Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

DISI - Via Sommarive 14 - 38123 Povo - Trento (Italy)
http://www.disi.unitn.it

UNDERSTANDING NATURAL LANGUAGE METADATA

Aliaksandr Autayeu, Fausto Giunchiglia, Pierre Andrews

March 2010

Technical Report # DISI-10-026

Understanding Natural Language Metadata

Aliaksandr Autayeu, Fausto Giunchiglia, Pierre Andrews

Abstract

Handling everyday tasks such as search, classification and integration is becoming increasingly difficult and sometimes even impossible due to the increasing streams of data available. To overcome such an information overload we need more accurate information processing tools capable of handling big amounts of data. In particular, handling *metadata* can give us leverage over the data and enable structured processing of data, however, while some of this metadata is in a computer readable format, some of it is manually created in ambiguous natural language. Thus, accessing the semantics of *natural language* can increase the quality of information processing. We propose a *natural language metadata* understanding architecture that enables applications such as semantic matching, classification and search based on natural language metadata by providing a translation into a formal language which outperforms the state of the art by 15%.

1 Introduction

Information overload is what every modern information worker complains about. The volumes of information and demands, let us put aside expectations of the information workers, grow faster than the tools evolve. But we do not want less information, instead, we want better management tools, which will alleviate or solve the problem of information overload. Yet there is an instrument, known for thousand of years and extensively used in libraries to get a leverage over massive amounts of books. Traditionally, we did not search books themselves, we searched a library catalogue, full of data about books. Data about data or *metadata* accompany every significant piece of information.

Increasing amounts of metadata are being generated automatically and most of it is well processed by filter search. Problems begin when we enter the realm of natural language (NL): we carefully compose titles for our papers, books and blog posts; many of us are encouraged to write meaningful subject lines of emails we send; we tag photos, posts and videos in social networks; we create folder structures in email client or in personal file systems, carefully authoring our own small classifications, kind of mini- or lightweight ontologies – we manually generate all kinds of *Natural Language Metadata* (NLM).

Then we spend time sorting emails into those folders and wishing the files we receive would sort themselves out in the appropriate places in our home folder. We sift through a business catalogue, searching for relevant categories

and having received another catalogue, wish to have it aligned automatically with a freshly filtered one. Many of these tasks have been solved and use algorithms which operate on lightweight ontologies [16], such as the "get specific" algorithm [17] for classification of documents in hierarchies or S-Match [15] and minimal S-Match [13] for matching of ontologies. See Section 2 for details on these applications. The core of many of these algorithms uses a formal language (FL) that enables reasoning about the data being processed. However, semantic applications face a well-known chicken and egg problem [18]: for these applications to yield meaningful results, the data they work on, should be represented in a formal language or have semantic annotations to enable automatic reasoning. And there is little of both applications and data.

Expecting the users to write in formal language is unrealistic and while for automatically generated metadata the semantics is predetermined by the coding standard, the semantics of metadata written in natural language remains hidden. Uncovering the semantics of natural language metadata and translating it into a formal language will enable such tools to reason about it and thus give us the ability to leverage semantic services on our data. Modern natural language processing (NLP) tools have evolved over the domain of newswire or similar text. However, the language used in natural language metadata differs from the one used in normal texts, such as news stories and books.

We briefly overview the problems one faces in the task of translating natural language metadata into a formal language and present a solution for processing natural language metadata. We present a modular pipeline architecture which enables automatic semantics generation for natural language metadata. We use this architecture to translate natural language labels, such as categories of web or business directory into their formal counterpart. It can be used in a fully automated manner as well as incorporating the user's feedback. Although initially intended to process a subset of the language, namely natural language metadata, the presented solution can also be applied to a generic text, translating essential parts of it into a formal language. Our solution can be used as a preprocessing step for such applications as semantic classification, matching and search. We provide a synthetic evaluation of the solution on two datasets, representing a sample of natural language metadata.

The paper is structured as follows. In Section 2 we describe the model applications that motivated our research. Section 3 overviews the related work and compares our work with it. Section 4 describes the datasets that represent the domain of natural language metadata. Section 5 walks through an example of translation to a formal language, highlighting the problems on the way. Section 6 describes in details the proposed solution. Section 7 provides a synthetic evaluation of our solution. Section 8 concludes the article.

2 Motivation

Our study of natural language metadata is motivated by the applications which use a formal counterpart of a natural language metadata. These applications can

benefit from an improved understanding of natural language metadata. Many of them need the same steps of processing:

- recognizing atomic concepts in language metadata by mapping natural language tokens into senses from a controlled vocabulary,
- disambiguating the senses drawn from the controlled vocabulary and
- building complex concepts out of the atomic ones.

We describe three representatives of these applications below before discussing our NL translation technique that enables such semantic services in the following sections.

Semantic Matching. Semantic matching could be seen as an operator that takes two tree-like structures (such as classifications or schemas) and produces correspondences between those tree nodes that correspond semantically to each other. Semantic matching employs two key ideas: a) it computes semantic relations like *equivalence* and *more general*; and b) it computes them by analyzing meaning (concepts) encoded in the labels of the input trees [15].

However, the formal representation of the concepts of each tree node label, on which the algorithm operates to compute the correspondences, needs to be created first. Most often the tree node labels are written in a natural language, and, therefore, as a first step towards reasoning, the algorithm needs to translate a natural language into its formal counterpart, in this case propositional description logic.

Semantic Classification. Hierarchical classifications represent a natural way of organizing knowledge. However, keeping them up to date requires putting new information items (e.g. documents) into the appropriate places in the hierarchy. This problem is addressed by the "get-specific" algorithm [17]. This algorithm follows a knowledge-centric approach and first converts a natural language classification into a formal classification, where the labels are expressed in a concept language. A concept for a classified document is built from the document's keywords translated into concepts and joined with conjunctions. Then the algorithm reasons over these concepts.

Here again, this semantic service requires a formal representation of the concepts of each classification node to be effective and we thus need to translate the natural language to such a formal representation.

Semantic Search. Search is a key application for information workers. One of the proposals to improve search is to go from a syntactic search, which operates on arbitrary sequences of characters and computes string similarity, to a semantic search, which operates on concepts and computes semantic relatedness [12]. However, the documents which are searched and the search terms are written in a natural language, where concepts need to be identified first to enable semantic search.

Once again we encounter the need to go from natural language to its formal counterpart for the search terms and the document concepts. This is another application which is appropriate for the techniques we propose.

3 Related Work

Many algorithms are based on reasoning in a formal language. However, users are accustomed to a natural language and it is difficult for them to use a formal one. A number of approaches has been proposed to bridge the gap between formal and natural languages.

Controlled languages, such as Attempto [10], have been proposed as an interface between natural language and first-order logic. This, as well as a number of other proposals based on a controlled language approach [27, 26, 7], require users to learn the rules and the semantics of a subset of English. Moreover, users need to have some basic understanding of the first order logic to provide a meaningful input. The difficulty of writing in a controlled language can be illustrated by the existence of editors, such as ECOLE [25], aiding the user in the editing of the controlled language.

