THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

Department of Computer Science and Engineering

EFFECTIVE DOCUMENT-ELEMENT SYNOPSIS GENERATION

BRENDAN HARNETT
Spring 2010

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Prasenjit Mitra
Assistant Professor of IST
Thesis Supervisor

John Hannan
Associate Professor of Computer Science
Honors Advisor

* Signatures are on file in the Schreyer Honors College.

Effective Document-Element Synopsis Generation

Brendan Harnett

April 14, 2010

Abstract

In order to keep abreast of many scientitific developments in their field, modern researchers spend significant amounts of time reading those academic papers which report or summarize the conclusions of experiments similar to their own. To quickly determine the results of these experiments, researchers heavily rely on document-element entities such as figures, tables, and algorithms which are not part of the running text of a paper itself, but are instead pictorial representations of the results or conclusions described in the paper. However, such document-elements are almost always difficult to interpret or understand without an accompanying "synopsis" - a series of sentences selected from the paper itself for the purpose of describing the document element. In our experiments, we manually identify ideal synopses for 160 document-elements. We then test the effectiveness of algorithmic methods proposed by Bhatia et al. to automatically generate synopses by comparing the generated synopses to the ideal ones [1]. Interestingly, our experiments produced results very similar to those outlined in Bhatia et al., leading us to believe that our findings are fairly consistent for papers in different conferences, regardless of the subject matter. But allthough synopsis generation for the collection of all document-elements was consistent, effectivness varied when comparing each document-element type individually with synopses for figures being the most complete, and those for algorithms being the least.

Acknowledgements

I'd like to thank Sumit Bhatia for providing me with the framework, guidance, and tools necessary for carrying out my experiments and developing a written thesis. In addition, I'd like to thank Dr. Prasenjit Mitra for his continued oversight and support throughout the thesis development process.

Contents

1 Introduction

While electronic databases of academic research papers continue to grow, implementing methods for efficiently navigating these databases has become increasingly important. Researchers desire ways to quickly discover the results of experiments in their field, and in particular, the results of experiments similar to their own which have been performed by others. Almost all academic papers include more than just plain text, and usually contain visual aids supplementing or summarizing data in the paper. These visual-aids, also referred to in this paper as "document-elements", tend to be broken down into the categories of figure, table, and algorithm. Figures most often depict graphs, while tables, unsurprisingly, present information in a tabular format, and algorithms describe pseudo-code. For our study we collected papers from a variety of conferences to compute statistics on the number of document elements appearing in each conference. These statistics are shown in Table 1.

Conference	No. of Papers	Figures		Tables		Algorithms	
		Total	Avg	Total	Avg	Total	Avg
CIKM05	159	492	3.09	265	1.67	31	0.195
CIKM06	137	538	3.92	262	1.91	31	0.23
CIKM07	132	558	4.23	384	2.91	22	0.17
CIKM08	267	1048	3.93	634	2.37	88	0.22
SIGIR05	138	338	2.45	330	2.39	9	0.07
SIGIR06	152	373	2.45	335	2.2	4	0.03
SIGIR07	221	412	1.86	425	1.92	16	0.07
SIGIR08	207	439	2.12	460	2.22	25	0.12
SIGIR09	207	428	2.07	373	1.8	22	0.11
SIGMOD05	116	742	6.4	92	0.79	18	0.16
SIGMOD06	99	622	6.28	96	0.97	35	0.35
SIGMOD07	137	979	7.15	185	1.35	70	0.51
SIGMOD08	132	1123	8.51	180	1.36	92	0.7
SIGMOD09	124	923	7.44	141	1.14	92	0.74
STOC05	85	118	1.39	7	0.08	11	0.13
STOC06	79	89	1.13	12	0.15	6	0.08
STOC07	78	84	1.08	7	0.09	15	0.19
STOC08	85	83	0.98	4	0.05	30	0.35
STOC09	79	99	1.25	6	0.08	16	0.2
VLDB05	142	1047	7.37	146	1.03	37	0.26
VLDB06	136	1042	7.66	191	1.4	63	0.46
VLDB07	150	1276	8.51	209	1.39	91	0.61
VLDB08	68	1159	17.04	132	1.94	68	1
VLDB09	42	755	17.98	123	2.93	28	0.67
WWW05	81	503	6.21	103	1.27	13	0.16
WWW06	214	714	3.34	255	1.19	18	0.08
WWW07	229	901	3.93	355	1.55	37	0.16
WWW08	234	829	3.54	332	1.42	36	0.15
WWW09	198	811	4.1	385	1.94	38	0.19

