

Using Term-matching Algorithms for the Annotation of Geo-services

Miha Grcar¹, Eva Klien²

¹ Jozef Stefan Institute, Dept. of Knowledge Technologies, Jamova 39,
1000 Ljubljana, Slovenia
miha.grcar@ijs.si

² Institute for Geoinformatics, Robert-Koch-Str. 26–28,
48149 Münster, Germany
klien@uni-muenster.de

Abstract. This paper presents an approach for automating semantic annotation within service-oriented architectures that provide interfaces to databases of spatial-information objects. The automation of the annotation process facilitates the transition from the current state-of-the-art architectures towards semantically-enabled architectures. We see the annotation process as the task of matching an arbitrary word or term with the most appropriate concept in the domain ontology. The term matching techniques that we present are based on text mining. To determine the similarity between two terms, we first associate a set of documents [that we obtain from a Web search engine] with each term. We then transform the documents into feature vectors and thus transition the similarity assessment into the feature space. After that, we compute the similarity by training a classifier to distinguish between ontology concepts. Apart from text mining approaches, we also present two alternative techniques, namely hypothesis checking (i.e. using linguistic patterns such as “*term*₁ is a *term*₂” as a query to a search engine) and Google Distance.

Keywords: geo-services, semantic annotation, text mining, search engine querying, machine learning, term matching

1 Introduction and Motivation

This paper presents an approach for automating semantic annotation within service-oriented architectures that provide interfaces to databases of spatial-information objects. The automation of the annotation process facilitates the transition from the current state-of-the-art architectures towards semantically-enabled architectures. The techniques presented in this paper are being developed in the course of the European project SWING¹ which deals with introducing semantics into spatial-data infrastructures to support discovery, composition, and execution of geo-services.

¹ Semantic Web Services Interoperability for Geospatial Decision Making (FP6-026514) <<http://www.swing-project.org>>

In SWING, semantic annotation is understood as the process of establishing explicit links between geographic information that is served via OGC² services and the vocabulary defined in the domain ontology (i.e. the vocabulary of a specific geoinformatics community). Once the bridge between the two sides is established, the domain ontology can be employed to support all sorts of user tasks.

The main purpose of this paper is to present data mining techniques that facilitate the annotation process. The annotation process can be seen as the task of matching an arbitrary word or term with the most appropriate concept in the domain ontology. Most of the term matching techniques that we present are based on text mining (see Sections 3.1–3.2). To determine the similarity between two terms, we first associate a set of documents with each term. To get the documents, we query search engines, on-line encyclopaedias, dictionaries, thesauri, and so on (query being the term in question). We then transform the documents into feature vectors. By doing so, we transition the similarity assessment into the feature space. Several text mining approaches are at hand to compute the similarity in the feature space – either by computing centroids or by training classifiers to distinguish between ontology concepts. Apart from the techniques based on document similarity, we also present two alternative techniques, namely hypothesis checking (i.e. using linguistic patterns such as “*term*₁ is a *term*₂” as a query to a search engine; see Section 3.3) and Google Distance (see Section 3.4). All the techniques are demonstrated on a toy example from the domain of mineral resources.

2 Related Work

Several knowledge discovery (mostly machine learning) techniques have been employed for ontology learning tasks in the past [6]. Text mining seems to be a popular approach to ontology annotation because the text mining techniques are shown to produce relatively good results.

We reference much of the related work from the corresponding sections. In the context of text mining we discuss centroid computation and classification [1] (see Section 3.1.1), Support Vector Machines [11] (see Section 3.1.2), *k*-NN, and classification in general [8] (see Section 3.1.2). Apart from the text learning techniques we also deal with linguistic patterns introduced by Hearst [7] (see Section 3.3), and Google Distance [2] (see Section 3.4).

3 Baseline for the Annotation of Geo-services

Normally, geo-data is served by a database of spatial-information objects through a standardized interface. In SWING, we use OGC-defined standard interfaces, namely Web feature services (WFS) [9], to access spatial-information objects. Web feature services are required to implement the capability to describe objects (termed “features”) that they serve (see Fig. 1, the left-hand side). These descriptions

² Open Geospatial Consortium <<http://www.opengeospatial.org>>

(schemas) contain the definition of each available class of objects in a similar fashion as a data structure is defined in an object-oriented modeling or programming language: the schema provides the class name and its attributes; each attribute is described by its own name and the corresponding data type.