As an interface for ontology authoring a number of controlled natural languages have been proposed in [7, 2, 5]. The approach of [2] uses a small static grammar, dynamically extended with the elements of the ontology being edited or queried. Constraining the user even more, the approach of [5] enforces a one-to-one correspondence between the controlled language and the ontology language. The authors in [7], following a practical experience, tailored their controlled language to the specific constructs and the errors of their users. Some of these and other controlled languages have been critiqued [21] due to their domain and genre limitations.

For querying purposes, [28] proposes a natural language interface to the ontologies by translating natural language into SPARQL queries against a selected ontology. This approach is limited by the extent of the ontology with which the user interacts.

Another way to bridge the gap between formal and natural languages has been proposed in [11], where the authors propose to *manually* annotate web pages, rightfully admitting that their proposal introduces a "chicken and egg" problem.

The approach of [19] of automatically translating hierarchical classifications into OWL ontologies is more interesting. However, by considering the domain of products and services on the examples of eCl@ss and UNSPSC, some simplifying domain-specific assumptions are made, which hold in this domain, but which do not hold in a general case.

Differently from the mentioned above approaches, our work does not impose the requirement of having an ontology, the user is not required to learn a syntax of a controlled language, and we do not restrict our consideration to a specific domain. This article develops the theme of [29], improving it in several ways, such as extending the analysis to a wider sample of metadata, using a classification of named entities and introducing a lightweight parser.

Table 1: Dataset characteristics

Dataset	Labels	Sample	Unique labels, %	Label length, tokens max	Label length, tokens avg
LCSH	335 704	44 490	100.00	24	4.0
NALT	43 038	13 624	100.00	8	1.6
DMoz	494 043	27 975	40.48	12	1.8
Yahoo	829 081	132 350	16.70	18	2.0
eCl@ss	14 431	3 591	94.51	31	4.2
UNSPSC	19 779	5 154	100.00	19	3.5

4 Sets of Natural Language Metadata

Many types of metadata are available in the world and on the web. Some is generated automatically, for example the information attached to photos by cameras, and this metadata has a well defined, machine readable meaning. On the contrary, some metadata contain natural language created manually, such as article's titles, keywords or business catalogue's category names, and their meaning is not formalized and one has to extract it to enable automatic processing powered by reasoning over the meaning.

Natural language processing is a well established field and contains many developed and mature techniques. However, many of these techniques suffer a performance degradation when applied to a different domain [3].

To study the domain of natural language metadata, we have analysed the following datasets: DMoz, eCl@ss, LCSH, NALT, Topia, Iconclass, UNSPSC, Yahoo! Directory. These datasets belong to the domain of natural language metadata and illustrate different uses of natural language metadata, for example for classification and for indexing. They include web directory category names, business catalogue category names, thesauri and subject headings. Table 1 summarizes some key characteristics of these datasets, and in the following we provide a more detailed description.

DMoz or Open Directory Project[1] is a well known web directory, collectively edited and maintained by a global community of volunteer editors. It is one the largest web catalogues and it powers directory services for many sites[2], including popular search engines, such as Google.

eCl@ss[3] is an "international standard for the classification and description of products and services". One of the project's goals is to improve the collaboration between enterprises. It is edited by professional editors.

Iconclass[4] is "a classification system designed for art and iconography". It covers the art domain and is widely used by museums and art institutions to classify items.

[1] `http://dmoz.org`
[2] 114 sites according to DMoz
[3] `http://www.eclass-online.com/`
[4] `http://www.iconclass.nl/`

LCSH[5] stands for "Library of Congress Subject Headings". It is a thesaurus of subject headings maintained by the U.S. Library of Congress for use in bibliographic records. LCSH is edited and used by librarians and library users for classification of library items to enable and facilitate uniform access and retrieval in many of the world libraries.

NALT[6] stands for "National Agricultural Library Thesaurus". NALT is a hierarchical vocabulary of agricultural and biological terms used extensively to aid indexing and retrieval of information within and outside of U.S. Department of Agriculture.

Topia is a hierarchical thesauri which covers the art domain.

UNSPSC[7] stands for "United Nations Standard Products and Services Code". It is a "globally used classification hierarchy for products and services owned by the United Nations Development Programme (UNDP) and managed by GS1 US". Edited by professional editors and being a classification system, it enables accurate classification of products and services for companies.

Yahoo! Directory[8] is a "catalog of sites created by Yahoo! editors who visit and evaluate websites and then organize them into subject-based categories and subcategories".

5 Translating Metadata: An Example

Natural language metadata, being a subset of natural language, is ambiguous and hard to reason about. These problems need to be addressed to enable metadata use in semantic applications such as the ones described in Section 2. One of the approaches to this problem, described in [16], is to translate the NL metadata into a propositional Description Logic language L^C to reason about the set of items (e.g. documents) semantically described by the formula.

If we consider the example from [29]: "Bank and personal details of George Bush", we can identify several key steps of the translation process and highlight some processing problems. We refer interested readers to [16] for a more detailed theoretical account of lightweight ontologies and their translation.

We consider *atomic concepts* as the basic building blocks of the L^C formulas. Any controlled vocabulary containing word senses can provide such atomic concepts (for example, we use WordNet [8]). Atomic concepts can be roughly divided into two large groups: common nouns and adjectives, and proper nouns, also known as named entities.

First, we identify in the label potential atomic concepts. In our case, the following atomic concepts can be identified in the label: n#1 ("bank"), a#2 ("personal"), n#3 ("detail"), n#4 ("George Bush"), where the concepts are assigned unique IDs mapped to unambiguous senses in our controlled vocabulary.

[5] `http://www.loc.gov/cds/lcsh.html`
[6] `http://agclass.nal.usda.gov/`
[7] `http://www.unspsc.org/`
[8] `http://dir.yahoo.com/`

Second, we build complex concepts out of the atomic concepts and logical connectives of L^C. We derive logical connectives out of syntactic relations between words. For example, we translate prepositions like "of" into logical conjunction between sets ($\sqcap$) and coordinating conjunctions like "and" and "or" into logical disjunctions between sets of items ($\sqcup$).

Finally, we build the structure of the formula taking into account how the words are coordinated in the label. In our example, we put a conjunction between "detail" and "George Bush".

As a result we have the L^C formula, representing the concept, unambiguously describing the set of documents about this concept. In our example, the final formula is $(n\#1 \sqcup a\#2) \sqcap n\#3 \sqcap n\#4$.

The translation process contains several steps, where we can make mistakes due to incorrect processing of natural language. For example, the word "personal", if recognized by the POS tagger as a noun instead of an adjective, might be mapped to the wrong sense in the controlled vocabulary. The tokens "George" and "Bush" should be recognized as a single concept, namely a proper noun, and pointed to the appropriate person, disambiguating between George H. W. Bush and George W. Bush.