Table 1: Document-Element distribution among conferences

As can be seen from the table, the largest portion of document elements are figures, the second most are tables, and the least amount of document elements are algorithms. Since these items can provide important summaries of results, researchers have a strong interest in being able to search through the document-elements themselves, selecting a document-element's corresponding paper only when the document-element has been deemed sufficiently relevant to the user. However, often the caption associated with a document-element does not contain enough information to determine the document-element's relevance. For this reason, it is essential that the sentences within the paper that are most relevant to a document-element be presented to the user along with the document-element itself. These sentences, collectively referred to as a synopsis, have proven historically to be effective at describing a document-element to a

user who lacks direct knowledge of a paper's content. Consider the graph shown in figure 1 (referred to as Figure 9 in the original paper).

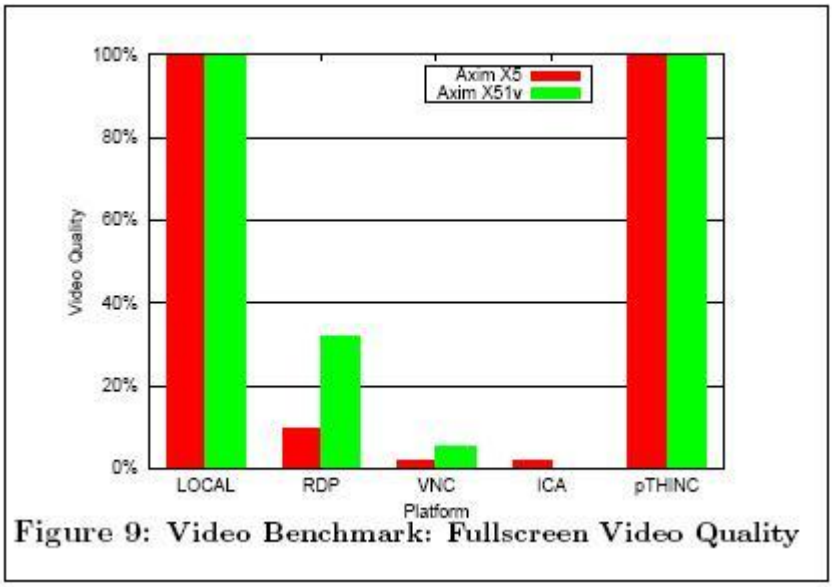

Figure 1: Example document element with standard caption from [5] .

We can see that although the caption for the figure aids our understanding of the data being described, it still leaves much to be desired. In this instance, a synopsis generated by our methods greatly increases our understanding of the graph's message, as shown in Figure 2.

Figure 9 shows the video quality for each platform.
pTHINC is the only thin client able to provide perfect video playback quality, similar to the native PDA video player.
All of the other thin clients deliver very poor video quality.
With the exception of RDP on the X51v which provided unacceptable 35% video quality, none of the other systems were even able to achieve 10% video quality.
VNC and ICA have the worst quality at 8% on the X5 device.

Figure 2: Generated synopsis for document-element in Figure 1.

The goal of our work is to propose methods that would automate the process of selecting ideal sentences for a document-element's synopsis while balancing the needs for an informative synopsis and one that is not too long for a user to read.

2 Related Work

A wide variety of tools and methods have been proposed for searching and extracting information from document-elements. For extracting useful information from Tables, Liu et al. present *TableSeer*, a search engine designed specifically for tables in digital documents [2]. They utilize a vector-space algorithm called *TableRank* to rank returned search results. Additionally, the CiteSeerX digital library provides functionality for table search[1].

A problem with the search engines mentioned is that they do not provide textual information to help a user determine the relevance of a document-element. Futrelle has introduced a method for summarizing diagrams using the structure of the diagram itself, captions, and running text in the document [3]. Huang et al. use graphical and textual information to interpret the semantics of scientific charts[4].

