PubTag: Generating Research Tag-Clouds with Keyphrase Extraction and Learning-to-Rank

Paula Ríos
Department of Computer Science
University of Chile
Santiago, Chile
prios@dcc.uchile.cl

Aidan Hogan
IMFD Chile / Department of Computer Science,
University of Chile
Santiago, Chile
ahogan@dcc.uchile.cl

***Abstract*—We investigate automated methods to generate tag-clouds for Computer Science researchers based on keyphrase extraction methods and learning-to-rank models. Given as input the identifier of an author in a bibliographical database (currently DBLP), the method extracts links to the PDFs containing the full-text of the paper. Keyphrase extraction methods are then applied to extract multi-term tags from the text. In order to select the most important tags for the researcher, we propose a set of features that serve as input for a variety of learning-to-rank models. Evaluation is conducted with respect to 12 Computer Science professors, who score a selection of keyphrases extracted from their papers indicating their relevance as a description of research topics. These scores are used to train and compare various learning-to-rank models for reordering the most important keyphrases, which in turn are used to generate final tag clouds for the professors. We further validate the proposed approaches by asking professors to evaluate the final tag-clouds.**

***Index Terms*—tags clouds, learning to rank, dblp**

I. Introduction

Identifying the research interests of a particular person has applications for expert-finding systems, for collaboration networks, and more generally, for summarising the research competencies of a particular person, department or institution. However, identifying the key interests of a researcher is a non-trivial task, even perhaps for the researcher themselves.

A key resource for identifying the interests of an individual researcher is their publication record, often tracked by a bibliographical database such as Google Scholar, Scopus, or – in the case of Computer Science researchers – DBLP. Embedded in the text of these publications are *keyphrases* [1] that help to characterise the topic of the particular paper; analysing the keyphrases from the collection of papers then allows to characterise the research interests of an author.

Still, potentially many keyphrases could be extracted by such a method, where there is thus a need to prioritise and visualise the resulting terms. A common approach for providing a visual summary of such information is to provide a *tag-cloud* [2], which puts more visual emphasis on *tags* (in this case, keyphrases) that are deemed to be more important.

Although various systems exist that (semi-)automatically generate tag-clouds from text [3], we could not find one suitable for the task of summarising the interests of a researcher. Many tag-cloud generators only support extracting single-term keywords, where collocations such as "`information extraction`" are fractured into "`information`", "`extraction`". Furthermore, the criteria for prioritising tags for researchers may be different than in other domains/applications, where we explore learning-to-rank methods trained on a labelled dataset from the target domain.

In this paper, we propose and evaluate a method for generating tag-clouds from the publication record of an individual researcher. This method has four main stages: (1) parse the publication record of the researcher, find links to full-text papers, and **extract the text**; (2) apply **keyphrase extraction** techniques to the text of each paper, aggregating the keyphrases across papers; (3) apply **learning-to-rank** models to prioritise keyphrases; (4) generate a customisable **tag-cloud**.

To evaluate our approach, we implement it in a system called PubTag for generating Computer Science tag-clouds: given a URL of a researcher's publication record in the DBLP bibliographical database, we implement a system that uses the above method to generate tag-clouds. The system can be configured to use one of four learning-to-rank models to generate the final order of tags. We perform experiments for 12 Computer Science professors, asking them to evaluate both the keyphrases extracted from their papers, as well as the final tag-clouds generated by the proposed method/system.

Our results are, in general, positive, and also provide insights into how similar/different are the results of individual learning-to-rank models. More generally, our findings suggest some subjectivity between researchers on which tags are important, and indeed which features of tags are important.

II. Related Work

Keyphrase Extraction: Keyphrase extraction identifies words and/or phrases that characterise the subject of a given document (aka. keyphrases) [4]. Keyphrase extraction approaches typically involve three high-level processes [5]: (i) *candidate selection*: an initial list of interesting words/phrases are extracted, potentially using a variety of linguistic or statistical techniques; (ii) *property calculation*: a set of features are collected for the initial words/phrases; (iii) *scoring and*

This work was funded in part by the Millennium Institute for Foundational Research on Data (IMFD) and Fondecyt, Grant No. 1181896.

