Ontology-Based Categorization of Web Services with Machine Learning

Adam Funk, Kalina Bontcheva

Department of Computer Science
University of Sheffield
Regent Court, Sheffield, S1 4DP, UK
a.funk@dcs.shef.ac.uk, k.bontcheva@dcs.shef.ac.uk

Abstract

We present the problem of categorizing web services according to a shallow ontology for presentation on a specialist portal, using their WSDL and associated textual documents found by a crawler. We treat this as a text classification problem and apply first information extraction (IE) techniques (voting using keywords weight according to their context), then machine learning (ML), and finally a combined approach in which ML has priority over weighted keywords, but the latter can still make up categorizations for services for which ML does not produce enough. We evaluate the techniques (using data manually annotated through the portal, which we also use as the training data for ML) according to standard IE measures for flat categorization as well as the Balanced Distance Metric (more suitable for ontological classification) and compare them with related work in web service categorization. The ML and combined categorization results are good and the system is designed to take users' contributions through the portal's Web 2.0 features as additional training data.

1. Introduction

The web is no longer just a set of documents; services now feature prominently, and both research and industry devote considerable effort and interest to service-oriented architecture (SOA), web services, and related technologies. In a web of services or a service-oriented-architecture, discovery is an essential task for building and using applications. Unfortunately the widely used service description techniques only cover the syntactic level. Previous research projects showed that semantic descriptions (such as OWL-S, WSMO, and SAWSDL) can enable precise service discovery, but they are not widely deployed (Della Valle et al., 2008).

The main existing solution for service discovery is UDDI (Universal Description, Discovery and Integration[1]), a standard for programmatically publishing and retrieving structured information about web services. Some companies operate public UDDI repositories, but its success has not been widespread because of complicated registration, missing monitoring facilities, and other difficulties. A few portals have been designed to act as web service repositories, but they all rely on manual registration and review, so coverage is limited and information falls out of date easily. Furthermore, the standard web search engines do not provide effective ways to search for services or to allow filtering according to availability and service parameters.

The Service-Finder project aims to address this problem for a wider audience by offering a comprehensive framework for service discovery through a portal[2], so we must address the problem of creating semantic descriptions of Web Services. The TAO project[3] dealt with semi-automatic creation of semantic service descriptions, but focused on legacy applications and relied on the existence of substantial software documentation. Service-Finder offers automatic creation of service descriptions for a wide range of publicly available services and enables service consumers (not just providers) to collaboratively enrich the semantic descriptions according to Web 2.0 principles.

2. The research problem

The information flow in Service-Finder is structured as follows. The Service Crawler (SC) 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 processes them using information extraction techniques to produce semantic annotations for the Conceptual Indexer and Matcher (CIM). The CIM acts as a semantic repository and back end for the Service-Finder Portal (SFP)—it answers the latter's queries and records users' annotations in the repository along with the automatic annotations. The Clustering Engine (CE) also provides recommendations by clustering users and services based on the users' behaviour and interests (Brockmans et al., 2009; Della Valle et al., 2008).

This paper focuses on a specific problem in Service-Finder: automatically categorizing web services so that portal users can find them effectively. Although users can improve the category annotations, the aim is to provide reasonably good ones to start with so users will find the portal useful and contribute—so they do not have to face 23 000 uncategorized services with only keyword searching.

[1]`http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm`

[2]`http://demo.service-finder.eu/`

[3]`http://www.tao-project.eu/`

Input from SC to AA		
Number of `.arc.gz` files		5
Total compressed size		441 MB
Number of documents		250 k
Preprocessing		
Suppressed	HTTP status	49k
	unwanted provider IDs	5k
	empty documents	3k
	duplicates reduced	23k
Preprocessing results		
Documents	HTML	37k
	WSDL	110k
	abstract	25k
total (31% reduction)		173k
Output from AA to CIM		
Number of RDF-XML files		30
Total compressed size		40 MB
Number of RDF triples		4.5 M
Number of Providers		8700
Number of Services		25 000

