Proposed Method
$J1$	–	0.7990	0.7370^{+}
$J2$	0.7821	–	0.7465

Table VI. Comparing the performance of proposed method with that of human judges. $^{+}$ indicate a statistically significant difference in performance as compared to the human judge using one-way ANOVA at 95% confidence interval.

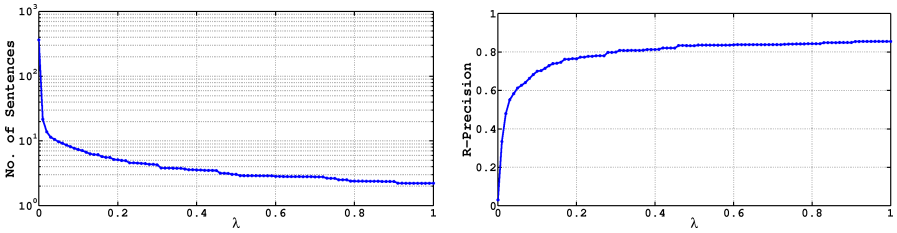
The penalty parameter λ, as defined in equation 9, controls the length of generated synopses by penalizing the inclusion of additional sentences in the synopses. In order to study the behavior of generated synopses with varying λs, we generated synopses for different values of λ, varying from 0 to 1, with increments of 0.01. For each value of λ, we compute the average length of synopses (in number of sentences), precision, recall and F1 measure. The results are summarized in Figure 7. Note that the variation of the average length of the synopses is shown on a log scale. In all other graphs, the y-axis is plotted on a linear scale.

From the figure, we observe that the average length of synopses decreases as λ is increased. For very small values of λ, almost no penalty is levied for the inclusion of additional sentences. The model tries to maximize the *information content* and as a result, we end up with pretty long synopses. As we gradually increase λ, the amount of penalty also increases and the less relevant sentences are filtered out. For very high values of λ, the model favors highly concise synopses and includes only the top few relevant sentences. The figure also shows the variations in precision, recall and the F1 score with different values for λ. Initially, with low values of λ, a majority of the sentences are selected which results in high recall and low precision values. Increasing λ results in the selection of fewer but highly relevant sentences. Thus, we observe an increase in precision and decrease in recall values. The F1 score, which is the harmonic mean of precision and recall, follows an interesting trend. It first increases rapidly with increasing λ, achieves a maximum at $\lambda = 0.07$ and then gradually falls. The F1 values remain stable in the range $0.40 - 0.48$ for $\lambda = 0.05 - .30$. The average synopses length in the same range lies in between 4.28 to 9.72 sentences. Here, the use of the penalty parameter λ provides us with a simple but powerful means of generating variable length synopses according to the needs of the users. Initially, using a moderate value of λ (say 0.3), we can provide a concise and highly informative synopsis. Then, if the user wishes to know more about the document-element, synopses generated with lower values of λ can be presented and the λ value dynamically adapted.

5.5 Quality Experiment

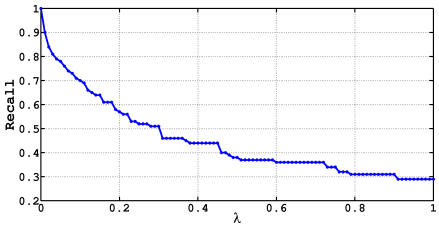
The aim of this experiment was to compare the synopses generated by our proposed approach with that generated by the current state-of-the-art methods and investigate and demonstrate the utility of synopses for document-element search engines. For this experiment, we used three different methods as described below.

(1) **Reference Sentence:** For this method, the synopses were generated for all the test cases by extracting all the corresponding reference sentences from the document text.

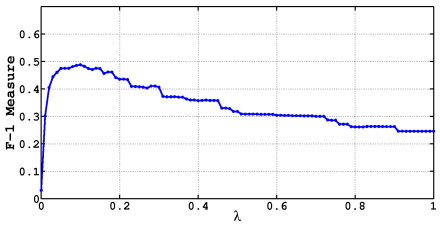


(a) No. of Sentences with λ

(b) Precision with λ



(c) Recall with λ



(d) F1 with λ

Fig. 7. Effect of penalty parameter λ on (a) Average No. of Sentences Selected, (b) Precision, (c) Recall and (d) F1 Measure.