On the other hand we have real-world entities such as trees, rivers, minerals, quarries, and so on. These are modeled as axiomatized concept definitions in the form of a domain ontology that captures a specific view on the world (see Fig. 1, the right-hand side). The core idea is to employ the domain ontology for the discovery and composition of Web feature services, and also for the retrieval of spatial-information objects (i.e. for the invocation of Web feature services).

In order to support these user tasks, we need to establish a bridge between the WFS schema on one side and the domain ontology on the other. The process of establishing this link is called annotation. We see the annotation as a two-step process. The first step is a simple syntactic translation from a WFS schema description into the appropriate ontology language. We use WSML [3] as the ontology-description language in SWING. We call a WFS described in WSML a “feature-type ontology” (FTO). FTOs do not differ much from the original WFS descriptions apart from being formulated in a different description language.

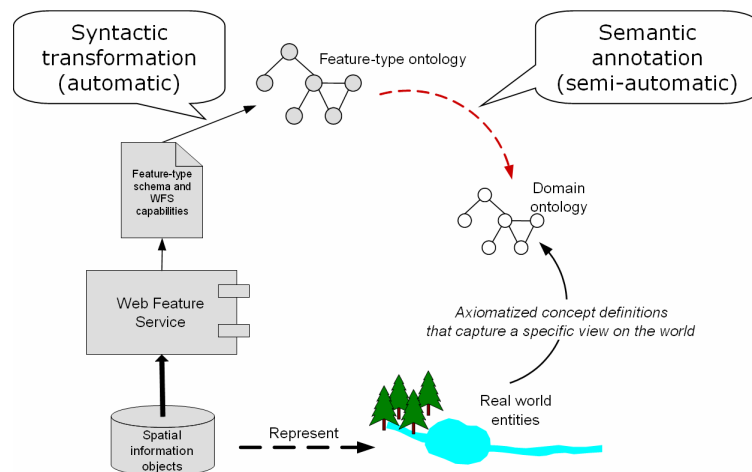


Fig. 1. The two-step semantic annotation process.

The first step thus establishes the syntactic compatibility between a WFS and the domain ontology (i.e. both descriptions are put into the same ontology-description language). However, the two descriptions are not yet semantically interlinked. The second step thus associates concepts and properties from FTOs with the domain ontology concepts. This process is described in the following section.

3 Automating the Annotation Process

Let us first define the problem of mapping one concept to another in more technical terms. We are given a feature-type ontology (FTO) concept as a single textual string (e.g. OpenPitMine) and a domain ontology which is basically a directed graph in which vertices represent concepts and edges represent relations between concepts. Each concept in the domain ontology is again given as a single textual string (e.g. D:Quarry³). The task is now to discover that OpenPitMine is more closely related to D:Quarry as to for instance D:Legislation or D:Transportation. Also important to mention is that every FTO concept has a set of attributes. Each attribute is given as a single textual string (e.g. OpenPitMine.SiteName) and has its corresponding data type (the data type is not expected to provide much guidance in the annotation process since it is usually simply *string*). Concepts in the domain ontology can similarly be described with the surrounding concepts, e.g. D:Quarry-hasLocation-QuarryLocation⁴.

A straightforward approach would be to try to compare strings themselves. Even by taking attribute strings into the account coupled with some heuristics we cannot hope for good results – this can serve merely as a baseline.

In the following we present several promising approaches that use alternative data sources (mostly the Web) to discover mappings between concepts. We limit ourselves to a scenario where attributes are not available (i.e. we are given merely a FTO concept and a set of domain ontology concepts). The task is to arrange domain ontology concepts according to the relatedness to the FTO concept. In the examples we will use OpenPitMine as the observed FTO concept and domain ontology concepts D:Quarry, D:Legislation, and D:Transportation.