Table 1: Typical AA input, preprocessing, and output

2.1. Data

A batch of AA input from the SC consists of several files in the Heritrix[4] Internet Archive format; Table 1 summarizes one of the last batches processed. The SC also provides an index which associates each document with one service identifier. (The services and providers are identified by URIs, and a service's URI can be easily converted to the relevant provider's URI.)
The AA's task is to extract information (such as contact addresses), classify documents and services, and semantically annotate the services, providers, and interesting documents, using tools built on the GATE (Cunningham et al., 2002) infrastructure. This paper presents work on the service categorization task.
Table 1 also shows the results of preprocessing the archive files to produce datastores of GATE documents (relating to about 8 000 providers and 23 000 services). The preprocessor merges exact duplicate documents[5], suppresses WSDL documents that the XML parser rejects, and suppresses documents whose provider URIs indicate that the services are not usable from the crawled WSDL files because they contain invalid endpoints (they have been incorrectly generated or the endpoints have been deliberately edited out of the publicly available versions).
The AA uses the WSDL documents to generate instances of *Endpoint*, *Interface*, and *Operation*, and to instantiate relations between them and the *Service* instances, but carries out very little textual information extraction (IE) on them. The *abstracts* are compiled by the SC from various textual elements and attributes in the WSDL files (service name, documentation, operation names, input and output parameters, etc.). The SC generates one abstract for each service URI, choosing the best WSDL file associated with the service (typically the shortest WSDL URL containing the provider name). The AA carries out textual IE on the abstracts as well as the HTML files.

2.2. The categorization task

The category ontology used in the project contains 59 subclasses of *Category* arranged in a shallow tree (with some multiple inheritance) under seven main branches, as shown in Figure 1. Each service can be annotated with more than one category, as shown in Figure fig:categories. (See (Della Valle et al., 2008; Brockmans et al., 2008) for further details of the ontology design.)
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 AA pipeline which was already being developed to apply gazetteers and rules to the documents for other purposes. §3.1 explains this approach in more detail and presents the evaluation results.
After the first release of the portal, members of the project manually annotated 224 services through the portal to produce 387 category annotations (using 45 of the service categories), which we exported to use for evaluation of 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 exported and used in the same way. §3.2 explains and evaluates the ML techniques used.