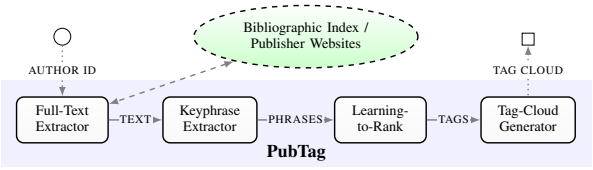


Fig. 1: System Architecture

selecting keyphrases: feature values are combined with a formula or machine learning model to score/select keyphrases.

Learning-to-Rank: While machine learning classifiers assign an input to a fixed number of output classes, learning-to-rank assigns an ordering over a set of inputs [6]. There are three main categories of learning-to-rank methods [7]: *point-wise methods* accept a single input and apply classification or regression to assign that input either a class or a score; *pair-wise methods* accept pairs of inputs and train classifiers or regression models to indicate whether or not the first input is less than the second input in the ordering, where this model can then be used to order unseen pairs; *list-wise methods* accept a list of inputs and directly output an ordering. Some works have proposed to apply learning-to-rank methods for refining the results of keyphrase extraction. Amongst these works, Jiang et al. [8] showed that the RankingSVM [9] learning-to-rank method improves a baseline keyphrase extraction method using a Naive Bayes classifier; Eichler and Neumann [10] similarly apply the RankingSVM method to keyphrase extraction; while Wang and Li [11] propose and apply the CoRankBayes method for keyphrase extraction.

Tag Clouds: In recent years, tag clouds have become more and more popular as an informal graphic – what Viegas and Wattenberg [12] call a "vernacular visualisation" – to convey a set of phrases (tags) and their importance [2]. Tag clouds have been used for a variety of applications, with one of the most prominent being to summarise search results. Amongst these works, Kuo et al. [13] propose to use tag clouds to summarise the search results of the PubMed system; Koutrika et al. [14] use tag clouds to summarise search results over tables representing information about courses.

Contributions: We apply keyphrase extraction and learning-to-rank methods to automatically generate a tag-cloud describing the interests of a given researcher. Our main contributions are: (1) a novel labelled dataset of 1,126 author-scored keyphrases extracted from the text of papers coauthored by 12 Computer Science professors; (2) an evaluation of four learning-to-rank frameworks using the above dataset, and a comparison of their results with an unsupervised keyphrase extraction framework; (3) a novel end-to-end system called PUBTAG, which automatically generates a customisable tag-cloud of research interests from a personal publication record.

III. PROPOSED APPROACH

Our proposed approach for generating research tag-clouds is depicted in Figure 1. We first describe the process to extract the full-text of papers; we then describe the keyphrase extraction phase, the learning-to-rank methods used to refine the scoring of keyphrases, and the generation of tag clouds.

Full-Text Extraction: We assume as input a link to a personal bibliographical publication record, where currently the system supports the DBLP index. The system uses the RDF metadata provided by the DBLP index to identify links to publisher pages where the paper can be downloaded. For each paper, the system then parses and searches the publisher page (ACM, Springer, Elsevier, etc.) in order to identify links to PDF documents; for this step to work, it is important that the system be run within the paywall of a particular institution.[1] While the detection of PDF links is straightforward on some publisher websites (looking for hyperlinks in the HTML pointing to a file with extension `.pdf`), other publishers are less straightforward, and generate the link when the page is loaded using a client-side script; since such cases are quite common, we use Selenium[2] to generate these PDF links and extract them. In some cases, no PDF link may be found or may remain behind a paywall, in which case we skip this paper. Finally, we use PDFMiner.six[3] to extract text from the PDFs.

Keyphrase Extraction: Over the full-text of the papers of the given author produced by the previous step, we now apply keyphrase extraction. We considered two seminal keyphrase extraction methods. KEA [1] selects candidate keyphrases of at most three (stemmed) words that are not proper nouns and do not start or end with a stop word; features include TF–IDF and phrase depth (how early the keyphrase appears in the document); scoring/selection is performed with a Naive Bayes model trained on labelled data. RAKE [15] also selects candidate keyphrases with at most a fixed number of words; a feature for each word is computed based on the number of unique words with which it co-occurs in a candidate keyphrase divided by its appearances in the document; scoring is based on summing the scores of individual words in a candidate keyphrase. Based on initial experiments, we use RAKE [15] for keyphrase extraction: it does not assume, *a priori*, a labelled dataset[4], plus many research interests are defined as proper nouns, which KEA ignores; furthermore, RAKE detects more technical keyphrases due to its co-occurrence feature.

