Automatic Research Summaries in DBLife

Andrew B. Goldberg, David Andrzejewski
CS 764 - Final Project

May 16, 2007

1 Introduction

The Cimple project on Community Information Management [4] is a project with the goal of developing a software platform for the effective management of data related to a given online community. The DBLife project [3] is a prototype system to intended to help test and extend the ideas of the Cimple project, focused specifically on the database research community.

In DBLife, each researcher in the database community has their own super-homepage. This page contains information about that researcher, automatically extracted from various sources on the web. From this page, a user can learn about this researcher's collaborators, recent talks, and publications. However, there is currently no way to get an easily digestible, at-a-glance summary of general research interests over an entire career.

Our CS 764 project uses statistical natural language processing (NLP) [8] techniques to automatically create two types of research summaries, enabling users to quickly and easily learn about the research of a given database researcher in DBLife. The first type is a chronological summary, where the researcher's career is partitioned into contiguous segments of calendar years, and those segments are then tagged with appropriate research topic labels or keyphrases. The second type is an aggregate career summary, showing which labels and keyphrases were most commonly attributed to the researcher over the entire career. See Figure 1 for an example summary generated for AnHai Doan. Note that the italicized phrases are automatically extracted keyphrases, while the other topic labels come from a controlled topic list and are automatically chosen based on their similarity to the author's work.

2 Data Sources

To build research summaries, we require two data sources. Clearly, we need data related to researchers' work. To this end, we use the publication record of a given researcher from their DBLP-like page in DBLife. This contains references to the author's publications, including the standard information like the title, year, co-authors, and conference or journal where the work appeared. Since

AnHai Doan

Time period categories/keyphrases

From 1994 to 1996

- approximation and uncertainty
- *abstracting probabilistic actions*
- *modeling probabilistic actions*
- *decision-theoretic planning*
- *empirical analysis*
- *sound abstraction*
- *constraint mass assignment framework*

From 1998 to 2001

- data cleaning, data translation, data exchange, schema matching, record linkage
- data integration, heterogeneous database systems, interoperability
- relational query processing, query optimization

From 2002 to 2005

- web data processing, web services, e-commerce
- data integration, heterogeneous database systems, interoperability
- data cleaning, data translation, data exchange, schema matching, record linkage

From 2006 to 2007

- data cleaning, data translation, data exchange, schema matching, record linkage
- web data processing, web services, e-commerce
- metadata management, data semantics

Overall category distribution

Category	Fraction
data cleaning, data translation, data exchange, schema matching, record linkage	0.2778
data integration, heterogeneous database systems, interoperability	0.2222
web data processing, web services, e-commerce	0.1667
relational query processing, query optimization	0.1111
other	0.0833
approximation and uncertainty	0.0833
metadata management, data semantics	0.0556

Figure 1: An automatically generated research summary for AnHai Doan.

DBLife does not collect full-text articles or abstracts, we were limited to using only the titles of publications.

The second data source we need relates to the way we will represent researchers' topics of interest. With the goal of being able to later perform higher-level analysis, we wanted to represent topics using a fixed set of topic labels. (We initially experimented with using unigram, bigram, or trigram statistics to present a researcher's interests, but the great variety in such statistics makes any sort of higher-level analysis extremely difficult.) Thus, in order to get topic labels with the appropriate level of granularity (neither too specific, nor too general), we used a controlled list of research areas within the database field found at `http://scratchpad.wikia.com/wiki/Dblife_bibs`. Each research area has a manually chosen label and set of representative publications. Each label and corresponding set of publication titles form a category.

As discussed in the next section, at a high level, a time period in an author's career is matched to a category label by comparing the researcher's publication titles in that time period to the publication titles associated with that cate-

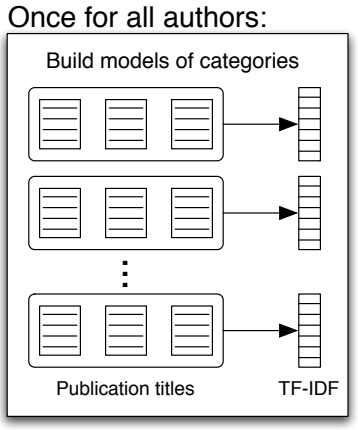

Figure 2: Generating a model for each category.

gory. If fewer than 3 categories from the controlled list can be matched to a given time period, additional keyphrases are extracted directly from the titles of publications from that time period, using the TextRank algorithm [11].

3 Procedure for Building Research Summaries

An overview of the procedure for creating the research summaries is as follows:

1. For each category
   Build category model
2. For each author
   Segment the author's career
   For each segment
   Build segment model
   Find the categories which are most similar to this segment
   Generate keyphrases from segment (Optional)
   Build author research summary from segments

Step 1 is illustrated in Figure 2 and discussed in detail in Section 3.1, and Step 2 is depicted in Figure 3 and presented in Sections 3.2–3.6 and Section 4.