6 Metadata Processing Pipeline

Many of the problems arising during the translation of natural language metadata into formal language are well known NLP problems, therefore we follow a NLP pipeline design shown in Figure 1.

We introduce some optional dialog boxes that allow the pipeline to be used in two modes: fully automated and user-assisted. The latter is introduced as a solution to sub-par performance of some difficult processing steps, such as word sense disambiguation (see section 6.7). It allows the user to introduce corrections into the decisions made by the pipeline.

We follow with the description of the pipeline modules. Each module addresses a specific problem, which we describe together with a proposed solution.

For the modules in the early stages of processing, such as tokenization, POS tagging and Named Entity Recognition, the state of the art NLP solution is to use a supervised learning algorithm. Due to their maturity, there is little difference in the performance of the state of the art algorithms and we therefore use the state of the art maximum entropy based probabilistic algorithm provided by the OpenNLP[9] tools. However, as discussed in the following paragraphs, we train this algorithm on the particular Natural Language found in our metadata domain because the standard, full text models provided by the state of the art are not well suited to the task.

[9] `http://opennlp.sourceforge.net/`

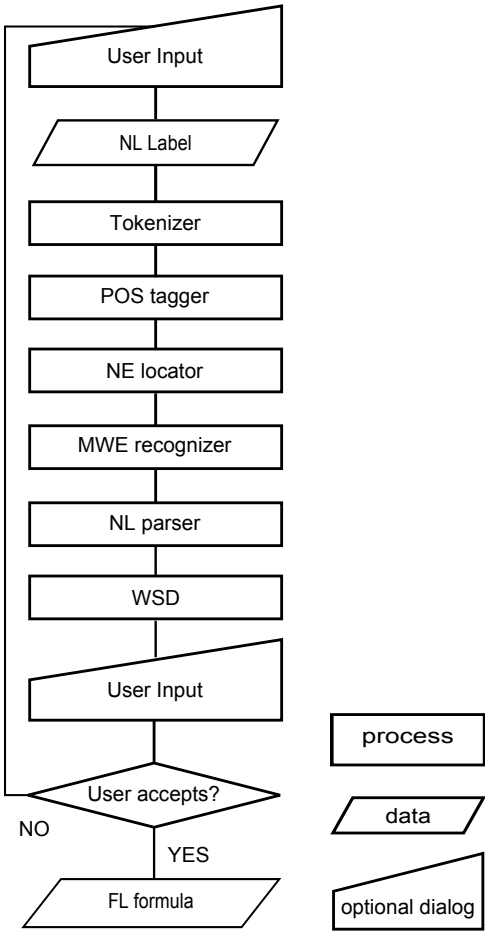

Figure 1: Metadata Processing Pipeline

6.1 Tokenization

6.1.1 Problems

Tokenization is the first step in almost any language processing. Although a relatively simple task, in our domain of natural language metadata the standard tools encounter several difficulties. These difficulties arise from various non-standard use of such punctuation elements as commas, round and square brackets, slashes, dashes, dots, ellipsis and semicolons: , () [] \ / : ... -. In addition, in several datasets we have noticed a non-standard use of punctuation, such as missing conventional space after a comma.

Consider the following example "Hand tools (maint.,service)" from eCl@ss. This label uses a dot for an abbreviation and is followed immediately by a comma with a missing conventional space afterward, all of which is within round brackets. Such combinations are rare in normal texts and therefore the performance of standard tools, trained on such texts, degrades.

6.1.2 Solution and Evaluation

We performed a 10-fold cross-validation on each of our annotated datasets with the OpenNLP "standard" model, and also tested a *combined model* trained on the merged datasets. Table 2 summarizes the results of the experiments.

We report the results using precision per label (PPL) measure for major datasets. Namely, we count the percentage of correctly tokenized labels. In columns we report the performance of different tokenizer models on a particular dataset. In rows we report the performance of a model trained on a particular dataset, on the other datasets. Figures on the diagonal and for the combined model are obtained by a 10-fold cross validation.

The next to last row reports the performance of the OpenNLP standard model. The last row is a combined model trained on the combination of the datasets available. Although in many cases the performance improvement is marginal, there are noticeable improvements in the cases of eCl@ss, Icon and LCSH. One can also notice that the model trained only on this particularly difficult datasets also outperforms the standard OpenNLP model.

The analysis of errors made by a tokenizer unveils that the main reason of this performance improvement is that punctuation is used in some short labels more intensively than in normal text. Therefore a retrained model grasps this difference better than the standard one.

6.2 POS Tagging

6.2.1 Problems

Most of the state of the art POS tagging algorithms are based on supervised learning approaches. To determine a part of speech for a particular word, a tagger extracts a feature set out of it and a classifier estimates the probabilities for all tags from a tag set to be the correct tag for this particular word. Most

Table 2: Tokenizer performance, Precision Per Label, %

Model	DMoz	eCl@ss	LCSH	NALT	UNSPSC	Yahoo
DMoz	**99.95**	55.22	78.11	98.97	100.00	98.67
eCl@ss	99.73	**94.29**	97.70	99.97	99.98	99.45
LCSH	99.93	87.41	**99.79**	99.87	100.00	99.85
NALT	98.82	69.17	85.55	**100.00**	100.00	98.48
UNSPSC	97.09	47.98	43.63	98.80	**100.00**	96.76
Yahoo	99.90	55.69	88.47	99.12	100.00	**99.90**
OpenNLP	*99.86*	*79.39*	*95.57*	*99.96*	*100.00*	*99.77*
combined	**99.95**	**94.26**	**99.51**	**100.00**	**100.00**	**99.90**

popular features include prefixes and suffixes (morphology) of the word and its neighbours (context).

In the domain of natural language metadata the traditional POS taggers are challenged by a shorter context. There are fewer neighbour tokens available, if they are available at all, as in many cases the average label length is under 2 tokens.

In addition, the prefixes of words from normal texts differ from the ones generated for the words of metadata phrases, as often the capitalization convention is different. For example, in thesauri the capitalization rule is often mixed between higher and lower levels of terms hierarchy. Compare, for example, the top level label "Biological Sciences" to the bottom level label "freshwater fish" taken from the NALT dataset. On the contrary, in web directories, the capitalization rule is stable across levels, but different from the normal text. Consider a typical label taken from the Yahoo dataset: "Classical Chinese Art", where all the words are capitalized.

Moreover, the POS tag distribution for the short phrases is completely different from the one of the normal text, as for example, verbs are almost absent: on average, there are 3.5 verbs (VB) in a whole dataset, ranging from 0.0001% to 0.15% of all the tokens of the dataset.

6.2.2 Solution and Evaluation

Similarly to the tokenizer, the POS tagging algorithms are mature and state of the art algorithms have similar performance. Therefore we have chosen the state of the art POS tagger from OpenNLP tools trained on the combined datasets. It is based on Conditional Maximum Entropy Model [1, 22].