In Section 3.1 we first introduce the idea of concept comparison by populating concepts with (textual) documents that reflect semantics of these concepts. To enable the realization of these ideas in the context of SWING we first need to resolve the fact that the concepts are not a-priori populated with documents. Section 3.2 presents two promising techniques of using a Web search engine (in our particular case: Google) to acquire the “missing” documents. Sections that follow (3.3 and 3.4) present two alternative ways of using the Web for the annotation. Rather than dealing with documents, these approaches deal with term co-occurrences and linguistic patterns, respectively.

3.1 Comparing Documents to Determine Concept Similarity

Suppose we have a set of documents assigned to a concept and that these documents “reflect” the semantics of the concept. This means that the documents are talking about the concept or that the domain expert would use the concept to annotate (categorize) these documents.

³ With prefix D: we denote concepts that belong to the domain ontology.

⁴ This denotes a domain ontology concept named Quarry with “attribute” QuarryLocation which is linked to the concept via the hasLocation relation.

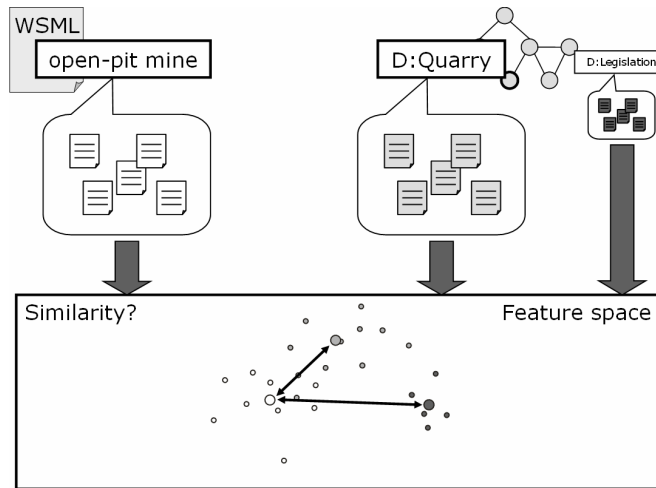


Fig. 2. Transitioning the similarity assessment into the feature space by transforming the documents into the corresponding *tfidf* feature vectors. This figure also illustrates how the comparison of centroids (discussed later on in Section 3.1.1) can be used to conclude that OpenPitMine (represented with white documents/dots) is associated with D:Quarry (represented with light gray documents/dots) stronger than with D:Legislation (represented with dark gray documents/dots). This conclusion is based on the fact that the white centroid is closer to the light gray centroid than to the dark gray centroid. The centroids are represented with the larger dots.

In such cases we can compute the similarity between two concepts. We are given a FTO concept (in our case OpenPitMine) and several domain ontology concepts (in our case D:Quarry, D:Transportation, and D:Legislation) with their corresponding document sets (see Fig. 2). We first convert every document into its bag-of-words representation, i.e. into the *tfidf* representation [6]. A *tfidf* representation is actually a sparse vector of word-frequencies (compensated for the commonality of words – this is achieved by the *idf* component – and normalized). Every component of a *tfidf* vector corresponds to a particular word in the dictionary. With “dictionary” we refer to all the different words extracted from the entire set of documents. If a word does not occur in the document, the corresponding *tfidf* value is missing – hence the term “sparse vector”. Each of these vectors belongs to a certain concept – we say that the vector is *labelled* with the corresponding concept. This gives us a typical supervised machine learning scenario. In the following subsections we present three different approaches to concept-concept similarity computation using different machine learning approaches.

Comparing Centroids to Determine Concept Similarity. A centroid is a (sparse) vector representing an (artificial) “prototype” document of a document set. Such prototype document should summarize all the documents of a given concept. There are several ways to compute the centroid (given *tfidfs* of all documents in the corresponding set). Some of the well-known methods are the Rocchio formula,

average of vector components, and (normalized) sum of vector components. Of all the listed methods, the normalized sum of vector components is shown to perform best in the classification scenario [1]. In the following we limit ourselves to the method of normalized sum. We first represent documents of a particular concept C as normalized *tfidf* vectors \vec{d}_i . Now we compute the centroid as given in Eq. 1.

$$\vec{c} = \frac{1}{\|\vec{c}\|} \sum_{\vec{d}_i \in C} \vec{d}_i . \quad (1)$$

Having centroids computed for all the concepts, we can now measure similarity between centroids and interpret it as similarity between concepts themselves (we are able to do this because a centroid summarizes the concept it belongs to). This is illustrated in Fig. 2. Usually we use cosine similarity measure [6] to measure similarity between two centroid vectors.