3. Methodology

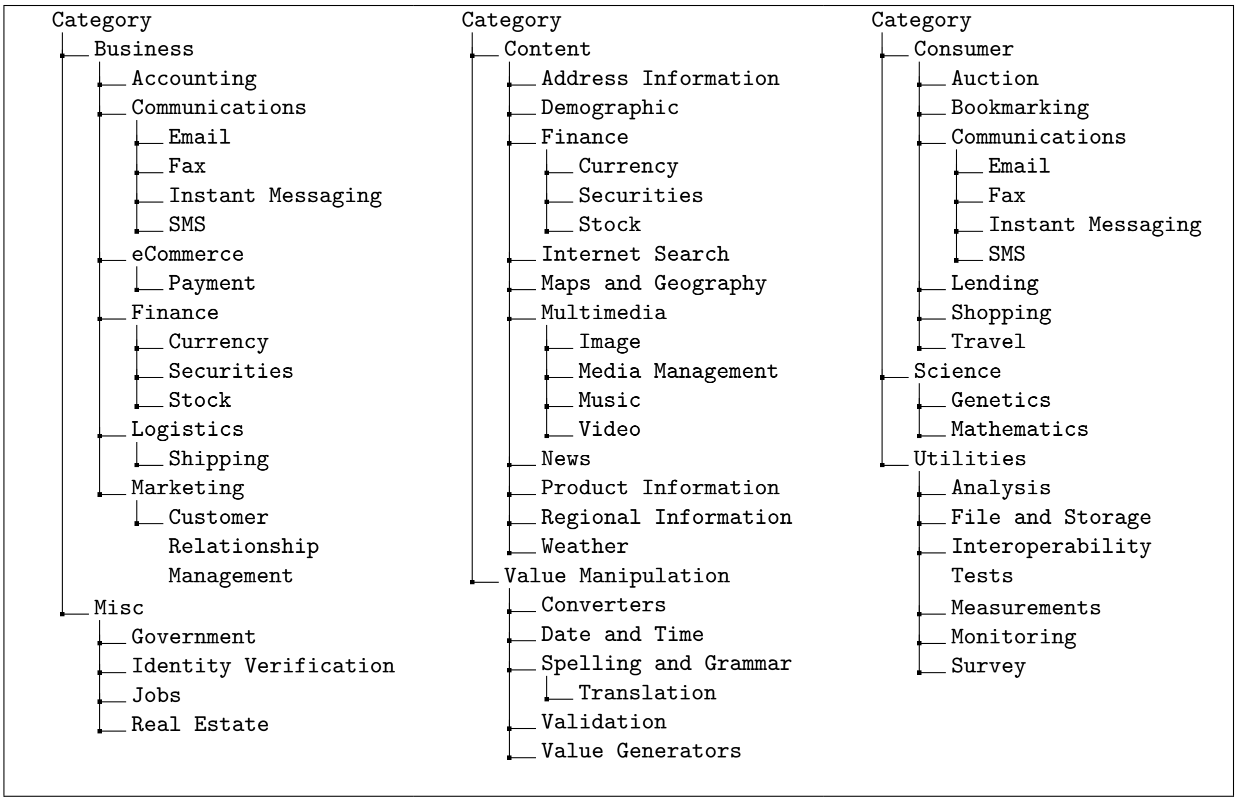
3.1. Keywords, rules, and multipliers

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); Table 2 shows examples for three of the categories. This gazetteer processor was configured to be case-insensitive but to match whole words only (for example, *Address* would match part of *address list* but nothing in *headdress*).
Because some place where keywords can appear are more significant than others, JAPE rules[6] later in the pipeline create special annotations over the gazetteer

[4]`http://crawler.archive.org/`

[5]Each provider is associated with one or more services. Each input document is associated by the SC with one service, but merged documents that refer to different services (of the same provider) are associated with all of them. Each abstract relates to only one service.

[6]JAPE processors in GATE compile rules that specify regular expressions of annotations and create additional annotations, manipulate features, and execute Java code when matches are detected. (Cunningham et al., 2000)



```
Category
  Business
    Accounting
    Communications
      Email
      Fax
      Instant Messaging
      SMS
    eCommerce
      Payment
    Finance
      Currency
      Securities
      Stock
    Logistics
      Shipping
    Marketing
      Customer
      Relationship
      Management
  Misc
    Government
    Identity Verification
    Jobs
    Real Estate

Category
  Content
    Address Information
    Demographic
    Finance
      Currency
      Securities
      Stock
    Internet Search
    Maps and Geography
    Multimedia
      Image
      Media Management
      Music
      Video
    News
    Product Information
    Regional Information
    Weather
  Value Manipulation
    Converters
    Date and Time
    Spelling and Grammar
      Translation
    Validation
    Value Generators

Category
  Consumer
    Auction
    Bookmarking
    Communications
      Email
      Fax
      Instant Messaging
      SMS
    Lending
    Shopping
    Travel
  Science
    Genetics
    Mathematics
  Utilities
    Analysis
    File and Storage
    Interoperability
    Tests
    Measurements
    Monitoring
    Survey
```

Figure 1: Service categories ontology

Genetics	*News*
Genetics	News
genetic	current events
bioinformatic	*Address Information*
bioinformatics	Address Information
arabidopsis	Address
protein	Addresses
proteins	Addressing
RNA	postcode
DNA	zip code
MeSH	phone
	telephone

Table 2: Examples of keywords from the gazetteer for three categories

matches, each of which has a multiplier m which starts at 1 and is increased according to the following rules:

- +3 if the match is in an HTML `<title>`, `<h1>`, `<h2>`, `<h3>`, or `<h4>` element;
- +2 if the match is in a `<p>` element that contains a significant number of interesting keywords (from various gazetteers in the application);
- +2 if the document has an *interesting* value[7] $2.0 \leq i$;
- +1 if the document has an *interesting* value $1.0 \leq i < 2.0$.

Each category keyword match in a document is later treated as m (the multiplier) votes for the relevant category for each service associated with the document. These votes are compiled for every service, which is labelled with the two highest-scoring categories. (This arbitrary limit was agreed within the project.)