We performed experiments with the standard OpenNLP models, with a 10-fold cross-validation on each of the datasets, and tested some combined models. We report the results for the major datasets in a similar way to Table 2, using a precision per token (PPT) measure in Table 3 and a precision per label (PPL) measure in Table 4. Namely, we count the percentage of correctly tagged tokens (PPT) and correctly tagged labels (PPL). The "OpenNLP" row reports the performance of OpenNLP standard model. The "path-cv" row reports the 10-fold

Table 3: POS tagger performance, Precision Per Token, %

Model	DMoz	eCl@ss	LCSH	NALT	UNSPSC	Yahoo
DMoz	**95.15**	14.30	45.28	75.46	58.57	92.00
eCl@ss	56.67	**97.69**	63.48	34.08	89.49	71.05
LCSH	86.39	77.81	**96.89**	84.24	85.35	91.17
NALT	49.15	66.15	65.23	**97.27**	47.88	44.04
UNSPSC	61.76	65.21	42.20	34.75	**97.59**	77.00
Yahoo	92.69	36.83	57.23	76.84	54.86	**98.15**
OpenNLP	*64.48*	*66.28*	*72.76*	*58.12*	*74.94*	*61.67*
all-except	93.74	82.98	73.30	86.72	89.38	95.45
path-cv	**99.67**	99.65	99.45	**99.77**	99.63	**99.84**
combined	99.32	**99.93**	**99.74**	99.76	**99.82**	99.70

Table 4: POS tagger performance, Precision Per Label, %

Model	DMoz	eCl@ss	LCSH	NALT	UNSPSC	Yahoo
DMoz	**93.98**	14.12	27.54	75.37	49.69	91.87
eCl@ss	48.80	**91.28**	28.60	28.73	69.65	62.11
LCSH	81.98	48.79	**91.38**	81.91	68.14	88.16
NALT	46.97	23.61	28.82	**96.42**	13.21	34.05
UNSPSC	57.07	45.08	22.76	31.03	**92.39**	75.46
Yahoo	89.54	15.20	34.84	75.04	45.91	**97.91**
OpenNLP	*49.89*	*19.02*	*27.26*	*40.55*	*33.20*	*47.44*
all-except	91.59	58.40	53.25	84.77	76.19	94.77
path-cv	96.64	93.34	92.64	96.29	92.72	98.35
combined	**99.10**	**99.69**	**99.24**	**99.74**	**99.40**	**99.68**

cross-validation precision figures for the case where the context was extended to include labels in the preceding levels of the classification hierarchy. The last row is a combined model trained on the combination of all datasets available.

The "all-except" row is of particular interest, because it reports the performance of the model trained on all available datasets, except the one it will be tested on. For example, the model to be tested on DMoz data will include all datasets as training data, except DMoz itself. We can already notice a performance improvements compared to the standard OpenNLP model. The performance improvements are in a 15-30% range, with the only exception being LCSH case.

We believe that the differences in the POS tag distribution between normal text and natural language metadata is the main reason of these improvements. Short labels mostly describe (sets of) objects and they do it by using proper and, often modified by adjectives, common nouns, more frequently than in normal text, where verbs constitute a larger portion of words.

Looking at Table 4, we can notice that the data confirms the trend reported by Table 3 with even more drastic performance improvements ranging in *all*

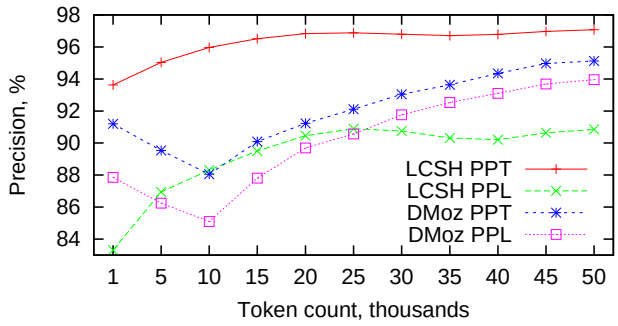

Figure 2: Incremental Training for LCSH and DMoz

cases from 26% to almost 50%.

The "path-cv" rows show the importance of an extended context, where we included labels from the preceding (higher) levels of the hierarchy, which are sometimes available. Comparing the figures in bold with the figures in the "path-cv" row, we can notice an increase in performance reaching 4.5% (PPT) and 2.6% (PPL) with the averages of 2.5% (PPT) and 1.2% (PPL).

We performed incremental training to explore the stability of the models obtained. For clarity, we report the first 50 000 tokens of two major datasets in Figure 2, with a notice that the rest shows similar trend. Namely, the performance tends to stabilize and reach a plateau. In few cases, like DMoz, it fluctuates in the beginning before stabilizing. We found a count of tokens needed for the model to reach a plateau to be larger than reported in [29].

We analyzed the errors made by the POS taggers by checking the confusion matrix and some misclassified word examples. Misclassifications can be divided into two major classes. In datasets rich in named entities, the most frequent misclassifications are between nouns and proper nouns, such as NN (nouns) misclassified as NNP (proper nouns) and vice versa. They range from 46% to 55% of the errors. In other datasets the most frequent misclassifications occur between nouns and adjectives, such as NN (nouns) misclassified as JJ (adjectives) and vice versa. They range from 40% to 97% of the errors. This leads us to the conclusion that named entities need a particular attention in the form of a recognition module, which is necessary and can improve overall processing performance (see Section 6.3).

POS tags provide some fundamental information about the language and they are used extensively in many NLP tasks either as source information, or as a feature. This is why we paid particular attention to the POS tagger performance, as POS tag information shows that natural language metadata really constitutes a separate domain of the language.

6.3 Named Entity Recognition

6.3.1 Problems

Named entities (NEs) pose several problems for the task of natural language metadata understanding:

- we need to identify them;
- we need to classify them, because different classes of NEs require different processing;
- we need to disambiguate them.

As for tokenization and POS tagging, NEs in natural language metadata behave differently than the NEs in normal text. The first type of issue is the non-standard joining of NEs, such as in the label "NS Wales Queensland" where there is no separation of any kind between the two geographical NEs. The second type of issue is that, in some datasets, the entities such as personal names and locations are written in a "backward" rather than "forward" manner, as in the label "van Ruisdael, Jacob". The third type of issue is that, to make the label shorter, NEs are sometimes joined together, as in the label "Green, Henry and Charles". Note that these examples are also ambiguous to human readers.