Table 1. The performance of some of the discussed algorithms on our toy example. In the case of the centroid-to-centroid similarity computation, the numbers represent cosine similarity between the terms; in the case of the Normalized Google Distance, the numbers represent the distance measure, and in the case of k -NN, the sum of all the cosine similarities measured between a FTO document and a domain ontology document from the corresponding neighborhood. The results are given for two different contexts: the general context and the context of “extracting material” (note that the contextualization is not applicable if Google definitions are used as the data source). Top 30 search results are considered when querying the search engine. Only English definitions and English search results are considered. In the case of k -NN, k is set dynamically by taking all the documents within the cosine similarity range of less than (or equal to) 0.06 into account. The number that represents the strongest association between the corresponding two terms is emphasized.

		open pit mine	
		general context	“extracting material”
Centroid Google search	quarry	0.11	0.39
	legislation	0.01	0.09
	transportation	0.02	0.05
Centroid Google definitions	quarry	0.39	N/A
	legislation	0.01	N/A
	transportation	0.05	N/A
k -NN Google search	quarry	0.61	2.82
	legislation	0	0.43
	transportation	0	0.10
k -NN Google definitions	quarry	3.17	N/A
	legislation	0	N/A
	transportation	0.35	N/A
NGD	quarry	0.02	1.92
	legislation	0.42	3.55
	transportation	0.50	1.71

Employing Classification to Determine Concept Similarity. We already mentioned that every *tfidf* vector is *labelled* with the corresponding concept and that this gives us a typical supervised machine learning scenario. In a typical supervised machine learning scenario we are given a set of training examples. A training example is actually a labelled (sparse) vector of numbers. We feed the training examples to a classifier which builds a model. This model summarizes the knowledge required to automatically assign a label (i.e. a concept) to a new yet unlabelled example (we term such unlabelled examples “test examples”). This in effect means that we can assign a new document to one of the concepts. We call such assignment (of a document to a concept) classification.

How do we use classification to compare two concepts? The approach is quite straightforward. We take the documents belonging to a particular FTO concept (in our case the documents of OpenPitMine) and strip them of their label thus forming a test set. Now we assign each of these documents to one of the domain ontology concepts (i.e. we classify each of the documents to one of the domain ontology concepts). The similarity between a FTO concept and a domain ontology concept is simply the number of FTO-concept documents that were assigned to that particular domain ontology concept. Many classifiers assign the same document to all the concepts at the same time but with different probabilities or confidence. We can compute the sum of these probabilities/confidence values instead of simply counting the documents.

There are many different classifiers at hand in the machine learning domain. Herein we discuss a very popular classifier – Support Vector Machine (SVM). In addition we also discuss how the same task is performed with the k -nearest neighbors (k -NN) algorithm which has the property of a “lazy learner”. The latter means that k -NN does not build a model out of the training examples – instead, it uses them directly to perform classification.

Classification with SVM. In its basic form, SVM is able to classify test examples into only two classes: positive and negative. We say that SVM is a binary classifier. This means that training examples must also be only of the two kind: positive and negative. Since examples are vectors, we can see them as points in a multi-dimensional space. The task of SVM is to find such hyper-plane that most of the positive training examples lie on one side of the hyper-plane while most of the negative training examples lie on the other side. Formally, SVM is an optimization problem that can be solved optimally. Recently it has been shown that this can actually be done in linear time for linear kernels [10], which is quite a breakthrough regarding the usefulness and quality of SVM.

Even though SVM is binary, we can combine several such classifiers to form a multi-class variant of SVM. Several multi-class variants are discussed and evaluated in [4].

Classification with k -NN. We already mentioned that k -NN is one of the “lazy learners” which means that it does not build a model out of training examples. It performs the classification of a document by finding k most similar documents of all the documents that belong to the domain ontology concepts. The similarity between the document and a domain ontology concept can be computed as the number of documents (from the set of k most similar documents) that belong to that domain ontology concept. Instead of simply counting the documents we can compute the sum of the corresponding cosine similarities. As an alternative to defining a constant

neighborhood size, we can set k dynamically by taking all the documents within the cosine similarity range of less than (or equal to) a predefined threshold into account.

