Ontology-Based Categorization of Web Services with Machine Learning

Adam Funk and Kalina Bontcheva
University of Sheffield, Department of Computer Science
Regent Court, 211 Portobello, Sheffield, S1 4DP, UK
a.funk,k.bontcheva@dcs.shef.ac.uk

28th October 2009

1 Introduction

The Service-Finder project aims to address the problem of web service discovery for a wide audience through a portal[1] which presents automatic semantic descriptions of a wide range of publicly available Web Services and also enables service consumers (not just providers) to enrich them according to Web 2.0 principles. In this project, the Service Crawler carries out focused crawling for web services and archives WSDL files and related HTML files, then passes batches of these data to the Automatic Annotator (AA), which analyses them to produce semantic annotations to the semantic repository and back end for the web portal. [3]

This paper focuses on a specific AA problem: automatically categorizing web services from the crawled documents so that portal users can find them effectively. Although users can improve the categorizations, we must provide some good ones so that users will find the portal useful and contribute (instead of facing about 23 000 uncategorized services with only keyword searching).

The project ontology contains 59 subclasses of *Category* for web services, such as *Genetics*, *SMS*, and *Media Management*, arranged in a shallow tree (down to three levels below the top class) in seven main branches (such as *Business* and *Science*). Each service can be annotated with more than one category. (See [3] for further details.)

Since we originally had no manually annotated data to use for training but needed to generate category annotations quickly for the first release of the portal, we devised a weighted voting system based on keywords; this was relatively easy to integrate with the pipeline that was already being developed using GATE [2] for other IE purposes.

After the first release of the portal, members of the project manually annotated 224 services through the portal to produce 387 category annotations, which we used to

[1] http://demo.service-finder.eu/

evaluate the keyword-based system and as training data for machine learning (ML). At that point, we did not have enough annotations from outside users to make a significant contribution, but in the future, their annotations will also be used the same way.

2 Keywords and weighted voting

For our first approach to categorization, we devised an *ad hoc* gazetteer of keywords and phrases associated with service categories (the category names, parts of the multiword names, morphological variations (such as plurals), synonyms, and related words obtained by examining the documents), and weighted the values of the keywords using JAPE rules [1] to associate each match with a multiplier m which starts at 1 and varies up to 5 if it occurs in a significant place such an HTML `<title>` or heading element. Each match is later treated as m votes for the relevant category for the document's services. These votes are compiled for every service, which is then labelled with the two highest-scoring categories.

We used the manually annotated data to evaluate this component using the traditional IE measures. The first two versions of the tool scored about $P = 36\%$, $R = 15\%$; extending the gazetteers produced $P = 20\%$ and increased $R = 32\%$. We also evaluated the third version according to the Balanced Distance Metric (BDM), which gives partial credit for getting a superclass or subclass of the key, as well as for hitting another node on the same branch of the class tree; these *augmented* measures are better for ontological classification [9]. We obtained $AP = 38\%$ and $AR = 40\%$, which show that the system makes a significant number of "near misses", but we were not satisfied with these results. We ruled out language bias (99% of the abstracts and 87% of the HTMLs were in English) and chose a different approach.

3 Machine learning

We treat this as a text classification problem and use the Support Vector Machine (SVM) technique, which is well documented for this purpose. SVM text classification is carried out using n-grams of features of sequential units of text (such as unmodified tokens, part-of-speech tags, orthographic features, or lemmata), but previous research indicates that increasing n from 1 to 2 slows down performance and rarely improves the results, and increasing n beyond 2 typically deteriorates the results. [7, 8, 4]

SVM ML can learn and apply only one class to each instance (here, each document) so we had to simplify the training data by using the most specific class in the category ontology for each document, although we then devised a way to generate more than one category for a service.

We carried out several document classification experiments on the manually annotated services using standard parameters for GATE's SVM tool, treating each document as a bag of words) and one service category per document as the target class, one-against-others classification, and a linear kernel. (See [8] for the technical details of the parameters. All the ML experiments described here are evaluated with 4-fold cross-validation.)

Table 1: ML trials: scoring one category per document

Documents	Threshold	traditional %			BDM %		
		P	R	F_1	AP	AR	AF_1
Abstracts	0.1	27.2	29.5	28.3	29.5	29.5	29.5
Abstracts & HTMLs	0.1	47.8	47.8	47.8	66.8	66.8	66.8
Abstracts & HTMLs	0.3	68.0	40.9	50.7	74.1	51.0	60.4
	0.5	85.3	19.8	31.4	88.1	29.6	44.3
	0.8	97.9	8.1	14.7	98.3	17.7	30.0

Table 2: ML trials: scoring services after voting from the documents

Documents	Threshold	Max.	traditional %			BDM %		
		cat.	P	R	F_1	AP	AR	AF_1
Abstracts & HTMLs	0.1	1	54.3	49.8	52.0	74.7	51.7	61.1
	0.1	2	35.0	52.6	42.0	50.0	55.7	52.7
	0.3	1	58.1	52.8	55.3	79.1	55.0	64.9
	0.3	2	49.8	53.5	51.6	68.8	56.5	62.0

The first experiment used 0.1 as the classification probability threshold, first over 229 abstracts (one per service) and then over 1019 documents (including the abstracts). The first part of Table 1 shows the traditional and BDM measures for document classification.