During the course of the project, various alternative approaches were explored. The method presented here was chosen based on trade-offs between the level of detail, practicality, efficiency, and simplicity.

For example, the research summary for a single author depends only on that author's publication titles, our controlled list of category labels, and the associated category models. This has the benefit of making it relatively simple

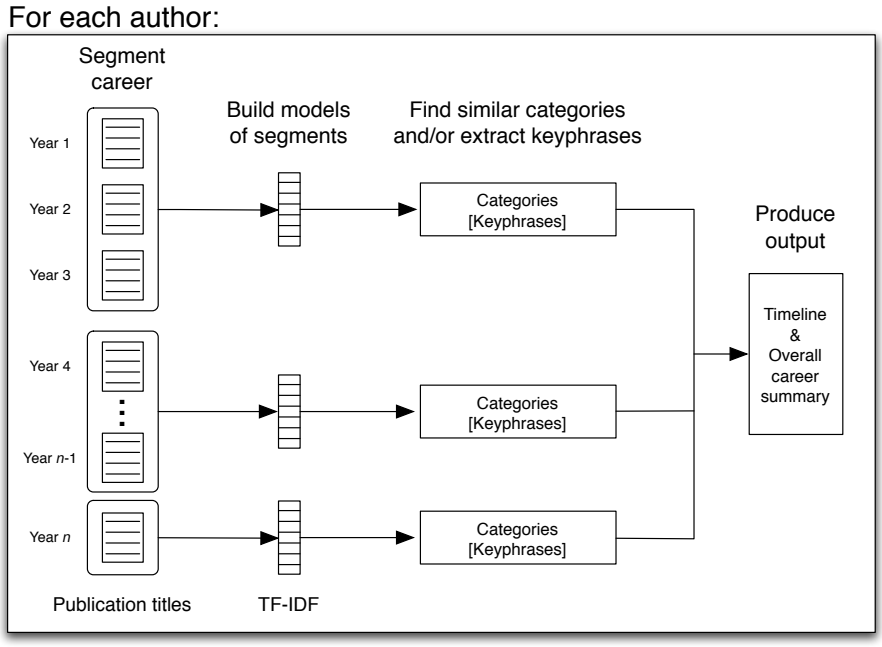

Figure 3: Generating an author's research summary.

and efficient to update the research summary for an author, or to add a new author to the system. A system which incorporates co-authorship relations or other global information may be able to exploit the additional data in some way, but could be significantly more difficult to implement, update, or modify. For the purposes of this project, and creating a new DBLife feature, we opted for the simpler representation which relies only on local information with respect to a single author.

3.1 Building Category Models

In order to determine which category labels best describe a time period in the author's career, we need a common data representation with an easily computable similarity measure. *Tf-idf* vectors [8] are a well-studied and useful representation for bag-of-words text data, with a simple distance measure, the cosine between two vectors. *Tf* refers to the term frequency within a document, and *idf* refers to the inverse document frequency of a term, or a quantity inversely proportional to the number of documents containing that term. Combining these two quantities puts more emphasis on terms which are more rare amongst all documents, and therefore more likely to be useful for making meaningful distinctions between documents.

For each category (a total of 47, ignoring the "other" and "unknown topic" categories), we gather the associated publication titles. We then represent each

category as a *tf-idf* vector, where the *tf* component is based on all the titles in a single category, and the *idf* is taken over all categories (making it more like "inverse category frequency"). These *tf-idf* representations of each category allow us to compare author publication titles to the titles in a given category, and assign category labels to author time periods appropriately.

3.2 Segmenting the Author's Career

We segment a career based on the text contained in the author's publication titles. We represent each year by a *tf-idf* vector, where *tf* is based on all the publication titles in that year, and *idf* is over all the years ("inverse year frequency"). To compare the similarity of two adjacent years, we use the cosine similarity of the year *tf-idf* vectors.

We want the segments of the author's career to correspond to coherent "eras" in their careers as researchers. For example, someone may start their career focused on query optimization, but then later they may be more active in data integration. Ideally, our segmentation would reflect this, grouping their query optimization years together in one segment and their data integration years together in a different segment. Of course, real research interest transitions will never be so abrupt or clean-cut, but this intuition roughly guides what we wish to accomplish.

To decide which years to draw the segment boundaries between, we use Hearst's TextTiling algorithm [5]. For each candidate boundary, this algorithm computes the similarity between the texts which lie within a fixed-length window immediately on either side of the boundary. Computing these boundary-similarity scores for all possible boundaries (here we only allow boundaries between years and compare only one year on each side) gives a graph similar to that shown in Figure 3.2. Segment boundaries are then chosen at the deepest "valleys", with the tallest "peaks" on either side. Intuitively, this approach tries to make the boundary cuts between highly dissimilar segments, leaving highly similar time periods together in the same segment. This is in agreement with our original goals outlined above.