Where our methods differ from previous efforts is in our use of the running text for actual summarization of a document-element. We believe that given the high likelihood that each document-element in a document is referred to by at least part of the running text, presenting appropriate sentences describing the document-element would greatly increase a user's ability to make a relevance judgement.

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

3 Setup

Before a system for generating ideal synopses can be developed, the notion of an ideal synopsis must be defined. From our collection of conference papers stored in pdf-format we found ideal synopses for 160 different document-elements with the majority of them appearing in different papers. This was no trivial task, since it required that countless hours be spent examining sentences in a variety of conference papers to see which ones would best suit a selected document-element. We found that relevant sentences in the majority of cases tended to be clustered together, either appearing around a sentence that referred to a document-element by name, such as a sentence containing the phrase "Figure 23 shows...", or appearing in the same paragraph as a reference sentence. Sentences with equations or symbols that do not translate well to text were generally not selected since they depended on elements of the paper not immediately relevant to the document element. Sentences which directly refer to a document-element are of course relevant by definition.

Upon selecting ideal synopses for document-elements, we performed the task of converting all of our conference papers stored in pdf format to text files using the xpdf conversion tool with the raw parameter.[2] This was necessary to allow our scripts to effectively parse the papers in order to generate synopses. Unfortunately, it becomes difficult to read text files containing complex mathematical symbols and equations, and for this reason such equations and symbols were usually removed from the text files after conversion. We also removed any headings, sub-headings, document-element captions, and after internal document-element data remaining after the conversion to text. These types of text contribute no additional value in understanding the nature of a document-element and would likely confuse a user if presented to them with other sentences.

Once our pdf-files were converted, we ran a series of scripts to generate data for every paper that contained one or more of our 160 document-elements. Each text file had an associated sentence file, caption file, paragraph file, and manual synopsis file.

- Sentence files list each sentence of the original text file on a single line. By storing the sentences in this format, we can refer to each sentence by its line number in the sentence file.
- Caption files store the full caption for every document-element in a paper.
- Paragraph files list the sentence numbers of those sentences which appeared in the same paragraph as a reference sentence.
- Manual synopsis files contain the sentence numbers of those sentences which we had manually deemed to be most relevant to a document-element.

[2] http://www.foolabs.com/xpdf/

4 Description of Features

Our goal is to score each sentence of a paper based on how relevant that sentence is to a given document element. We represent a sentence as a feature vector with each dimension of the vector representing that sentence's score for a particular feature. We assess the relevance of a sentence by both content and contextual features outlined in [1], but we describe them again in detail in the following sections.

4.1 Content Based Features

We first devise methods for scoring sentences based on the particular words that they contain.

4.1.1 Similarity with Caption (CapSYM)

Since the caption often provides vital information necessary for understanding the content of a document-element, we want to attribute higher scores to those sentences which contain many of the same words as those in the caption. We do so by first removing all stopwords from the caption and then stem the remaining words using Porter's algorithm [3]. We use this result to form a query where all sentences in the document are assigned scores based on their similarity to the query. We chose a variation of the Okapi BM25 similarity measure for sentence scoring due to its successful use in a variety of Information-Retrieval tasks [7, 8]. Our method is defined as follows:

If q is the generated query then the BM25 score of a sentence s in document D is computed as:

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

where:

N is the total number of sentences in the document,

sf_t is the sentence frequency, or 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_{avg} is the average length of sentences in D

k_1, k_3, and b are constants set to 2, 2 and .75 respectively.

The BM25 score is the sum of scores for each individul term t of our query q. The log function term of equation 1 is the *Inverse Sentence Frequency* and assigns higher scores to those words which appear in both our query and a sentence but not in many sentences throughout the document. The second term

represents the frequency of a each query term t in a sentence s, normalized by sentence length and scaled by k_1. This term assigns a higher score to those sentences where a query term occurs more frequently without favoring longer sentences. We use K_1 as a constant to determine how heavily we want to weigh the frequency of a term in a sentence, ignoring term frequency altogether if $K_1 = 0$. The third term scales scores for a query term by the number of times that term appears in a query. The parameter $b\,(0 \leq b \leq 1)$ controls the extent to which we normalize the length of sentences, where $b = 0$ indicates no length noramalization and $b = 1$ refers to full normalization. Upon computing similarity scores for all sentences in a document, we assign a feature value of 1 to the top 20 scoring sentences and a feature value of 0 to all others.