3.2 Google Definitions and Contextualized Search Results

“If Google has seen a definition for the word or phrase on the Web, it will retrieve that information and display it at the top of your search results. You can also get a list of definitions by including the special operator ‘define:’ with no space between it and the term you want defined. For example, the search ‘define:World Wide Web’ will show you a list of definitions for ‘World Wide Web’ gathered from various online sources.” (excerpt from Google Help Center <<http://www.google.com/help/features.html#definitions>>)

Googlebots crawl the Web all the time. In their crusades they gather terabytes of data which is then processed in order to discover information that is potentially of particular interest to Google users (such as products for sale on-line, weather forecast, travel information, and images). One of such separately maintained information repositories are the definitions of words or phrases as found on the Web.

Google definitions can be used to compensate for the missing document instances – each definition (known by Google) can be seen as one document. In this way we can “populate” concepts with documents and then perform the mapping (i.e. the annotation) as already explained in Section 3.1.

To get back to our example, if we populate concepts OpenPitMine, D:Quarry, D:Transportation, and D:Legislation with document instances and then compare OpenPitMine (which is a FTO concept) to the domain ontology concepts (i.e. the other three concepts), we get centroid-to-centroid similarities as shown in Table 1. Since it is hard to find definitions for n -grams such as “open pit mine” (i.e. 3 or more words in a composition), we additionally query Google for the definitions of “pit mine” and “mine”, weighting the contribution of these definitions less than the one of the initial composed word (if the “complete” definition exists, that is).

There are still some issues that need to be considered when using this approach. For one, a word can have several meanings, i.e. its semantics depends on the context (or the domain). Google does not know in which context we are searching for a particular definition – it thus returns all definitions of a particular word or phrase it keeps in its database. “Mine”, for instance, can be defined either as “excavation in the earth from which ores and minerals are extracted” or as “explosive device that explodes on contact”. It is important to somehow detect the documents that do not talk about the geospatial domain and exclude them from the annotation process.

Note that we can also populate concepts with Google search results (in contrast or even in addition to populating it with definitions). In this case we can put the search term into a context by extending it with words or phrases describing the context. For example: to populate concept OpenPitMine in the context of “extracting materials” with documents, we would query Google for “open pit mine extracting materials” and consider for instance the first 50 search results. Centroid-to-centroid similarities for this approach are also shown in Table 1.

3.3 Using Linguistic Patterns for Hypothesis Checking

We can use a Web search engine to estimate the truthfulness of a hypothesis given as a statement in a natural language. If we query Google for “quarry is an open pit mine”, it returns 13 hits (at the time of writing this paper). We also get 3 hits for the query “mine is a quarry”. In contrast, we do not get any hits for queries “quarry is a transportation” or vice versa, and “quarry is a legislation” or vice versa. We can check for synonymy between any two words (or even n -grams) w_1 and w_2 with this same pattern expressed as a template: “ w_1 is a w_2 ” or “ w_2 is a w_1 ”. Hearst [7] introduced several such patterns for the acquisition of hyponyms. These patterns are thus called Hearst patterns.

Intuitively it seems that synonymy is the relation that is most suitable for the annotation task because we can infer similarity between two concepts from the “truthfulness of synonymy” (expressed for instance as the number of Google search results when “checking” the synonymy hypotheses) between these two concepts. However, hyponymy can be used to extend the set of synonymy hypotheses. The idea is to actually populate concepts with instances (in the true ontological sense) and then try to find synonymies between these instances. This is particularly useful in cases when the hypotheses checking on concepts (their string representations, more accurately) fails or yields inconsistent results. The system called KnowItAll [5] uses Hearst patterns and a set of Web search engines to populate concepts with instances.