Table 5 reveals differences of "quantitative" nature. Namely, NEs are frequently used in some kinds of natural language metadata while not so frequently in others. One can note that in datasets with NEs they tend to span over a large portion of labels: 18% to 37%. In one group of datasets (DMoz, NALT, Yahoo) a whole label is frequently, but not always, a single named entity. Such labels constitute from 90% to 95% of all labels with NEs. In another group of datasets (LCSH and Iconclass), NEs are predominantly part of a label which contains other tokens as well. In addition, in datasets where NEs are present in sufficient quantities, they frequently, but not always, tend to span the whole label. Also, the distribution of the entities across levels of the hierarchy is not uniform, they tend to cluster in the middle levels of the hierarchy and below some specific labels, such as letter bars like "A" and facet specifiers like "By Country".

The analysis of our extended samples of metadata allowed us to see that the assumption made in [29] about labels being either named entity (NE) or not, does not always hold. Even in the cases of DMoz, NALT and Yahoo the mixed labels constitute from 4.11% to 9.62%, with an average of almost 7%, while in LCSH and Iconclass cases mixed labels reach 98.01% and 97.67%, respectively. For example, the LCSH dataset contains labels combining geographical named entities with named events (or person names) or named entities with disambiguation within a single label, such as illustrated by the label "Maat (Egyptian deity)".

Table 5: Named entities characteristics

Dataset	% of labels with		NEs	of them, %		
	NEs	NEs only	count	LOC	ORG	PERS
DMoz	36.29	34.80	10 244	75.82	9.21	7.34
eCl@ss	0.39	0.00	14	0.00	0.00	100.00
LCSH	37.55	0.75	24 836	79.51	12.46	2.15
NALT	1.75	1.59	242	92.15	5.37	0.41
Topia	0.00	0.00	0	0.00	0.00	0.00
Iconclass	4.30	0.10	43	93.02	0.00	6.98
UNSPSC	0.06	0.00	3	0.00	33.33	66.67
Yahoo	17.97	16.68	24 668	67.54	15.56	8.27

6.3.2 Solution and Evaluation

Our solution is to adopt a Named Entity Recognizer algorithm, based on OpenNLP NE algorithm and train it on our datasets to improve the understanding of the NEs in the labels.

The state of the art Named Entity Recognition (NER) algorithms are mature and the state of the art algorithms attain comparable performances. We have chosen the NER algorithm from OpenNLP and we also report the performance of the Stanford NER algorithm [9].

As our annotation and classification scheme we used the CONLL shared task [24] classes of NEs. We identify three major named entity types LOCation, ORGanization, PERSon and a fourth "catch-all" type MISCellaneous.

We performed a set of experiments, comparing the performance of the standard model supplied with a toolkit, with a custom model, trained and tested via 10-fold cross-validation on our annotated datasets. Table 6 reports the results of the experiments for the datasets containing large quantities of NEs. The "all-except" row has the same meaning as previously. The "std" row reports the performance of the OpenNLP model, trained with a standard feature set, without dictionaries. The standard feature set includes the following features:

- the token itself,
- the token lowercase flag,
- the flags indicating whether a token contains only 2 or 4 digits,
- the flags for presence of numbers, hyphens, backslashes, commas and periods in a token,
- the token capitalization pattern.

Additionally we performed a 10-fold cross-validation with a standard feature set of OpenNLP NER on a combination of all our datasets, reaching an F-Measure of 64.20%.

Although the figures represent a less uniform picture than in the previous tasks, one can note that the performance of the standard models is quite low

Table 6: NER performance, F-Measure, %

Model	DMoz	LCSH	NALT	Yahoo
OpenNLP	11.76	41.68	17.1	9.3
Stanford	22.37	31.15	2.26	15.96
std	32.19	**60.35**	0.57	**34.76**
all-except	**41.38**	3.83	**32.65**	33.72

and the custom models outperform them. For comparison, one state of the art approach for NER on normal text attains an F-Measure of 68.63% [6]. Only in the case of the LCSH dataset, the performance is close to the state of the art levels for the normal text. The large differences between the "std" and the "all-except" results for the LCSH dataset are explained by the fact that in LCSH NEs such as PERSON and LOCATION are frequently written in a "backward" fashion, separated by commas as illustrated above.

We conclude that the chosen approach is promising, as even in the absence of an important dictionary and context features we notice a performance improvement. This shows that some additional exploration is required to improve the feature set. For natural language metadata we identify three broad groups of features, depending on the available context:

- features available for a label only;
- features available for a label and the hierarchy of labels above it;
- features available when a complete dataset is available, such as tokens and labels frequencies, as used in [29].

We intend to explore and compare these feature sets to complete the investigation.

The last experiment with a combination of datasets shows the potential advantages of introducing the dictionary feature. Some of our datasets share covered domains (for example, the Web in the case of DMoz and Yahoo) and their set of named entities intersect. Thus, by combining the datasets the algorithm is able to learn more from them.

6.4 Multiword Expressions Recognition

6.4.1 Problems

Differently from the state of the art approaches in natural language processing, which are mostly probabilistic, translating natural language metadata into a formal language involves a fair amount of knowledge based processing. One of the main reasons is that atomic concepts, which are the basic building blocks in the target formal language, are taken from a dictionary, controlled vocabulary, or another knowledge base. These linguistic resources contain multiword

expressions (multiwords), such as "a cappella singing" and "red tape", reflecting the fact that natural language already has complex concepts. Recognizing multiwords allows exploiting their precise meaning assigned by a human expert.

In recognizing multiwords we face several problems [4], such as reduced syntactic and semantic transparency, recognizing fixed and non-modifiable expressions versus semi-fixed expressions, as well as other expression types [23]. In the context of our task the severity of some of these problems is somewhat alleviated by the fact that we are backed by a linguistic resource. This splits the problems into two categories.

First, identifying the potential multiwords which are not present in the current vocabulary. Solving this problem allows us to enrich our linguistic knowledge by offering the user an option to check and add potential multiwords into the controlled vocabulary if we are in the interactive processing mode, or marking such cases for later processing in unattended processing mode. Given the difficulty of the problem, we leave this task for future work.

Second, recognizing existing multiwords present in the used vocabulary. This might include:

- disentangling them in case several of them are present simultaneously in the phrase,
- taking into account "obstacles" such as conjunctions and plurals,
- as well as taking into account multiwords spread over more than one level of hierarchy.

For example, the phrase "a cappella and gospel singing" contains two multiwords: "a cappella singing" and "gospel singing".

6.4.2 Solution

We use simple heuristics that recognize multiwords in the phrase taking into account several of the most common problems such as:

- non contiguous multiword instances,
- coordinating conjunctions,
- plurals,
- possible multiplication of tokens.

For example, we recognize both multiwords from WordNet present in the phrase "a cappella and gospel singing". Namely, we recognize "a cappella singing" and "gospel singing" despite the first being split by "and" and requiring "singing" token multiplication from the second.

First, we analyze consecutive label tokens for the presence in the multiword list taken from WordNet, compiling a list of candidates. In our example, we mark all tokens except "and" as potential candidates for two multiwords. We

make two lists of token indexes: {1,2,5} and {4,5}, where each number refers to the respective token of the label, such as 1 for "a", 2 for "cappella" and so on. When we check the token to be an expression candidate token, we test for it to be a derived form and check its lemmatized root form, for example, removing plural if necessary.