In order to decide how deep a valley warrants making a new segment boundary, we set the similarity threshold for drawing a segment boundary based on the mean and standard deviation of the rises and drops of cosine similarity between all years, as in [5]. In addition, our system uses a minimum segment length of L years for all segments except the first and last, in order to avoid an unhelpful over-segmentation. For our example results we set $L = 3$.

3.3 Building Time Segment Models

Once we have our segmentation, we represent each segment as a *tf-idf* vector built from all of the publications from all of the years contained within that segment. Note that these are different *tf-idf* vectors than used for segmentation—there we use one vector per year, whereas now we are compressing all the years in a particular segment down to one vector. Also, note that here we use the

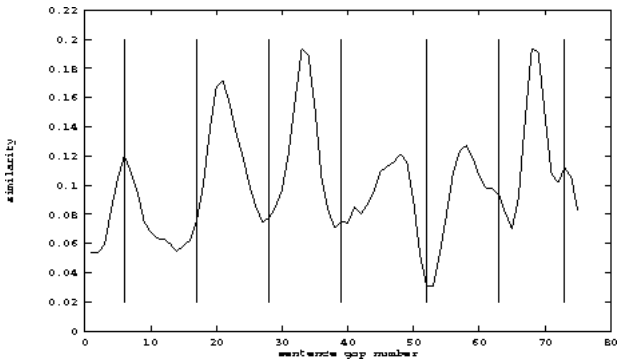

Figure 4: Example boundary-similarity chart, from [6]. In our case, each point on the x-axis corresponds to a year boundary (i.e., between 2006 and 2007). The y-axis is the cosine similarity between the *tf-idf* vectors representing the two years touching the boundary.

same vocabulary and *idf* values obtained when building category models (since the next step is to compare segments against the categories).

3.4 Matching Categories to Segments

To make the segments understandable, we next find the categories that most resemble the segments, and use those category labels to summarize the segment. The controlled list of category labels should then allow a user to get quick, high-level idea of what kind of research the author was involved with during that time period (e.g., "data integration between 2000 and 2006").

To determine which categories are most similar to this segment, we compare each segment *tf-idf* vector to the *tf-idf* vectors of all 47 categories using cosine similarity. For each segment, we then select the top K most similar categories (for these experiments $K = 3$). If the similarity scores for each of these top categories are above a certain threshold T (for these experiments $T = .05$), then we assign that category label to the author as a "topic" result for this time period.

3.5 Extracting Additional Keyphrases using TextRank

It may be the case that the publication titles from an author's time period are not similar to the publication titles from any of our categories. It may be that the author employs an idiosyncratic publication naming style, or that the author's primary field of research is not strictly database-related, and therefore not well-covered by our controlled list of categories (e.g., optimization or artificial intelligence researchers).

We obviously do not want to blindly take the top K category labels if they are not very similar to the author's publication titles (hence the cosine threshold

of the previous section). Therefore, if necessary because fewer than K categories surpass the threshold, we expand the list of labels for a segment using automatic keyphrase extraction. We accomplish this by applying the TextRank algorithm to the author's publication titles from this time period, and taking at least K keyphrases (but possibly more if there are ties in scores), and assigning them to the author as "keyphrase" results for this time period.

TextRank is a graph-based algorithm, similar in spirit to PageRank [12]. Whereas PageRank ranks Web pages with the help of the graph implied by their hyperlink connectivity, TextRank ranks individual words with the help of the graph implied by their positions in the text, with edges placed between words that co-occur within some fixed-length window (we use a window of size 2). TextRank ranks words based on their eigenvector centrality in the co-occurrence graph, and then forms meaningful keyphrases to extract by joining high-ranking words which co-occur as contiguous phrases in the original text (titles in our case). For example, if the words "database" and "security" are given high scores by TextRank for an author's publications, and they co-occur in the phrase "database security," TextRank will select that as a keyphrase.

Our system uses keyphrases extracted in this way to supplement our summary of a given time period when we cannot find sufficiently good matches from list of categories. While the TextRank-extracted keyphrases can sometimes be author-specific or unclear, we felt they were much more informative than simply using dissimilar category labels, or nothing at all. Furthermore, since these keyphrases may be noisier than the category labels from the control list, it may be beneficial to allow users to edit or correct these "keyphrase" results through a wiki interface. In addition, we could treat the user corrections as new category labels and use the current author's associated publications as the initial seed documents to form the new category *tf-idf* vectors.

3.6 Building the Research Summary by Time Segment

The end result of the previous steps is that each author's career is split into contiguous time segments, and each segment is characterized by up to K category labels and possibly K or more keyphrases. This can be represented as a timeline of time period segments and associated labels and keyphrases, allowing a user to get a high-level view of a researcher's interests over time.