3.4 Google Distance

Word similarity or word association can be determined out of frequencies of word (co-)occurrences in text corpora. Google Distance [2] uses Google to obtain these frequencies. Based on two requirements, namely (1) if the probability of word w_1 co-occurring with word w_2 is high then the two words are “near” to each other and vice versa, and (2) if any of the two words is not very common in the corpus, the distance is made smaller, the authors came up with the equation given in Eq. 2. They call it Normalized Google Distance (NGD).

$$\text{NGD}(w_1, w_2) = \frac{\max\{\log f(w_1), \log f(w_2)\} - \log f(w_1, w_2)}{\log M - \min\{\log f(w_1), \log f(w_2)\}} . \quad (2)$$

In Eq. 2, $f(w)$ is the number of search results returned by Google when searching for w (similarly $f(w_1, w_2)$ is the number of search results returned by Google when searching for pages containing both terms), and M is the maximum number of pages that can potentially be retrieved (posing no constraints on language, domain, file type, and other search parameters, Google can potentially retrieve around 10 billion pages).

It is also possible to put NGD computation into a context. This can be done simply by extending Google queries (the ones that are used to obtain frequencies) with words that form the context. Note that in this case M must be determined as the number of returned search results when searching for the words that form the context. The performance of NGD on our toy example is evident from Table 1.

We believe that NGD is not really the best way to search for synonymy because synonyms generally do not co-occur. It is more a measure of relatedness or association – nevertheless it can be tried out for the SWING annotation task. Also note that any other search engine that reports the total number of search results can be used instead of Google.

4 A Preliminary Experiment

We tested some of the presented methods on a dataset from the domain of minerals. We obtained 150 mineral names together with their synonyms. To list just a few: acmite is a synonym for aegirite, diopside is a synonym for alalite, orthite is a synonym for allanite, and so on (this dataset can be found at <http://www.csudh.edu/oliver/chemdata/minsyn.htm>). The mineral names were perceived as our domain ontology concepts while the synonyms were perceived as the feature-type ontology concepts. For each of the synonyms, the selected algorithms were used to sort the mineral names according to the strength of the association with the synonym in question. We measured the percentage of cases in which the correct mineral name was in the top 1, 3, 5, and 10 names in the sorted list. In other words, we measured the precision of each of the algorithms according to the top 1, 3, 5, and 10 suggested mineral names.

We employed 16 algorithms altogether: 7 variants of k -NN, 5 variants of the centroid classifier, and 4 variants of NGD. We varied the context and the data source (either Google definitions or Google search engine). We also varied whether the order of words in a term matters or not (if the order was set to matter then the term was passed to the search engine in quotes). Top 30 search results were considered when querying the search engine. Only English definitions and English search results were considered. In the case of k -NN, k was set dynamically by taking all the documents within the cosine similarity range of less than (or equal to) 0.06 into account. The final outcome of a k -NN algorithm was computed as a sum of all the cosine similarities measured between a synonym document and a mineral name document from the corresponding neighborhood. Table 2 summarizes the results of the experiment. The best performing algorithm is emphasized in the table.

5 Conclusions

This paper presents several techniques for automatic annotation in which “external” data sources (such as the Web) are used to compensate for the missing textual documents corresponding to concepts.

According to the preliminary experiment presented in Section 4, the presented techniques have a very good potential. From the results it is evident that – at least for the dataset used in the experiment – it is not beneficial to limit the search to Wikipedia (a free Web encyclopedia available at <http://www.wikipedia.org>) or Google definitions. However, it proved useful to perform the search in the context of “minerals”. Also important to notice is that k -NN outperforms the centroid classifier

when not put into a context. However, when the search is performed in the context of “minerals”, the centroid classifier outperforms k -NN. This occurs because the documents, gathered by the contextualized search, are less heterogeneous. Consequently the centroid is able to summarize the topic (which has explicitly to do with “minerals”) easier than if the documents were taken from the general context. Even more, the centroid cuts off the outliers (i.e. the noise) by averaging vectors’ components. On the other hand, when dealing with more heterogeneous set of documents from the general context, the centroid is “shifted” towards irrelevant (sub)topics (i.e. other than “minerals”) which results in poorer performance.

Table 2. The results of the preliminary experiment.