4.1.2 Similarity with Reference Sentence(RefSYM)

Reference sentences are often equally as important as captions in describing the content of a document-element. We therefore assign similarity scores to those sentences which have similar words to the reference sentences for a document-element using the same equation given for captions. The top 20 highest scoring sentence are then assigned a feature score of 1 with all other sentences being assigned a score of 0.

4.1.3 Cue Words

The cue words feature is a binary feature indicating whether or not a sentence contains one or more words from a pre-constructed list of "cue" words. It was implemented under the assumption that certain words appeared more frequently in relevant sentences. To develop this list, we selected all of the words in 200 different reference sentences as well as all of the words appearing in the two sentences before and after the occurence of those reference sentences. Of the words collected, we removed all stop words and selected the 245 most frequently occurring words to use in our list.

4.2 Context Based Features

The above mentioned features are important, but they only capture the content similarity of sentences. However, ideal synopses typically contain contextually important sentences as well. Such sentences are located near reference sentences and can help provide the user with a more coherent understanding of a document-element's content.

4.2.1 IfRefSent (IfRefSent)

If a sentence makes a direct reference to a document-element in question, we assign a score of 1 for the IfRefSent feature and 0 otherwise. Reference sentences tend to be considerably valuable in aiding a user's understanding of a document-element and they are used as a basis for the other two context-based features in this section.

4.2.2 Paragraph Location (IsInSamePara)

We found that those sentences which appeared in the same paragraph as a reference sentence had a higher probably of being included in a document-element's synopsis. Therefore, any sentence that appeared in the same paragraph as a reference was assigned a feature score of 1 and 0 otherwise. The paragraph files mentioned in Section 3 provided a means for knowing which sentences were in the same paragraph as a reference sentence.

4.2.3 Proximity Feature

A sentence is assigned a proximity feature score of 1 if the sentence is within 10 sentences before or after a reference sentence for a given document-element. Otherwise, it is assigned a proximity feature score of 0. This feature utilizes the fact that sentences located next to a reference sentence have a greater likelihood of being relevant to a document-element.

5 Classification

Our experiment uses Support Vector Machines (SVMs) to classify document sentences. We want to separate all sentences into one of either two classes - referred to as positive and negative - depending upon whether a given sentence is relevant to a document-element. SVMs perform classification by learning a separating hyperplane which divides a set of training examples such that positive examples, or in our case relevant sentences, are on one side of the hyperplane, and examples of the other type - non-relevant sentences - appear on the other side. Due to the fact that a relatively small portion of sentences in a document are relevant to a document-element, which in turn leads to an unbalanced distribution of points, we use different penalty parameters than those used by a standard SVM. Our SVM classification is thus defined as a solution to the following problem:

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

such that,

$$y_i \left(w^T \phi \left(x_i \right) + b \right) \geq 1 - \xi_i \tag{3}$$

and,

$$\xi_i \geq i = 1, ..., l. \tag{4}$$

In the above equations, x_i refers to our generated feature vectors, w and b are two quantities which together form the boundaries of the separating hyperplane, and$C+$ and $C-$ are the parameters that determine the misclassification penalty. Due to the fact that the ratio of non-relevant to relevant sentences is greater than one, we choose $C+$ and $C-$ such that the ratio of $C+$ to $C-$is greater than 1.

To perform the classification we use the LIBSVM classification library which allows us to control the $C+$ to $C-$ ratio [9].

6 Experiments and Results

We began our experiment by computing feature vectors. The vectors were generated for every sentence in a document that contained one of our 160 selected document-elements and were computed based on how they related to our document-elements according to the features described in Section 4. We realized, however, that 9 of our 160 documents did not have reference sentences and were therefore incapable of generating appropriate feature vectors. Our experiment was thus conducted with only 151 of the original document-elements.

We ran the libsvm classification tool on four different datasets while varying the weight parameter w from 1 to 10 in intervals of 0.1. For each weight we computed the Precision at N for N = {1,2,3,4,5} and R-Precision. Precision at N refers to how many of the top N sentences returned by the SVM were relevant to a document-element. R-Precision can be thought of as Precision at R, where R is the total number of relevant sentences that exist for a document-element.