The performance was much better for both abstracts and HTML documents so we experimented further with this combination. (Some services have no HTMLs or only a few small ones, but every one has an abstract with some content.) Table 1 shows the results for document classification with various classification threshold values. One normally expects increasing the threshold to increase P and decrease R, but the effect above 0.3 was quite severe, so we continued with only the low threshold values. (We consider R important because it is beneficial to try to provide at least one approximate category for every service on the portal; otherwise users are less likely to find the service at all and then improve the categorization.)

Those evaluations are scored with one key and one response category for each document, the key being the single manual category explained above. But the ML tool can assign different categories to documents related to the same service, so we let each service's documents vote to assign one or two categories to it, and scored services, to obtain the results in Table 2.

We have two explanations for the drop in P and F_1 when each service has two categories. First, if a service has only one manual annotation but gets the correct automatic label as well as its superclass, the second label is scored as spurious and the service gets a 50% score. For this work's goal (helping users find services on the portal) we do not

consider this a problem. Second, some services with few documents received only one automatic category even when two categories were allowed; these automatically received one correct category but lacked the second one and so get a 50% score.

4 Related work and conclusions

We have examined other published research about semantic annotation of web services and found one set of papers for a comparable task: the research for the ASSAM (Automated Semantic Service Annotation with Machine Learning) [6, 5] tool, which uses machine learning (Naïve Bayes and SVM classification) on combinations of bags of words from UDDI descriptions, WSDL files, and services' input and output messages to classify two sets of services according to ontologies with 11 and 26 categories, respectively, with accuracy up to around 75%.

Although our results so far are not as high as ASSAM's best, our task is more difficult: our service ontology has twice as many classes and a service can belong to any number of them. We therefore consider our results so far to be very good for our purpose. Another useful result of these experiments for Service-Finder has been to show that it is not worth pursuing incremental improvements using keywords, since it was relatively easy to get significantly better results with ML, which will (unlike the first approach) make use of the portal's collaborative tools in the future.

In our continuing work, we are experimenting with using separate classifiers for abstracts and HTML documents. (Using separate classifiers for different types of input improved the results for [5].) We are also developing an ontologically aware voting system to avoid assigning near-duplicate categories such as a class and its superclass.

Acknowledgements

This research is partially supported by the European Union's Seventh Framework Program project Service-Finder (FP7-215876).

References

[1] H. Cunningham, D. Maynard, and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November 2000.

[2] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[3] Emanuele Della Valle, Dario Cerizza, Irene Celino, Andrea Turati, Holger Lausen, Nathalie Steinmetz, Michael Erdmann, and Adam Funk. Realizing Service-Finder:

Web service discovery at web scale. In *European Semantic Technology Conference (ESTC)*, Vienna, September 2008. URL `http://videolectures.net/estc08_valle_rsf/`.

[4] A. Funk, Y. Li, H. Saggion, K. Bontcheva, and C. Leibold. Opinion analysis for business intelligence applications. In *First international workshop on Ontology-Supported Business Intelligence (at ISWC)*, Karlsruhe, October 2008. ACM.

[5] A. Heß and N. Kushmerick. Machine learning for annotating semantic web services. In *AAAI Spring Symposium on Semantic Web Services*, March 2004.

[6] A. Heß, E. Johnston, and N. Kushmerick. ASSAM: A tool for semi-automatically annotating web services with semantic metadata. In *International Semantic Web Conference (ISWC)*, Hiroshima, November 2004.

[7] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *10th European Conference on Machine Learning (ECML-98)*, Chemnitz, Germany, 1998.

[8] Y. Li, K. Bontcheva, and H. Cunningham. SVM Based Learning System for F-term Patent Classification. In *Proceedings of the Sixth NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access*, pages 396–402, May 2007. URL `http://gate.ac.uk/sale/ntcir6-papers/ntcir6-classification-main.pdf`.

[9] Diana Maynard, Yaoyong Li, and Wim Peters. NLP Techniques for Term Extraction and Ontology Population. In P. Buitelaar and P. Cimiano, editors, *Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text*. IOS Press, 2008.