4 Building the Career Research Summary

We can also collapse the time period results into a single, career-wide representation. For each category label, we assign a weight equal to the number of years spanned by all segments for which the category label was chosen as one of the top K. We also include an extra label "other" to account for years in which keyphrases were required. We can then build a histogram (or some other data display) showing the weights for each category label used. In the example output of Figure 1, we show the result as a normalized distribution over categories.

This career-wide summary shows which general categories were most similar to the author's publications over their entire career, and provides a quick snapshot of their interests.

The final output of the system is an XML file for each author, containing the years spanned by each segment, the labels and keyphrases associated with each segment, and the career-wide summary statistics. The DBLife system can then display this information to the user as needed. Also, as mentioned below, the information in this file can also be used to supplement or enhance other existing DBLife features.

5 Additional Applications and Future Work

With our time-period and entire-career research summaries in hand, it is possible that we could leverage these results to enhance the performance of other aspects of DBLife. For example, the entire-career research summary labels could be used to help populate the "Similar authors" section of a researcher's super-homepage. Currently this is based on relatively simple co-occurrence statistics, but a semantically richer comparison between authors' topic distributions could produce more meaningful results. The entire-career representation could also be useful in automatically selecting reviewers for publications submitted to a conference. In addition, it would be straightforward to map one or more of a time period's category labels to each of the constituent publications, and use citation statistics to determine in what areas a researcher has been most influential. Finally, our time-period results could also be used to gather statistics across all authors to identify research trends and "hot" or "cold" topics.

During work on this project, having access to publication titles only has been a crucial limitation. The small amount of text available in these titles results in datasets which are simply too sparse to be handled by many interesting statistical NLP models. With access to more text (e.g., abstracts), it may be possible to exploit the generalization capabilities of latent topic models such as Probabilistic Latent Semantic Analysis (pLSA) [7] or Latent Dirichlet Allocation (LDA) [1]. For example, a pLSA-based segmentation scheme [2] would be advantageous. This is because the latent topics in pLSA would allow our segmentation algorithm to match words as being similar by virtue of their membership in a common latent topic, as opposed to our word-only *tf-idf* representation.

Other extensions could involve trying to explicitly model the temporal evolution of research interests throughout a researcher's career [9, 10, 13]. An interesting potential application for these models would be to see which research topic transitions are most common, or to try to predict future research directions for a given author. For example, it may be the case that many researchers begin their careers in AI, but eventually transition to more database-related work. As mentioned above, these models would require significantly more text to to achieve meaningful results.

Lastly, the research summaries we produce could likely benefit from commu-

nity involvement through a wiki interface. In addition to converting keyphrases to coherent categories, users may be able to improve upon our initial segmentation or topic assignments (perhaps in a controlled fashion). Finally, it would be interesting to try to apply machine learning techniques to learn from the user modifications and tune the parameters of our system's various algorithms.

6 Conclusion

We have presented a system for constructing research summaries for authors in the DBLife project. Manual evaluations of system output suggest that summary content is generally correct and relevant. The results should allow DBLife users to quickly determine a researcher's primary research topics, both overall and throughout various phases of their career. We have outlined a number of potential applications and future enhancements to the current work, all of which ought to further improve DBLife.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. Topic-based document segmentation with probabilistic latent semantic analysis. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 211–218, New York, NY, USA, 2002. ACM Press.

[3] Pedro DeRose, Warren Shen, Fei Chen, Yoonkyong Lee, Douglas Burdick, AnHai Doan, and Raghu Ramakrishnan. Dblife: A community information management platform for the database research community (demo). In *CIDR*, pages 169–172, 2007.

[4] AnHai Doan, Raghu Ramakrishnan, Fei Chen, Pedro DeRose, Yoonkyong Lee, Robert McCann, Mayssam Sayyadian, and Warren Shen. Community information management. *IEEE Data Eng. Bull.*, 29(1):64–72, 2006.

[5] Marti Hearst. Multi-paragraph segmentation of expository text. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 9–16, New Mexico State University, Las Cruces, New Mexico, 1994.

[6] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *SIGIR '93: Proceedings of the 16th Annual International ACM/SIGIR Conference*, 1993.

[7] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.

[8] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[9] Qiaozhu Mei, Chao Liu, Hang Su, and Chengxiang Zhai. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 533–542, New York, NY, USA, 2006. ACM Press.

[10] Qiaozhu Mei and ChengXiang Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 198–207, New York, NY, USA, 2005. ACM Press.

[11] Rada Mihalcea and Paul Tarau. TextRank: Bringing order into texts. In *EMNLP'04*, 2004.

[12] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[13] Xuerui Wang and Andrew McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM Press, 2006.