The first set consisted of all the 151 test document-elements. An experiment with this set would yield valuable insights into the effectiveness of our methods across papers in many more conferences than those tested in Bhatia et al.

Second, we wanted to discover how Precision at N and R-Precision varied for each document-element type. We divided our major set into 3 component sets of 52 figures, 50 tables, and 49 algorithms while repeating our classification procedure on each set.

6.1 Main Set

For each of the weights used with the SVM, Precision at N and R-Precision were maximized at $w = 6.3$ for the set of 151 document-elements. For w values less than 6.3, the metrics were noticably worse, with initial w values between 1 and 3 failing to achieve R-Precision greater than 50%. For weights larger than 6.3, the precision values do diminish, but only by a few percentage points.

The maximized precision values of our experiment are presented in Table 2 with the results of the Bhatia et al. experiment shown in table 3. As can be seen, the results shown in each table are remarkably similar. Table 2 shows a modest increase over Table 3 for P@1, P@2, and R-Precision, while Table 3 shows a slight increase for P@3, P@4 and P@5.

P@1	P@2	P@3	P@4	P@5	R-Precision
0.8600	0.8667	0.8134	0.7650	0.7400	0.7107

Table 2: Precision values for our main data set.

P@1	P@2	P@3	P@4	P@5	R-Precision
0.8286	0.8500	0.8286	0.7929	0.7500	0.7032

Table 3: Precision values given for data set in [1].

6.2 Figures Set

Unlike the main set, there was no weight w which we tested that maximized each precision value. We therefore chose a weight that maximized R-Precision since knowing the percentage of relevant sentences retrieved is more useful than the other precision values in identifying the overall quality of our methods. The results are shown in Table 4. We can see that the figures set performs noticably better than the main set in every category, with R-Precision being improved by almost 11%.

P@1	P@2	P@3	P@4	P@5	R-Precision
0.9038	0.8942	0.8782	0.8365	0.7923	0.8114

Table 4: Precision values for set consisting of 52 figures.

6.3 Tables Set

We found that the majority of precision values were maximized with weight $w = 9.2$. As table 5 shows, the results are comparable to the precision values of the main set, but are worse than the figures set for every precision value except N = 1.

P@1	P@2	P@3	P@4	P@5	R-Precision
0.9200	0.8400	0.7933	0.7650	0.7280	0.6725

Table 5: Precision values for set consisting of 50 tables.

6.4 Algorithms Set

After examining the precision values for weights between 1 and 10, we recognized what looked like an upward trend in precision. We therefore computed the precision values with weights between 10 and 20 using increments of 0.1 to find that the R-Precision was maximized at $w = 13.4$. As is clear from the data in Table 6, the algorithms set had the worst precision of all the sets tested. Interestingly, the P@3 and P@4 values were identical.

P@1	P@2	P@3	P@4	P@5	R-Precision
0.7778	0.7000	0.6889	0.6889	0.6756	0.6325

Table 6: Precision values for set consisting of 49 algorithms.

7 Conclusions

An important result of our experiments was that the precision values for our main set of document-elements were very close to those found in [1]. The fact that the values were so similar even though we used papers from many more conferences leads us to believe that the effectiveness of our methods is less dependent on the subject-matter of the individual conferences. However, we do understand that our results may not adequately represent the true precision values of document-elements in papers of the STOC conferences, as those tended to be difficult to read due to the multitude of mathematical symbols and equations. We also realize that for each set the Precision at N = {4,5} might be lower because a few of our ideal synopses only contained 3 or 4 sentences, and in such an instance a Precision at 4 or 5 will always result in a precision less than 1.

There seems to be a possible correlation bewteen the amount that a document-element type occurs and its relative precision values. Figures occurred the most frequently in our collection and yielded the best precision while algorithms ocurred least frequently and yielded the worst. We attribute this difference to two primary factors: the proportion of a paper dedicated to a document-element, and the proximity with which relevant sentences occurred. Since figures and tables occurred more frequently within papers, a smaller portion of the running text was dedicated to each document-element, and so our system did not have to attempt to select as many relevant sentences to compute the R-Precision. Not only were descriptions of algoritms longer, but many of the most relevant sentences were more widely distributed throughout a paper. On the other hand, relevant sentences for figures and tables were almost always close to a reference sentence, typically appearing in the same paragraph.