(2) **Indri:** We used the Indri [11] search engine toolkit to generate query-biased summaries for document-elements. We indexed all the test documents using Indri and for each document-element, we queried the index using the same query formulated by extracting keywords from the caption and reference sentence as described in section 3. The synopses in this case are the *query-biased* snippets accompanying the corresponding documents returned as search results. One motivation [12] for using Indri was that it also supports long queries as is the case here.

(3) **Proposed Method:** We used the Naïve Bayes classifier for identifying relevant sentences because of its better performance than the SVM. For sentence selection, we used $\lambda = 0.3$. We chose this value of λ because at this value, the average synopses length is same as that of summaries produced by Indri (around 4 lines).

For this experiment, we chose a subset of 30 document-elements (10 each of figures, tables and algorithms) from our dataset. We generated synopses for each of these by the three methods described above which resulted in a total of 90 <document-element, synopsis> pairs. The two human evaluators ($J3$ and $J4$) evaluated the synopses generated by different methods. $J3$ and $J4$ were both graduate students in Computer Science and were different from the judges who prepared the synopsis gold set. For each document-element, the generated synopses were shown to the evaluators side by side and they were not told which synopsis was generated by which method. Further, the order in which the synopsis generated by differ-

[11] `http://www.lemurproject.org/indri`

[12] We would like to thank Susan Dumais for suggesting the use of Indri for this experiment.

Method	Document-Element Type	No. of times selected as First	Second	Third
Reference Sentence	Figure	1	5	4
	Table	2	4	4
	Algorithms	1	5	4
	All	4	14	12
Indri	Figure	0	4	6
	Table	0	4	6
	Algorithms	0	5	5
	All	0	13	17
Proposed Method	Figure	9	1	0
	Table	8	2	0
	Algorithms	9	0	1
	All	26	3	1

Table VII. Summary of rank orderings provided by the two evaluators for the three methods.

ent methods for a given document-element were shown to them was randomized. The evaluators were asked to rank the synopsis generated by three methods from most preferred to least preferred. Both $J3$ and $J4$ provided rank orderings for 15 document-elements (5 of each type).

The results are summarized in Table VII. Our proposed method emerges as a clear winner when compared to the other two methods. Our proposed method was the most preferred method for 26 out of 30 times, thrice it was the second most preferred method and only once was the least preferred method. The four times when the synopses generated by the proposed method were not the first choice of the evaluators, they chose the synopses generated by extracting all the reference sentences as the most preferred description of the given document-element. In all such cases, the document-element was referred to many times in the document text resulting in a comparatively large number of reference sentences (4 to 6). All these reference sentences indeed provide a good description of the document-element. Further, for many document-elements that had multiple reference sentences, synopses generated by the reference sentence method were chosen as the second most preferred method. Tables VIII and IX provide some examples of synopses generated by the three methods and the respective rankings assigned by the evaluators to these synopses.

Admittedly, the comparison of our proposed method with the two baselines described above may not be considered completely *fair* as the baseline methods do not utilize various content and context based information like the proposed method. However, these techniques have been employed by current state-of-the-art systems [Hearst et al. 2007; Liu et al. 2007] and our results highlight the inadequacy of these techniques in providing sufficient information for understanding document-elements and corroborate the need for synopsis generation for document-elements. The superiority of the proposed method, as evident from the results, shows that