We used the manually annotated data described in §2.2 to evaluate the AA's categorization of those services. Table 3 shows the traditional IE measures[8] for the three major datasets (which we call Crawls 1, 2, and 3) that have been processed through Service-Finder, as well as the Balanced Distance Metric (BDM) (Maynard et al., 2006) for the latest one. The traditional measures count each categorization as wrong if it is not an exact match, even if the key (manual) and response (automatic) categories are close in the ontology tree. BDM scoring, however, counts 1 for an exact match but 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. It counts a response as 0 only if its lowest superclass in common with the key is the top class. These *augmented* precision, recall, and F_1 measures are more suitable for ontology-based classification (Maynard et al., 2008).

The BDM scores for Crawl 3 are noticeably higher than the traditional ones; this indicates that the system is categorizing quite a few services with "nearly

[7] The IE pipeline adjusts each document's *interesting* feature, initially 0, between 0 and 3 according to the information it finds.

[8] Precision, recall, and F_1 (Manning and Schütze, 1999).

Crawl	traditional %			BDM %		
	P	R	F_1	AP	AR	AF_1
1	35.8	16.3	22.4			
2	39.3	12.4	18.9			
3	20.0	31.6	24.5	38.4	40.3	39.3

Table 3: Evaluation of keyword-based service categorization (limited to two categories per service)

Type	Language		%
Abstracts	English	24 181	99.3
	Spanish	35	0.1
	Romanian	29	0.1
	Czech	20	0.1
	other	94	0.4
HTMLs	English	36 248	87.0
	unclassifiable	3 959	9.5
	German	426	1.0
	Spanish	131	0.3
	French	99	0.2
	other	825	2.0

Table 4: Summary of language identification results

correct" classes from the category ontology. These results are still not very good, however. After Crawl 2, we improved recall by adding more keywords to the gazetteers, but this made precision deteriorate significantly. We had suspected that the documents in other languages than English might pose a problem, but we added an automatic language identification tool based on character n-grams (Schutz, 2008) to count the number of abstracts and HTML documents by language, as shown in Table 4 and concluded that adding translations to the gazetteers would not be fruitful. Instead, we pursued experiments in service categorization based on machine learning.

3.2. 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. (Joachims, 1998; Li et al., 2004; Li et al., 2007) 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. (Li et al., 2004; Pang et al., 2002; Funk et al., 2008)

One disadvantage of SVM ML is that it can learn and apply only one class to each instance (document, in this case), so we had to simplify the problem for services with more than one manually annotated category by using the lowest (most specific) class in the category ontology (or making an arbitrary choice between two equal categories). As we shall see later, however, it is possible to generate more than one category for a service with multiple documents.

We carried out several document classification experiments on the manually annotated services using standard parameters for GATE's SVM tool, with unigrams of tokens[9] as the instance attributes (i.e., 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 (Li et al., 2004; Li et al., 2007) for the technical details of the parameters.) All the ML experiments described here are evaluated with 4-fold cross-validation, which makes better use of the available corpus. The first experiment used 0.1 as the classification probability threshold, first over 224 abstracts (one per service) and then over 1019 documents (including the abstracts). The first part of Table 5 shows the traditional and BDM measures for document classification.

We noted that the performance was much better for both abstracts and HTML documents, and decided to carry out further experiments using this combination. (Working with HTMLs only is not practical since quite a few services have no HTMLs or only a few small ones, whereas every service has an abstract with some content.) Table 5 shows the results for document classification with various values for the classification threshold (the minimum probability required for the SVM to assign a classification). One normally expects increasing the threshold to increase precision and decrease recall, but the effect above 0.3 was quite severe here, so we decided to continue with just the two low threshold values. (We consider recall 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.)

The evaluation results presented so far in this section are scored per document, with one key and one response category for each document, the key being the single category selected from the manual annotations (as explained above). But the ML tool can assign different categories to documents related to the same service, so we now consider letting each service's documents vote (with equal weighting) to assign one or

[9]We used the standard ANNIE tokenizer for natural language on the HTML documents and a *source code tokenizer* on the abstracts. The second one splits camel-cased strings (e.g., `getUnsplicedSequence` → `get Unspliced Sequence`) as well as tokens separated by whitespace and inter-word punctuation.