Our results also show that the contextual features used to determine a sentence's relevance to a document element are more importan the the content-based features. Most of the relevant sentences were either reference sentences or sentences appearing close to or in the same paragraph as a reference sentence. The cue words feature was probably the least effective since, using our our set of 245 cue words, most sentences in a paper contained one of the cue words. In the future this can be easily remedied by cutting down the size of the cue words list. There is of course a trade-off that comes with modifying the size of a cue words list. If the list is too long we are more likely to find relevant sentences that have a cue word, but we do so at the cost of finding more false positives. If the list is too short, we risk failing to identify relevant sentences that do not have one of our cue words.

8 Future Work

Although our paper measured the precision values of different datasets, we never generated a complete synopsis for each of our document-elements. Doing so requires that a balance be struck between synopsis utility and synopsis length. We want the synopses to be thorough, but not so long as to frustrate a user. A technique for managing these competing priorities is through the use of a utility function U_k where:

$$U_k = g(k) - f(k) \tag{5}$$

In the above function, U_k refers to the utility of the kth highest scoring sentence in a document for a particular document-element. The utility function is designed so that any sentence where U_k>0 is presented in a document-element's synopsis. The function g is one that favors the selection of a sentence k, and f is another function opposing the selection of k. Different formulas can be used for g and f but generally g should be a function utilizing the computed score of a sentence k and f should be a function that increases with k. Future work would include developing formulas for g and f which maximize the utility of generated synopses.

Section 7 discussed the problem of our list of cue words potentially being too large. Future experiments should test how the precision values change with a smaller list of cue words, or with words collected from sentences other than the two sentences occurring before and after a reference sentence.

Even if our techniques efficiently return most of the relevant sentences for a document-element, it is still necessary to have those synopses evaluated by a user. Due to certain time constraints, we were unable to perform a test which would allow users to score our synopses and compare them with synopses generated by other methods or tools. We did, however, develop an application in PHP and MySQL that automates this process and we intend to use it once users begin assigning quality scores to our synopses.

References

[1] Bhatia, S., Lahiri, S., and Mitra, P. 2009. Generating synopses for document-element search. *In Proceeding of the 18th ACM Conference on information and Knowledge Management* (Hong Kong, China, November 02 - 06, 2009). CIKM '09. ACM, New York, NY, 2003-2006. DOI= http://doi.acm.org/10.1145/1645953.1646287

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

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

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

[5] J. Kim, R. A. Baratto, J. Nieh. pTHINC: A ThinClient Architecture for Mobile Wireless Web. *In Proceedings of the 15th International World Wide Web Conference (WWW)*, May 2006.

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

[7] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-beaulieu, and M. Gatford. Okapi at Trec-3. pages 109–126, 1995.

[8] C. D. Manning, P. Raghavan, and H. Sch¨utze. *Introduction to Information Retrieval*. Cambridge University Press, 2008

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

ACADEMIC VITA OF BRENDAN A. HARNETT

Brendan A. Harnett
304 South Tyson Ave.
Glenside, PA, 19038
Brendan.harnett@gmail.com

Education: Bachelor of Science Degree in Computer Science, Penn State University, Spring 2010
Honors in Computer Science
Thesis Title: Effective Document-Element Synopsis Generation
Thesis Supervisor: Prasenjit Mitra

Related Experience:

Internship with Software Deployment and Automation Engineering Team at Air Products & Chemicals
Supervisor: Todd Houser
January – August 2009

Internship with Information Technology Department at General Electric Water & Process Technologies
Supervisor: Adam Malinauskas
May – August 2008

Awards:

Dean's List 7/8 Semesters
Richard A. McQuade Scholarship Recipient: 2006-2010

Presentations/Activities:

President and Chapter Founder of Upsilon Pi Epsilon Computer Science Honor Society,

Co-created and managed technology summer camp program: Summers 2005-2007

Alpha Epsilon Pi Fraternity: Spring 2005-Present

Penn State Certified Six Sigma Yellow Belt