Axis	Conditions
descendant	Ln(result)>Ln(n), Rn(result)>Rn(n)
ancestor	Ln(result)<Ln(n), Rn(result)<Rn(n)
parent	Ln(result)<Ln(n), Rn(result)<Rn(n), Ln(result) PLn(n)
child	Ln(result)>Ln(n), Rn(result)>Rn(n), PLn(result) Ln(n)
preceding	Ln(result)<Ln(n), Rn(result)>Rn(n)
following	Ln(result)>Ln(n), Rn(result)<Rn(n)
preceding-sibling	Ln(result)<Ln(n), Rn(result)>Rn(n), PLn(result) PLn(n)
following-sibling	Ln(result)>Ln(n), Rn(result)<Rn(n), PLn(result) PLn(n)

Table 1: Checking XPath Location Steps

Reference Sentence (1): Given an element n, constraints determined by Ln, Rn, and PLn to locate result elements along eight XPath axes (i.e., child, parent, descendant, ancestor, following, preceding, following-sibling, preceding-sibling) are presented in Table 1. Thus, by examining the constraints in Table 1, it is not difficult to verify that the book node is a descendant of the all-items node, and the book node precedes the account node. The conditions for matching XPath location steps along the major XPath axes, relying only on Ln, Rn, and PLn, in the absence of any metadata level information, has been reviewed in Section 3.2 and Table 1.

Indri (3): ...a full meta-data index (FMI), obtained by enhancing the R-tree index proposed for XPATH LOCATION steps [13] with a new attribute for each XML element, which contains the element's (explicitly...ancestor, descendant, following, and preceding. Associating an additional number with each node, e.g., the PARENT NODE'S LN, written as PLN, permits LEVEL sensitive MATCHING as well. Given an ELEMENT n, CONSTRAINTS DETERMINED by LN, RN, and PLN to LOCATE RESULT ELEMENTS along eight XPATH AXES (i.e., CHILD, PARENT, DESCENDANT, ANCESTOR, following, PRECEDING, FOLLOWING-SIBLING, PRECEDING-SIBLING) are PRESENTED in Table 1. A multi-dimensional index structure, such as an R-tree, on the Ln/Rn/PLn...

Proposed Method (2): Ln and Rn suffice for matching along ordered and structural axes, such as ancestor, descendant, following, and preceding. Associating an additional number with each node, e.g., the parent node's Ln, written as PLn, permits level sensitive matching as well. Given an element n, constraints determined by Ln, Rn, and PLn to locate result elements along eight XPath axes (i.e., child, parent, descendant, ancestor, following, preceding, following-sibling, preceding-sibling) are presented in Table 1....Each element node is labeled with three numbers: Ln, Rn and PLn. Thus, by examining the constraints in Table 1, it is not difficult to verify that the book node is a descendant of the all-items node, and the book node precedes the account node....The conditions for matching XPath location steps along the major XPath axes, relying only on Ln, Rn, and PLn, in the absence of any metadata level information, has been reviewed in Section 3.2 and Table 1.

Table VIII. Rank orderings provided by the two evaluators for the three methods. Numbers in brackets indicate the rank order, 1 being the most preferred and 3 being the least preferred. Table courtesy[Cho et al. 2006]

Algorithm 2 Concurrent Best First Search

Input: $k, Q, PQ_n, U, \mathcal{S}, D_{q_1,\ldots,q_n}$

```
1: for i = 1 to n do
2:     while PQ_i not empty do
3:         Entry N = PQ_i.top
4:         if N is a node then
5:             for j such that PQ_j.contains(N) do
6:                 PQ_j.remove(N)
7:                 for C in N do
8:                     if C not in U and C covers q_j then
9:                         PQ_j.enqueue(C, D(Q,C))
10:        else if N is a data entry then
11:            S.insert(N)
12:            Update D_{q_i}
13:            if k-NN is discovered, continue from 1
14: end for
```