Crawl	Documents	Threshold	traditional %			BDM %		
			P	R	F_1	AP	AR	AF_1
3	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
		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 5: ML trials: scoring one category per document

Crawl	Documents	Threshold	Max. cat. per service	traditional %			BDM %		
				P	R	F_1	AP	AR	AF_1
3	A&H	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

Table 6: ML trials: scoring services based on voting from the documents

two categories to it.[10] Table 6 presents the traditional and BDM measures for service categories derived from this process, for thresholds of 0.1 and 0.3 and for one or two categories per service.

It is surprising to see the precision and F_1 drop when each service has two categories, but we can explain this in two ways. First, many of the services had only one manual annotation, so the second automatic one counts as a spurious classification, even if it is closely related: for example, several services were automatically labelled with *Genetics* and its direct superclass *Science*, although the manual annotators only marked them as *Genetics*—these services get a 50% score. Since the ultimate purpose of this work is to help users find services on the portal, however, we do not consider this a significant fault. Second, some services which had very few documents received only one automatic category even when two categories were allowed (because all the documents were labelled with the same category); several such services automatically received one correct category but lacked the second one—these too get a 50% score.

3.3. Integrated approach

We then integrated the machine learning and keyword-based categorization components in the AA pipeline in such a way that ML categorizations always overrule keyword-based ones, but the latter can still be used if necessary to make up to one or two categories per service. (The ML categorization is turned into a votable annotation on each document with a multiplier $m = 100$, sufficient to outvote keywords' annotations.) This produced the results shown in Table 7. The measures are slightly lower but this approach gives wider coverage of the services and categories (only 45 of the 59 categories were represented in the manually annotated data used for training the ML component).

[10]If an HTML document can be associated with more than one service, its categorization votes for each relevant service.

4. Related work

As a result of the proliferation of web services and growing interest in web service discovery, browsing, composition, and related matters in recent years, several approaches have been investigated in service classification.

Belhajjame & al. (Belhajjame et al., 2008) presented an algorithm to automatically annotate web services with information to support service discovery and composability (in particular, to allow data-driven composition of workflows). This work achieved good results for a very detailed type of annotation (concentrating on the service and operation details) over a limited domain (bioinformatics), but does not really apply to our broad coverage.

The *Woogle* web-service search engine included a clustering system (Dong et al., 2004) based on parameter names into meaningful groups to allow similarity search. This work also achieved good results, although it would not be applicable to the Service-Finder task, which uses *a priori* categories. Woogle's approach was also dependent on UDDI repositories, which (as explained in §1) we cannot rely on.

The ASSAM (Automated Semantic Service Annotation with Machine Learning) (Heß et al., 2004) tool loads WSDL files along with previously created ontologies and provides a GUI for the user to annotate the services, and learns from previous annotations in order to show suggestions and make manual annotation faster and easier. This system uses the authors' work on web service annotation with machine learning (Hess and Kushmerick, 2003; Heß and Kushmerick, 2004), which is related to our approach presented here. That work used 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. The authors obtained accuracy up to around 75%, and noted that it is beneficial to offer users a choice from

Crawl	Documents	Max. cat. per service	traditional % P	R	F_1	BDM % AP	AR	AF_1
3	A&H	1	49.3	45.9	47.6	69.6	48.3	57.0
		2	32.3	55.6	40.9	47.0	58.9	52.3
4	A&H	2	32.0	55.4	40.6	47.4	59.1	52.6

Table 7: Evaluation of combined service categorization by voting from the document classifications (ML always overrules keywords; ML threshold = 0.1)

a small number of predicted categories (this is similar to our point that "near miss" categories are still useful for our portal users); their best result for this purpose was to get the correct category in the top three predictions 82% of the time. It is also worth noting that they obtained better results by training and applying separate classifiers for the different types of input data. Although our results in Tables 6 and 7 are not as high as ASSAM's best (Heß and Kushmerick, 2004) our task is more difficult: our service ontology has twice as many classes and a service can belong to any number of them, although ML techniques only produce one label per instance. We currently deal with the second problem by treating each document rather than each service as an instance, and then collating the results and putting an arbitrary limit on the number of categories per service.