Learning-to-Rank: The RAKE tool performs keyphrase extraction with respect to a general metric. Our hypothesis is that the scoring and selection of keyphrases for Computer Science can be improved by taking into consideration domain-specific preferences. Hence we propose to associate the initial keyphrases extracted by RAKE with a set of features and apply learning-to-rank models to see if we can find a better selection/scoring of final tags. The features we propose are:

- *RAKE*: keyphrase extraction score produced by RAKE;

[1] In terms of copyright, we do not make the full-text of such papers available through the system; only selected keyphrases.

[2] https://www.seleniumhq.org/

[3] https://github.com/pdfminer/pdfminer.six

[4] We initially tried training KEA with user-defined keyphrases in papers, but could not extract enough such keyphrases for effective training.

- *TF–IDF*: appearances of the keyphrase in the document weighted by overall appearances in a broader reference corpus of documents;
- *phrase depth*: appearances of a keyphrase in the document with higher weights given to earlier appearances in the document; a keyphrase occurrence is weighted by $1 - \frac{\text{position}}{\text{word count}}$ where, e.g., the 100$^{\text{th}}$ word in a 500 word document is weighted 0.8; the keyphrase is assigned the sum of weights for all occurrences in the document.
- *document ratio*: the ratio of documents collected for the author containing the keyphrase at least once;
- *first year*: the first year in which the keyphrase appears;
- *last year*: the last year in which the keyphrase appears;
- *interval of years*: the last year minus the first;
- *Wikipedia:* a boolean value indicating whether or not a Wikipedia article exists with the keyphrase verbatim as a title (case insensitive, encoded characters).

Using these features, we incorporate the following learning-to-rank approaches (selected to provide a mix of different styles of techniques with source code available): (1) *Linear regression*: a point-wise approach;[5] (2) *Ranking SVM* [9]: a pair-wise approach based on an underlying SVM classifier; (3) *LambdaMART* [16]: a pair-wise approach based on gradient boosted decision trees;[6] (4) *AdaRank* [17]: a list-wise approach based on optimising a linear combination of "weak rankers".[7]

We will compare the performance of these learning-to-rank approaches in the evaluation section for re-ordering the results of the keyphrase extraction framework.

Tag-Cloud Creation: Based on the previous steps, we arrive at a list of keyphrases associated with a score. The final stage of the process is to generate the tag-clouds, which primarily involves fixing certain visual aspects. Most importantly, the system must decide the font sizes of individual keyphrases; for this, we normalise the font sizes according to the rank induced by the scores, thus considering only the order of keyphrases, not the absolute difference of their scores. When longer terms appear (in terms of visual length), the lengths must be normalised to fit the frame of the tag cloud. We use the WordCloud library to render the tag clouds.[8]

PubTag: We combine the previous techniques into the PUBTAG system (see Figure 1). A user enters a URL for a researcher's profile in a bibliographical index (currently only DBLP is supported). The above steps are then applied and a set of alternative tag clouds are presented to the user based on the various learning-to-rank methods supported; given that scraping and extracting text from PDFs takes time, we apply caching of the text for an author such that if a user returns later and enters the same URL, they will be able to review the results. The user can select their preferred tag cloud and apply some customisations to remove tags, reorder or add new tags, change visual aspects such as colours and fonts, etc.

[5]Using Scikit-learn
[6]Using Pyltr: https://github.com/jma127/pyltr
[7]Using AdaRank: https://github.com/rueycheng/AdaRank
[8]http://amueller.github.io/word_cloud/

IV. EVALUATION

We conduct a preliminary evaluation to address the following research questions (we mark key questions with "⋆"):