Reference Sentence (3): The function ConcurrentBestFirstSearch (Algorithm 2) utilizes the best first search nearest neighbor algorithm to find the k-NN of every query point.
Indri (2): ...query is a binary AND operation on the bit vector. For trajectories that cover the QUERY, the actual distance D(Q, P) is computed and stored as the lower-bound. For other trajectories, the appropriate D(qi, P) values are used. The FUNCTION CONCURRENTBESTFIRSTSEARCH (ALGORITHM 2) UTILIZES the best first SEARCH NEAREST NEIGHBOR ALGORITHM to FIND the K-NN of every QUERY POINT. The SEARCH is incremental so that the priority queues can be preserved and reused between subsequent executions...
Proposed Method (1): The function ConcurrentBestFirstSearch (Algorithm 2) utilizes the best first search nearest neighbor algorithm to find the k-NN of every query point....Array D(qi) is maintained by storing for each query point the distance from the last entry removed from the top of the corresponding priority queue. Both functions can be straightforwardly modified to support time-interval predicates as well as top-k searches, where more than one trajectories are retrieved. For completeness, we use the extended algorithm for our experimental evaluation, but present the simpler versions here for ease of exposition.

Table IX. Rank orderings provided by the two evaluators for the three methods. Numbers in brackets indicate the rank order, 1 being the most preferred and 3 being the least preferred. Algorithm courtesy[Hadjieleftheriou et al. 2005].

the proposed method can be successfully used to overcome the shortcomings of the state-of-the-art techniques.

6. CONCLUSIONS AND FUTURE WORK

We identified the problem of generating synopses for document-elements like tables, figures, and algorithms in digital documents to assist quick understanding by users who are searching for these document-elements. Machine-learning techniques are used to identify relevant sentences from the document text using a novel set of features that utilizes content and context information related to document-elements. We proposed a simple model to determine which sentences to include in the synopses

based on their similarity with the caption and the reference sentences corresponding to a document element and the proximity of the sentence to the reference sentences. The model tries to balance the information content and the length of the synopsis so that the generated descriptions are both useful and succinct. The usefulness of our proposed approach is confirmed by empirical evaluation. In the future, we will identify more features to improve the quality of generated synopses and investigate the use of synopses for improved document search and document summarization.

7. ACKNOWLEDGMENTS

We would like to thank Susan Dumais and the anonymous reviewers for their valuable comments and suggestions that helped us improve the quality of this work. This material is based upon work supported by the National Science Foundation under Grant Nos. 0535656 and 0845487. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Bhatia, S., Lahiri, S., and Mitra, P. 2009. Generating synopses for document-element search. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*. ACM, New York, NY, USA, 2003–2006.

Bhatia, S., Mitra, P., and Giles, C. L. 2010. Finding algorithms in scientific articles. In *WWW 2010*. 1061–1062.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Carbonell, Jaime and Goldstein, Jade. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*. 335–336.

Chang, C.-C. and Lin, C.-J. 2001. *LIBSVM: a library for support vector machines*.

Chang, C.-C., Lin, C.-J., and Hsu, C.-W. May,2009. *A practical guide to support vector classification*.

Chen, Y., Wang, G., and Dong, S. 2003. Learning with progressive transductive support vector machine. *Pattern Recognition Letters 24,* 12, 1845–1855.

Cho, S., Koudas, N., and Srivastava, D. 2006. Meta-data indexing for XPath location steps. In *SIGMOD Conference*, S. Chaudhuri, V. Hristidis, and N. Polyzotis, Eds. ACM, 455–466.

Corio, M. and Lapalme, G. 1999. Generation of texts for information graphics. In *In Proceedings of the 7th European Workshop on Natural Language Generation EWNLG'99*. 49–58.

Demner-Fushman, D., Antani, S., Simpson, M., and Thoma, G. 2009. Annotation and retrieval of clinically relevant images. *international journal of medical informatics 78,* 12, e59–e67.

Elzer, S., Carberry, R., Chester, D., Demir, S., Green, N., Zukerman, I., and Trnka, K. 2005. Exploring and exploiting the limited utility of captions in recognizing intention in information graphics. In *In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*. 223–230.

Futrelle, R. P. 1999. Summarization of diagrams in documents. *Advances in Automated Text Summarization*, 403–421.

Futrelle, R. P. 2004. Handling figures in document summarization. In *Text Summarization Branches Out. Workshop at ACL*. 61–65.