5. Conclusion

We consider the results in Table 6 and 7 to be very good for our purposes.

A useful result of these experiments for Service-Finder and in general has been to show that it may not be worthwhile to pursue incremental improvements in the first classification approach (using keywords and rules), since it was relatively easy to get significantly better results for this task with machine learning. Furthermore, training from manual annotations—unlike the first approach—can also take advantage of the portal's collaborative (Web 2.0) tools.

For future work in this type of classification task, we would be interested in testing several improvements. For the machine learning itself, experimenting with using separate classifiers for abstracts and HTML documents would be worthwhile, since using separate classifiers for different types of input texts significantly improved ASSAM's results (Hess and Kushmerick, 2003; Heß and Kushmerick, 2004).) It would also be useful to develop an ontologically aware voting system to avoid assigning near-duplicate categories: for example, if the top three voting results for a service are *Finance*, *Currency*, and *Logistics*, eliminate *Finance* (a direct superclass of *Currency*) and return only the other two, which are on separate branches.

Acknowledgements

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

6. References

Khalid Belhajjame, Suzanne M. Embury, Norman W. Paton, Robert Stevens, and Carole A. Goble. 2008. Automatic annotation of web services based on workflow definitions. *ACM Trans. Web*, 2(2):1–34.

S. Brockmans, M. Erdmann, and W. Schoch. 2008. Hybrid matchmaker and Service-Finder ontologies (alpha release). Deliverable D4.2, Service-Finder Consortium.

S. Brockmans, I. Celino, D. Cerizza, E. Della Valle, M. Erdmann, A. Funk, H. Lausen, W.Schoch, N. Steinmetz, and A. Turati. 2009. Service-Finder: First steps toward the realization of web service discovery at web scale. In *Interoperability through Semantic Data and Service Integration Workshop (ISDSI 2009) at SEBD*, Genova, Italy, June.

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

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. 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).*

Emanuele Della Valle, Dario Cerizza, Irene Celino, Andrea Turati, Holger Lausen, Nathalie Steinmetz, Michael Erdmann, and Adam Funk. 2008. Realizing Service-Finder: Web service discovery at web scale. In *European Semantic Technology Conference (ESTC)*, Vienna, September.

X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. 2004. Similarity search for web services. In *International Conference on Very Large Databases (VLDB)*, Toronto.

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

A. Hess and N. Kushmerick. 2003. Learning to attach semantic metadata to web services. In *Proceedings of the Second International Semantic Web Conference (ISWC 2003)*, pages 258–273. Springer.

A. Heß and N. Kushmerick. 2004. Machine learning for annotating semantic web services. In

AAAI Spring Symposium on Semantic Web Services, March.

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

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398 in Lecture Notes in Computer Science, pages 137–142, Chemnitz, Germany. Springer Verlag, Heidelberg.

Y. Li, K. Bontcheva, and H. Cunningham. 2004. An SVM Based Learning Algorithm for Information Extraction. Machine Learning Workshop, Sheffield. `http://gate.ac.uk/sale/ml-ws04/mlw2004.pdf`.

Y. Li, K. Bontcheva, and H. Cunningham. 2007. 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.

Christopher D. Manning and Hinrich Schütze. 1999. Evaluation measures. In *Foundations of statistical natural language processing*, chapter 8.1, pages 267–271. Cambridge, MA, MIT Press.

D. Maynard, W. Peters, and Y. Li. 2006. Metrics for evaluation of ontology-based information extraction. In *WWW 2006 Workshop on Evaluation of Ontologies for the Web (EON)*, Edinburgh, Scotland.

Diana Maynard, Yaoyong Li, and Wim Peters. 2008. 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.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment Classification Using Machine Learning Techniques. In *Proceedings of the 2002 Conference on EMNLP*, pages 79–86.

Alex Schutz. 2008. XtraK4Me: Extraction of keyphrases for metadata creation. GATE plug-in software, Semantic Information Systems and Language Engineering Group (SmILE), DERI, Galway, Ireland, August.