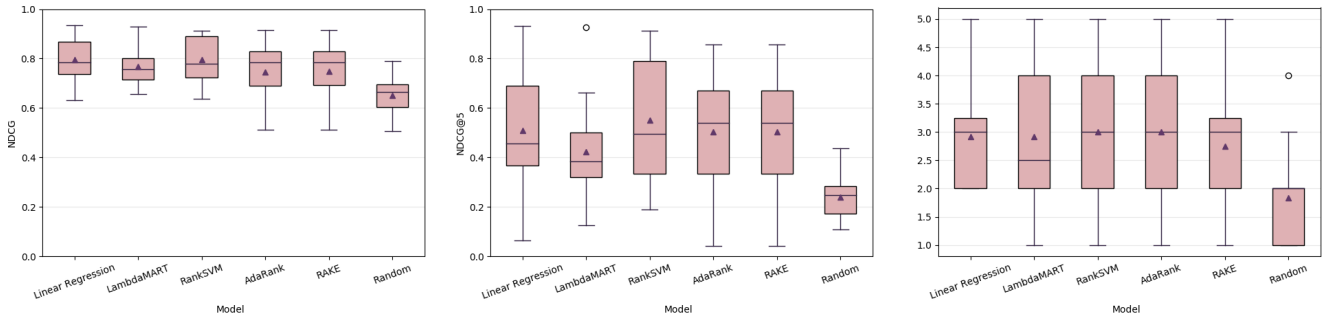
- *Text extraction*: For what ratio of papers can we find and download PDFs? How long does the extraction take?
- *Keyphrase extraction*: How do authors evaluate the keyphrases extracted for their own papers?⋆
- *Learning-to-rank*: Will the learning-to-rank methods with the defined features improve the ordering of keyphrases in the opinion of the authors and thus the representativeness of the final tag clouds of their research interests?⋆ Which learning-to-rank method performs best (if any)?⋆

The experiments are based on running the described process over the DBLP profiles of 21 professors in the faculty of the Computer Science Department at the University of Chile.

Text Extraction: We begin by attempting to extract the full-text of 1,091 papers listed in DBLP for the 21 professors. In terms of the papers for which we successfully extracted text, the average success rate was 58% (min: 27%, max: 88%); the variation is due to publishers with which individual professors tend to associate depending on their area. It takes 24 seconds to extract the text from each paper (min: 12 seconds; max: 88 seconds); on average the process – including text and keyword extraction – takes around 25 minutes per author.

Keyphrase Extraction: We next ask all 21 professors to score the top 100 extracted phrases on a Likert scale from 5 (very relevant) to 1 (not relevant at all) with respect to their research interests; we also ask the professors to suggest missing phrases; 12 professors respond to this call and provide 1,126 scores for keyphrases (some are left blank and are thus removed). In total, 576 keyphrases (51%) are assigned the minimum score of 1, while the other 550 keyphrases are distributed relatively equally for scores from 2–5; on the other hand, the professors suggested only a handful of additional keyphrases missing from the list.

Learning-to-Rank: We use the labelled dataset collected in the previous phase to train learning-to-rank models and generate tag clouds. Based on default values and initial experiments, we configure Ranking SVM with a maximum of 1,000 iterations; LambdaMART with 2,000 estimators and a learning-rate of 0.03, optimising for Normalised Discounted Cumulative Gain (NDCG); and AdaRank with a maximum of 100 iterations, a minimum of 30 iterations, and the NDCG metric. We also compare these four learning-to-rank approaches with those for baselines based on (i) a random ordering of keyphrases in the top-500 results of RAKE, and (ii) ordering based on the RAKE scores themselves. The NDCG and NDCG@5 results for cross-validation – testing for each professor the result of the model trained over the other 11 professors – are given in Figures 2a and 2b, respectively. We see that all methods outperform the random baseline. On the other hand, learning-to-rank methods have a relatively minor benefit over the unsupervised RAKE score, with AdaRank, in particular, offering a near identical ordering to RAKE; the



(a) NDCG comparison (b) NDCG@5 comparison (c) Final scores

Fig. 2: Evaluation of keyword extraction and learning-to-rank methods

best overall scores are given by RankingSVM and Linear Regression, though the gap to other methods is minor.