Goldstein, J., Kantrowitz, M., Mittal, V., and Carbonell, J. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *In Research and Development in Information Retrieval*. 121–128.

Goldstein, J., Mittal, V., Carbonell, J., and Kantrowitz, M. 2000. Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 Workshop on Automatic summarization*. Association for Computational Linguistics, Morristown, NJ, USA, 40–48.

Guglielmo, E. J. and Rowe, N. C. 1996. Natural-language retrieval of images based on descriptive captions. *ACM Transactions on Information Systems 14*, 237–267.

Hadjieleftheriou, M., Kollios, G., Bakalov, P., and Tsotras, V. J. 2005. Complex spatio-temporal pattern queries. In *VLDB*, K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, and B. C. Ooi, Eds. ACM, 877–888.

Hearst, M. A., Divoli, A., Guturu, H., Ksikes, A., Nakov, P., Wooldridge, M. A., and Ye, J. 2007. Biotext search engine: beyond abstract search. *Bioinformatics 23,* 16, 2196–2197.

Huang, W., Tan, C. L., and Leow, W. K. 2005. Associating text and graphics for scientific chart understanding. In *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*. IEEE Computer Society, Washington, DC, USA, 580–584.

Kataria, S., Browuer, W., Mitra, P., and Giles, C. L. 2008. Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents. In *Proceedings of AAAI*. 1169–1174.

Ko, Y. and Seo, J. 2008. An effective sentence-extraction technique using contextual information and statistical approaches for text summarization. *Pattern Recogn. Lett. 29,* 9, 1366–1371.

Kupiec, J., Pedersen, J., and Chen, F. 1995. A trainable document summarizer. In *Proceedings of the 18th Annual International Conference on Research and Development in Information Retrieval (SIGIR'95)*. ACM Press, New York, NY, USA, 68–73.

Liu, Y., Bai, K., Mitra, P., and Giles, C. L. 2007. Tableseer: automatic table metadata extraction and searching in digital libraries. In *JCDL*. ACM, 91–100.

Luhn, H. P. 1958. Automatic creation of literature abstracts. *IBM Journal of Research and Development 2*, 159–165.

Mani, I. and Maybury, M. T., Eds. 1999. *Advances in Automatic Text Summarization*. The MIT Press, Cambridge, Massachusetts.

Manning, C. D., Raghavan, P., and Schütze, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Metzler, D. and Kanungo, T. 2008. Machine learned sentence selection strategies for query-biased summarization. In *Proceedings of SIGIR Learning to Rank Workshop*.

Osuna, E. E., Freund, R., and Girosi, F. 1997. Support vector machines: Training and applications.

Passonneau, R., Kukich, K., Robin, J., Hatzivassiloglou, V., Lefkowitz, L., and Jing, H. 1996. Generating summaries of work flow diagrams. In *In the Proceedings of the International Conference on Natural Language Processing and Industrial Applications (NLPIA'96)*. 204–210.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program 14(3)*, 130–137.

Robertson, S. E., Walker, S., Jones, S., Hancock-beaulieu, M. M., and Gatford, M. 1995. Okapi at trec-3. 109–126.

Sandusky, R. and Tenopir, C. 2008. Finding and using journal-article components: Impacts of disaggregation on teaching and research practice. *Journal of the American Society for Information Science and Technology 59,* 6, 970–982.

Teufel, S. and Moens, M. 1997. Sentence extraction as a classification task. In *Workshop on Intelligent and Scalable Text summarization, ACL/EACL*.

Tombros, A. and Sanderson, M. 1998. Advantages of query biased summaries in information retrieval. In *SIGIR*. ACM, 2–10.

White, R., Jose, J. M., and Ruthven, I. 2003. A task-oriented study on the influencing effects of query-biased summarisation in web searching. *Inf. Process. Manage 39,* 5, 707–733.

Wu, T.-F., Lin, C.-J., and Weng, R. C. 2003. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research 5*, 975–1005.