Second, we test for a simple case of consecutive tokens forming an expression and mark the candidates. In our example this would mark the second candidate {4,5} "gospel singing" as a recognized expression.

Third, for non-adjacent candidates like {1,2,5} we check what separates the tokens. We allow only "and" and "or" conjunctions to separate the tokens of a potential candidate. Our first candidate satisfies this condition.

Fourth, we check that a candidate's non-adjacent tokens follow a basic noun phrase pattern of {adjectives... nouns}. In our example that allows us to mark {1,2,5} "a cappella singing" as a recognized expression.

Fifth, we check that in the case of coordinated tokens the label after recognition preserves coordination. For example, in the case of the label "gospels and singing" we would recognize {1,3} as a potential candidate. However, recognizing "gospel singing" here would lead to a break of coordination and to an ungrammatical label: "gospel_singing and".

Last, we conclude the recognition in the label by multiplying the tokens if necessary. Our example transforms into "a_cappella_singing and gospel_singing", where we use underscores to show recognized multiword expression.

In addition, we repeat these heuristics when including tokens from the label from higher (upper) levels of the hierarchy. For example, in the case of a hierarchy "Music/Gospels/Singing" we would check "music singing" and "gospel singing" to be a candidate multiwords. Conversely from a single label case, we do not change the label. We only enrich the list of senses of the tokens of the label in question with the senses of a multiword. In this example we would add the sense(s) of the "gospel singing" multiword to the "signing" token.

Empirically we see that less flexible idiomatic expressions, such as "red tape", are rarely used in metadata, especially in the hierarchical cases. Therefore it is often the case that the recognized multiword relates closely to the original token (as with "gospel singing" and "singing"). Thus, such heuristics, by meaningfully enriching the sense sets, allow the target algorithms to better exploit (often scarce) background knowledge.

The evaluation of this heuristic in combination to the required tasks described above is discussed in Section 7.

6.5 Lightweight Parsing

6.5.1 Problems

In some of the analysed datasets the average label length is about 2 tokens, while in others the average label is more than 4 tokens long. This might raise the question whether there is a need to parse such short labels? However, we should not underestimate the nature of our domain. Being a natural language

metadata, our labels often represent a condensed view of information. For example, a single category name represents many instances of business services in the case of eCl@ss or of the web sites in the cases of DMoz and Yahoo. A single mistake in the interpretation of such information-dense label might lead to a frequent misclassification and drastic performance degradation of the target application.

Therefore, we need to apply a parser to a label to get a more precise view of a label structure.

The average maximum label length across our dataset is 18 tokens. Viewed as a sentence, it is not a particularly long or complex one. This leads us to a hypothesis, that perhaps, a full-blown parser might not be necessary in our case and a simple rule-based approach might be sufficient.

The information gathered on the previous steps of processing needs to be "woven" together to create, depending on the target application, complex concepts, or a structure of a logical formula. The most important element in this process is the syntactic structure of the label and the semantics of its pieces.

For example, knowing that the tokens in round brackets disambiguate the preceding tokens – as in the LCSH dataset – allows an application building a formula out of a label to exclude the tokens in round brackets from the formula and instead use the concepts they represent for disambiguating the concept expressed by preceding tokens.

Similarly, knowing in which case a comma separates a modifier of a preceding token as opposed to separating phrases allows the pipeline to construct an accurate formula.

In other cases, knowing that a label being processed represents a facet or a letter-bar, as labels such as "By Country" and "A-Z" often do, allows the pipeline to make a conclusion about this label and the labels in the hierarchy below this one and, perhaps, treat such labels in a special way.

6.5.2 Solution

We introduce a lightweight parser which makes the proposed solution more universal through the possibility of implementing different semantic actions and using the pipeline for purposes different from a translation into formal language. For example, the parser makes it possible to control the input language or automatically enrich the controlled vocabulary with unrecognized concepts, marking them for later refinement by an expert.

The results of our work with the POS tagger enabled us to perform an accurate analysis of the natural language metadata language structure. Using the best model available for a particular dataset, we processed the full dataset, tokenizing the labels and tagging the tokens with POS tags. For each label we derived a POS tag pattern. For example the label "Coconucos Range (Colombia)" is tokenized into a set of tokens with the following POS tags:

NNP	NNP	(	NNP	)
Coconucos	Range	(	Colombia	)

Table 7: Metadata language structure characteristics

Dataset	POS tag patterns	90% coverage	Grammar coverage, %
LCSH	13 342	1 007	99.45
NALT	275	10	99.05
DMoz	975	9	99.81
Yahoo	2 021	15	99.46
eCl@ss	1 496	360	92.70
UNSPSC	1 356	182	90.42

A POS tag pattern corresponding to this label is "NNP NNP (NNP)". We grouped the labels by their POS tag patterns and analysed the reuse of such POS patterns.

Table 7 summarizes some metadata language structure characteristics. One can note that the number of POS tag patterns needed to achieve 90% coverage of a dataset's labels is often small enough for manual analysis. The number of patterns in LCSH case is almost 3 times larger than the largest of all the other datasets. However, under a close inspection we found out that due to a particular comma use in LCSH, a much smaller set of patterns, similar to those of other datasets, occurs in these labels. When the patterns from this smaller set are joined sequentially with commas, they form the mentioned above larger set of patterns.

We developed a set of lightweight grammars for each of our datasets, starting from a base noun phrase grammar and modifying it to include the peculiarities of the noun phrases as they are used in the natural language metadata, such as combinations of noun phrases, use of commas and round brackets for disambiguation. The grammars we constructed can be divided into two categories: the simpler ones with nine and ten rules (DMoz, eCl@ss and UNSPSC) and a bit more complex ones with fifteen and seventeen rules (Yahoo, NALT and LCSH). Fig. 3 shows an example of such a more complex grammar which covers the LCSH dataset, while Fig. 4 shows an example of a smaller grammar which covers the UNSPSC dataset.

We use Backus-Nair form (BNF) for representing the grammar rules. Our example grammar starts with a top production rule `Heading`, which encodes the fact that LCSH headings are built of chunks of noun phrases, which we call `ForwardPhrase`. In turn, a `ForwardPhrase` may contains two phrases `DisPhrase` with disambiguation elements as in the example above.

Disambiguation elements may be a proper or a common noun phrase, surrounded by round brackets: (`ProperDis`) and (`NounDis`). `NounDis` is usually a period of time or a type of object, like in "Rumplemayer, Fenton (Fictitious character)", while `ProperDis` is usually a sequence of geographical named entities, for example "Whitemarsh Hall (Philadelphia, Pa.)".