Finally we again ask all 21 professors to rate their tag clouds on a Likert scale of 5 (very good) to 1 (very poor). All tag clouds are presented together on the same page and in random order. We render the tag clouds in grey-scale and use a fixed neutral font to minimise factors not relating to the ordering (which we wish to evaluate) affecting the results. We include the same configurations, training protocols and baselines as before. The results are shown in Figure 2c where the best performing methods are AdaRank and RankingSVM. In general, the professors' scores tended to settle around a neutral mean/median, though we also see extremes in both directions. We believe that the results are affected by the area of the professor, their tendency to work on multiple topics, and their varying preferences in features; for example, some professors commented that they would prefer tags from more recent topics; another dimension of disagreement was on the level of specificity/generality of the tags, where "`computer science`" was rated 5 by some professors and 1 by others.

Taking the final results into account, we design the PUBTAG system to allow users to select their preferred tag cloud from a set produced by a number of learning-to-rank methods, and to further customise it by adding, removing and reordering tags; visual aspects can also be configured. Users may also download raw keyphrases for import into other visualisations.

V. CONCLUSION

We propose a method for generating tag clouds of research interests from a personal publication record based on keyphrase extraction and learning-to-rank models. We implement a prototype system called PUBTAG, which generates tag clouds for DBLP profiles. We evaluate the approach for 12 Computer Science professors. The process takes around 25 minutes per author. Learning-to-rank models show slight improvement versus unsupervised keyword extraction scores. The final tag cloud evaluations tend towards a neutral score.

For code, data and results we refer the reader to the following project: https://github.com/burningreds/cstagclouds. A demo of PUBTAG is available at http://pubtag.dcc.uchile.cl.

REFERENCES

[1] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "KEA: Practical Automatic Keyphrase Extraction," in *ACM Conference on Digital Libraries (ACM DL)*, 1999, pp. 254–255.

[2] M. A. Hearst and D. K. Rosner, "Tag Clouds: Data Analysis Tool or Social Signaller?" in *Hawaii International International Conference on Systems Science (HICSS-41)*. IEEE Computer Society, 2008, pp. 160–170.

[3] M. Gupta, R. Li, Z. Yin, and J. Han, "Survey on social tagging techniques," in *ACM SIGKDD Explorations Newsletter*. ACM, 2010, pp. 58–72.

[4] K. S. Hasan and V. Ng, "Automatic Keyphrase Extraction: A Survey of the State of the Art," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014, pp. 1262–1273.

[5] A. Medelyan, "NLP keyword extraction tutorial with RAKE and Maui," online, 2014, https://www.airpair.com/nlp/keyword-extraction-tutorial.

[6] T. Liu, "Learning to Rank for Information Retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[7] H. Li, "A Short Introduction to Learning to Rank," *IEICE Transactions*, vol. 94-D, no. 10, pp. 1854–1862, 2011.

[8] X. Jiang, Y. Hu, and H. Li, "A ranking approach to keyphrase extraction," in *ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2009, pp. 756–757.

[9] T. Joachims, "Training linear SVMs in linear time," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2006, pp. 217–226.

[10] K. Eichler and G. Neumann, "DFKI KeyWE: Ranking Keyphrases Extracted from Scientific Articles," in *International Workshop on Semantic Evaluation (SemEval@ACL)*. ACL, 2010, pp. 150–153.

[11] C. Wang and S. Li, "CoRankBayes: bayesian learning to rank under the co-training framework and its application in keyphrase extraction," in *ACM Conference on Information and Knowledge Management (CIKM)*, 2011, pp. 2241–2244.

[12] F. B. Viégas and M. Wattenberg, "Timelines - tag clouds and the case for vernacular visualization," *Interactions*, vol. 15, no. 4, pp. 49–52, 2008.

[13] B. Y.-L. Kuo, T. Hentrich, B. M. . Good, and M. D. Wilkinson, "Tag clouds for summarizing web search results," in *International World Wide Web Conference (WWW)*. ACM, 2007, pp. 1203–1204.

[14] G. Koutrika, Z. M. Zadeh, and H. Garcia-Molina, "Data clouds: summarizing keyword search results over structured data," in *International Conference on Extending Database Technology (EDBT)*. ACM, 2009, pp. 391–402.

[15] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining*, pp. 1–20, 2010.

[16] C. J. Burges, "From RankNet to LambdaRank to LambdaMART: An Overview," Microsoft, Tech. Rep., June 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/

[17] J. Xu and H. Li, "AdaRank: a boosting algorithm for Information Retrieval," in *SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2007, pp. 391–398.