Algorithm	Data source	Context	Quotes	Precision [%]			
				Top 1	Top 3	Top 5	Top 10
k -NN	Google srch.	general	no	82.67	90	92	93.33
Centroid	Google srch.	general	no	78	91.33	92	94
k -NN	Google def.	general	no	70	76	77.33	79.33
Centroid	Google def.	general	no	76	77.33	78.67	79.33
k -NN	Google srch.	“site: wikipedia.org”	no	43.33	60.67	70	76
k -NN	Google srch.	“minerals”	no	86.67	97.33	98.67	100
Centroid	Google srch.	“minerals”	no	91.33	96.67	98.67	99.33
NGD	Google srch.	general	no	8.67	18	21.33	30.67
NGD	Google srch.	“minerals”	no	12.67	21.33	29.33	42.67
k -NN	Google srch.	general	yes	80.67	89.33	91.33	93.33
Centroid	Google srch.	general	yes	78	91.33	93.33	94.67
k -NN	Google srch.	“site: wikipedia.org”	yes	27.33	38.67	39.33	42
k -NN	Google srch.	“minerals”	yes	88	98	99.33	100
Centroid	Google srch.	“minerals”	yes	93.33	98.67	99.33	100
NGD	Google srch.	general	yes	16	26	36.67	54.67
NGD	Google srch.	“minerals”	yes	11.33	22.67	36.67	58

These conclusions lead us to an idea of how to improve the presented algorithms. We believe that it would be beneficial to first cluster the documents and then perform the classification on each cluster separately. The final classification result would then be obtained by maximizing or averaging the per-cluster classification results.

Also noticeable from Table 2, as we suspected, NGD is not very successful in detecting synonymy. Furthermore, it is much slower than the other presented algorithms as it is querying the Web to determine term co-occurrences. Last but not least, we can see that the best performing algorithms perform slightly better if the

search query is put into quotes (i.e. if the order of words in the term that represents the query is set to matter).

The focus of our future work will be on the implementation of the SVM-based term matching algorithm and the implementation of the clustering of documents. We have high hopes for these two technologies. We also lack the implementation of the hypothesis checking; however, we believe that using linguistic patterns for string matching will be inferior to the text mining approaches. The next step will be to go beyond string matching and also consider the neighborhood of a concept similarly to [12] (provided that the concept is a part of an ontology or a similar structure).

The implemented technologies will soon become a part of OntoBridge, an open-source system for semi-automatic data-driven ontology annotation (i.e. for mapping or mediation). OntoBridge will be one of the deliverables of the European project SWING.

Acknowledgments. This work was partially supported by the IST Programme of the European Community under SWING – Semantic Web Services Interoperability for Geospatial Decision Making (FP6-026514) and PASCAL Network of Excellence (IST-2002-506778).

References

1. Cardoso-Cachopo, A., Oliveira L., A.: Empirical Evaluation of Centroid-based Models for Single-label Text Categorization. INSEC-ID Technical Report 7/2006 (2006)
2. Cilibrasi, R., Vitanyi, P.: Automatic Meaning Discovery Using Google (2004)
3. de Bruijn, J.: The Web Service Modeling Language WSML (2005)
4. Duan, K.-B., Keerthi S., S.: Which is the Best Multiclass SVM Method? An Empirical Study. In LNCS vol. 3541/2005, Springer Berlin Heidelberg (2005)
5. Etzioni, O., Cafarella, M., Downey, D., et al.: Web-scale Information Extraction in KnowItAll (Preliminary Results). In Proceedings of WWW2004, New York, USA (2004)
6. Grear, M., Klien, E., Fitzner, D., I., Maué, P., Mladenic, D., Grobelnik, M.: D4.1: Representational Language for Web-service Annotation Models. Project Report FP6-026514 SWING, WP 4, D4.1 (2006)
7. Hearst, A., M.: Automatic Acquisition of Hyponyms from Large Text Corpora. In Proceedings of COLING-92 (1992)
8. Mitchell, T. M.: Machine Learning. The McGraw-Hill Companies, Inc. (1997)
9. Open Geospatial Consortium: Web Feature Service Implementation Specification, Version 1.0.0 (OGC Implementation Specification 02-058) (2002)
10. Thorsten, J.: Training Linear SVMs in Linear Time. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 217–226, PA, USA (2006)
11. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
12. Gligorov, R., Aleksovski, Z., ten Warner, K., van Harmelen, F.: Using Google Distance to Weight Approximate Ontology Matches. In Proceedings of WWW 2007, Banff, Alberta, Canada (2007)