The core of the grammar is a `Phrase` rule, corresponding to the variations of noun phrases encountered in this dataset. It follows a normal noun phrase sequence of: a determiner followed by adjectives, then by nouns. Alternatively,

```
 1 Heading := FwdPhrase {"," FwdPhrase}
 2 FwdPhrase := DisPhrase {Conn} DisPhrase
 3 DisPhrase := Phrase {"("ProperDis | NounDis")"}
 4 Phrase := [DT] Adjectives [Nouns] | [Proper] Nouns | Foreigns
 5 Adjectives := Adjective {[CC] Adjective}
 6 Nouns := Noun {Noun}
 7 Conn := ConjunctionConn | PrepositionConn
 8 Noun := NN [POS] | NNS [POS] | Period
 9 Adjective := JJ | JJR
10 ConjunctionConn := CC
11 PrepositionConn := IN | TO
12 Proper := NNP {NNP}
13 NounDis := CD | Phrase [":" Proper]
14 ProperDis := ProperSeq ":" Phrase | ProperSeq CC ProperSeq
15 Period := [TO] CD
16 ProperSeq := Proper ["," Proper]
17 Foreigns := FW {FW}
```

Figure 3: LCSH BNF grammar

it could be a noun(s) modified by a proper noun, or a sequence of foreign words. All the datasets' grammars share the nine base rules which start from the `Phrase` rule and encode the basic noun phrase grammar, with minor variations. Consider in Fig. 4 the grammar which covers the UNSPSC dataset and compare the rules 2-10 of this grammar with the rules 4-12 of the grammar in Fig. 3. One can note that these rules are almost identical.

To each rule we can attach some semantic actions. A semantic action can produce an atomic concept, a complex concept, a piece of a logical formula or its connective. These actions can also be used to treat specially a disambiguation element, a letter-bar label like "A-Z" or a facet-indicating label like "By

```
 1 Label := Phrase {Conn (Phrase | PP$ Label)}
 2 Phrase := Adjectives [Nouns] | Nouns
 3 Adjectives := Adjective {Adjective}
 4 Nouns := Noun {Noun}
 5 Conn := ConjunctionConn | PrepositionConn
 6 Noun := NN [POS] | NNS [POS] | DT RB JJ | Proper
 7 Adjective := JJ | JJR | CD | VBG
 8 ConjunctionConn := CC | ,
 9 PrepositionConn := IN | TO
10 Proper := NNP {NNP}
```

Figure 4: UNSPSC BNF grammar

Country".

In the column "Grammar coverage" of the Table 7 we show how many labels of the dataset are covered by our grammar. While the coverage is high, it does not reach 100%, as this is not possible with the flexibility of natural language. This opens two possibilities for the pipeline to process a small percentage of labels which are not covered by the grammar:

- In a more controlled setting it might behave like a controlled language and refuse to accept a label that does not conform to the grammar, asking the user to edit it.
- Alternatively, such labels could be processed by a simpler heuristic, put into a log file for a later editing and conversion, or even discarded.

6.6 Adding Robustness

Despite the fact that we target the domain of the natural language metadata with its special characteristics, some of our example applications, such as semantic search, need to process generic texts to extract concepts. A simple addition to our approach allows reusing it for such cases.

We employ the fact that our grammars are based on a noun phrase grammar and that a target application needs only complex concepts extracted out of a document. Therefore it is possible to apply our solution to a generic text, if it will be given some preprocessing.

To make generic texts "digestable" by our pipeline we add a chunker to the processing pipeline. A chunker is a standard natural language processing component, which identifies the high level syntactic structure of a sentence. For example, given a sentence "Green apples are juicier than red apples." it identifies the following components: "green apples", "are juicier than", "red apples", where the first and the last are actually, noun phrase chunks. This component, accompanied by the models trained for a generic texts, breaks the sentences of a document into chunks, selects only noun phrase chunks and passes them for further processing by our pipeline.

This allows employing the same processing algorithms which produce a formalism that a target application requires, be it complex concepts or propositional description logics formulas both for natural language metadata and for generic texts.

6.7 Word Sense Disambiguation

Word Sense Disambiguation is a standard problem in natural language processing. As SENSEVAL competition shows, this problem is noted for particularly difficult to beat simple baseline approaches. For example, a 4% improvement over a baseline is considered good [20]. We take the set of heuristics presented in [29], which reach comparable to the state of the art performance of 66.51% precision and adopt them to our framework.

Table 8: Evaluation results summary

Dataset	Labels	Accuracy, %	Previously, %	Improvement, %
source	2 854	83.43	67.73	+15.70
target	6 628	81.05	65.89	+15.16

7 Evaluation

We have evaluated the proposed solution for natural language metadata annotation using a synthetic approach. We have taken the large dataset [14] used for evaluation of semantic matching, which is a technique used to identify semantically related information by establishing a set of correspondences, usually between two tree-like structures which are often denoted as "source" and "target". This dataset is a composition of three web directories: Google, Yahoo! and Looksmart. The "source" part of it contains 2 854 labels, while the "target" part contains 6 628 labels. While containing parts of the Yahoo! directory and being from the same domain of natural language metadata, this dataset does not intersect with the ones we have used in our experiments and for training. Therefore it is appropriate to use it for evaluation purposes as it represents unseen data.

We have manually annotated this dataset with tokens, POS tags, named entity information, assigned correct senses from WordNet and, finally, created correct logical formulas for every label. Thus we created a golden standard, which enables us to evaluate our solution.

Table 8 summarizes the evaluation results. The column "Accuracy" contains the percentage of labels, for which the pipeline created correct formulas while the column "Previously" contains the accuracy of a previously used solution [15]. As it can be seen we have obtained a substantial improvement of approximately 15% over the previous results.

An analysis of mistakes showed that 19.87% (source) and 26.01% (target) of labels contained incorrectly recognized atomic concepts. For example, in the label "Diesel, Vin" two concepts "Diesel" and "Vin" were recognized, instead of a single one: "Vin_Diesel". Vice versa, in the label "Review Hubs", instead of two concepts "Review" and "Hubs", only one wrong concept "Review_Hubs" was recognized. The cause of these mistakes is the POS tagger error. Namely, as already noticed in the Section 6.2, the most frequent misclassification between proper and common nouns. For these cases further analysis of the erroneous formula does not make sense, because the atomic concepts are the basic building blocks of the formula, which should be recognized properly for the formula to be correct. For the rest, that is for the labels with correctly recognized atomic concepts, we found out that in 49.54% (source) and 52.28% (target) of cases the formula structure (that is, logical connectors or "bracketing") was recognized incorrectly. For example, in the label "Best & Worst Sites" the "&" sign is used as a conjunction, but was not recognized and this resulted in a wrong formula structure. The remaining half of the mistakes are word sense disambiguation

mistakes of different kinds. In some cases, 40.26% (source) and 41.11% (target) the algorithm pruned too much senses, leaving out the correct ones. For example, in the label "Cult Movies" the disambiguation algorithm pruned all senses of the concept "Cult" due to the POS tagger mistake. Similarly, in the label "Marching" the algorithm pruned correct senses due to the POS tagger mistake, which led to treating the word as a different part of speech. In the remaining 10.20% (source) and 6.61% (target) of cases the algorithm kept some extra senses that should have been pruned. In this category noticeable are the examples with named entities, represented by common words. For example, in the label "Matrix Series" the concept "Matrix" refers to the movie. The "movie" sense of the word "matrix" is not present in the vocabulary, which, instead, contains many other senses of the word "matrix". It is interesting to note that the movie itself was recognized correctly in the label "Matrix, The" located one level below this one, as "The_Matrix", although due to the lack of a sense in the vocabulary, the label remained senseless. Another similar example is provided by the label "Queen", which refers to the famous music band.

8 Discussion

In this article we presented a flexible solution for understanding natural language metadata. Our solution can be customized and it can power various target applications which operate on natural language metadata, such as semantic matching, hierarchical classification and search.

We believe that the following problems should be addressed in our future work:

1. First and foremost, an evaluation which shows the impact of our solution on the performance of the target applications needs to be performed. Although many target applications currently use some kind of a heuristic or ad-hoc approaches to understand natural language metadata we would like to explore the impact of a systematic approach on different target applications, as well as the most important factors, influencing the final performance of the target application from the point of view of understanding natural language metadata.

2. Second, we would like to explore and address different issues in various modules of the proposed solution.

   To begin with, while the performance of current state of the art language processing algorithms is similar, it is similar with regard to a normal text. Therefore, a thorough testing of other state of the art language processing algorithms is needed to confirm our hypothesis that this behaviour continues in the domain of natural language metadata.

   In addition, our target applications heavily use various sources of background knowledge, such as WordNet, to form a concept space and a base

for reasoning. We think that our solution will benefit from a more extensive use of the target application background knowledge along the processing pipeline and possibilities of tighter integration of knowledge-centric language processing methods are worth exploration.

Moreover, in different modules of our approach we see a number of issues worth exploring, such as using semantic parsers to disambiguate the formula structure, generalizing and unifying the grammars we developed, applying minimally supervised word sense disambiguation algorithms and exploring the possibility of enriching the linguistic background knowledge in the interactive usage scenario.

3. Third, we see using a non-sequential decision making architecture as an interesting alternative to a classic pipeline architecture.

References

[1] Adam L. Berger, Stephen D. Della Pietra, and Vincent J. D. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[2] Abraham Bernstein and Esther Kaufmann. GINO - a guided input natural language ontology editor. In *International Semantic Web Conference*, pages 144–157, 2006.

[3] John Blitzer, Ryan Mcdonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *In EMNLP*, 2006.

[4] Nicoletta Calzolari, Charles Fillmore, Ralph Grishman, Nancy Ide, Alessandro Lenci, Catherine MacLeod, and Antonio Zampolli. Towards best practice for multiword expressions in computational lexicons. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1941–1947, 2002.

[5] Anne Cregan, Rolf Schwitter, and Thomas Meyer. Sydney OWL syntax - towards a controlled natural language syntax for OWL 1.1. In *OWLED*, 2007.

[6] Michael Crystal, Alex Baron, Katherine Godfrey, Linnea Micciulla, Yvette J. Tenney, and Ralph M. Weischedel. A methodology for extrinsically evaluating information extraction performance. In *HLT/EMNLP*, 2005.

[7] Ronald Denaux, Vania Dimitrova, Anthony G. Cohn, Catherine Dolbear, and Glen Hart. Rabbit to OWL: Ontology authoring with a CNL-Based tool. In *Workshop on Controlled Natural Language (CNL 2009)*, 2009.

[8] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May 1998.

[9] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, 2005.

[10] Norbert E. Fuchs, Kaarel Kaljurand, and Gerold Schneider. Attempto controlled english meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In *FLAIRS Conference*, pages 664–669, 2006.

[11] Norbert E. Fuchs and Rolf Schwitter. Web-annotations for humans and machines. In *ESWC*, pages 458–472, 2007.

[12] Fausto Giunchiglia, Uladzimir Kharkevich, and Ilya Zaihrayeu. Concept search. In *ESWC*, pages 429–444, 2009.

[13] Fausto Giunchiglia, Vincenzo Maltese, and Aliaksandr Autayeu. Computing minimal mappings. In *Proc. of the 4th Ontology Matching Workshop*, 2009.

[14] Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko. A large dataset for the evaluation of ontology matching systems. *The Knowledge Engineering Review Journal*, 24:137–157, 2008.

[15] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: algorithms and implementation. In *Journal on Data Semantics, IX*, 2007.

[16] Fausto Giunchiglia and Ilya Zaihrayeu. Lightweight ontologies. In *Encyclopedia of Database Systems*, pages 1613–1619. 2009.

[17] Fausto Giunchiglia, Ilya Zaihrayeu, and Uladzimir Kharkevich. Formalizing the get-specific document classification algorithm. In *ECDL*, pages 26–37, 2007.

[18] J. Hendler. Agents and the semantic web. *IEEE Intelligent Systems*, 16(2), 2001.

[19] Martin Hepp and Jos de Bruijn. GenTax: A generic methodology for deriving OWL and RDF-S ontologies from hierarchical classifications, thesauri, and inconsistent taxonomies. In *ESWC*, pages 129–144, 2007.

[20] Rada Mihalcea and Andras Csomai. SenseLearner: Word sense disambiguation for all words in unrestricted text. In *ACL*, 2005.

[21] Jonathan Pool. Can controlled languages scale to the web? In *5th International Workshop on Controlled Language Applications (CLAW 2006 at AMTA 2006)*, 2006.

[22] A. Ratnaparkhi. A maximum entropy part of speech tagger. In *Proceeding of the Conference on Empirical Methods in Natural Language Processing*. University of Pennsylvania, 1996.

[23] Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. Multiword expressions: A pain in the neck for NLP. In *CICLing*, pages 1–15, 2002.

[24] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0306050, 2003.

[25] Rolf Schwitter, A. Ljungberg, and David Hood. ECOLE — a look-ahead editor for a controlled language. In *Proceedings of EAMT-CLAW03*, pages 141–150, 2003.

[26] Rolf Schwitter and Marc Tilbrook. Controlled natural language meets the semantic web, 2004.

[27] Rolf Schwitter and Marc Tilbrook. Lets talk in description logic via controlled natural language. In *Logic and Engineering of Natural Language Semantics (LENLS2006)*, 2006.

[28] Chong Wang, Miao Xiong, Qi Zhou, and Yong Yu. PANTO: A portable natural language interface to ontologies. In *ESWC*, pages 473–487, 2007.

[29] Ilya Zaihrayeu, Lei Sun, Fausto Giunchiglia, Wei Pan, Qi Ju, Mingmin Chi, and Xuanjing Huang. From web directories to ontologies: Natural language processing challenges. In *ISWC/ASWC*, pages 623–636, 2007.