DESCRIPTIVE PHRASES: UNDERSTANDING NATURAL LANGUAGE METADATA

Aliaksandr Autayeu

April 2010

Technical Report # DISI-10-032

PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Descriptive Phrases: Understanding Natural Language Metadata

Aliaksandr Autayeu

Advisor:

Prof. Fausto Giunchiglia

Università degli Studi di Trento

April 2010

Abstract

Fast development of information and communication technologies made available vast amounts of heterogeneous information. With these amounts growing faster and faster, information integration and search technologies are becoming a key for the success of information society. To handle such amounts efficiently, data needs to be leveraged and analysed at deep levels. Metadata is a traditional way of getting leverage over the data. Deeper levels of analysis include language analysis, starting from purely string-based (keyword) approaches, continuing with syntactic-based approaches and now semantics is about to be included in the processing loop.

Metadata gives a leverage over the data. Often a natural language, being the easiest way of expression, is used in metadata. We call such metadata "natural language metadata". The examples include various titles, captions and labels, such as web directory labels, picture titles, classification labels, business directory category names. These short pieces of text usually describe (sets of) objects. We call them "descriptive phrases". This thesis deals with a problem of understanding natural language metadata for its further use in semantics aware applications.

This thesis contributes by portraying descriptive phrases, using the results of analysis of several collected and annotated datasets of natural language metadata. It provides an architecture for the natural language metadata understanding, complete with the algorithms and the implementation. This thesis contains the evaluation of the proposed architecture.

Keywords
descriptive phrases, natural language metadata, natural language processing, natural to formal language translation

Acknowledgements

I thank my colleagues, friends and family.

I particularly thank:

Fausto Giunchiglia, my scientific advisor, for all the patience, advice and teaching he gave me. Fausto, I am grateful for having you as my teacher. You are a great visionary. I have learned a lot from you and continue to learn. Thank you.

Pierre Andrews, whose support on the finishing stages of my PhD was invaluable. Pierre, it is a pleasure to work and talk with you. I wish you would have joined our group earlier. Thank you for the productive collaboration.

Alex Malevich, my former advisor, for inspiring me to pursue this path.

Raffaella Bernardi and **Roberto Basili**, for reviewing the thesis and providing valuable comments.

Marina Repich and **Andrei Papliatseyeu**, **Ruslan Asaula**, **Aliaksandr Birukou** and **Olga Bryl** for being great friends and a great company. Marina, your kind words in tough times mean a lot for me. Olga, your exquisite humour is a delicacy one rarely finds. Ruslan, our mountain walks were a joy. Aliaksandr, your talent to organize and move things on inspired me to continue to "push, push, push".

Mikalai Krapivin for interesting and productive collaboration and joyful chats, as well as serious conversations. Nick, thank you.

Enzo Maltese, **Juan Pane**, **Pavel Shvaiko** and **Uladzimir Kharkevich** for being great colleagues. Sharing ideas, working together and receiving your feedback was useful and pleasant.

Margherita Sini and **Johannes Keizer** for interesting, intensive and productive collaboration in the thesauri matching experiment.

Tanya Yatskevich for few, but ever-so-important conversations.

Natalia Pirjanowicz for many artistic collaborations which kept my mind open and reminded me about life outside the university.

Gianluca Samarelli for teaching me to "sail through life". Gianluca, you are a gem. Our deep and funny conversations cheered my soul and helped me to understand better the italian way of doing things.

Luca Valenzin for wise words and a fine blend of venetian and american humour. Luca, it is a great pleasure and fun to do all these small and big down-to-earth projects with you.

Laura Maino and **Giuliano Zendri** for shedding just a bit of light on the mysteries of Trentino people and the land.

Raffaella Lenzi for opening the world of italian language for me and helping to make my experience smooth.

Javier Tordable for an inspiring collaboration and a great experience. Javier, I learn a lot from you. Thank you!

Galina Kamburova, **Diana Mardare**, **Alessandro Tomasi**, **Flavia Evandri** and **Manuel Zucchellini** for the insiders' help in fighting the bureaucracy.

My family have started and supports me in my journey through life. My curiosity is a great achievement of my grandad and my beloved granny. My persistence is a trait carefully raised by my mom and my granny. Your efforts in upbringing and your support helped me to arrive to the graduation. Mom and Granny, I love you and I am always with you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

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 ease 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*, goes with every significant piece of information.

Computers create increasing amounts of metadata automatically and search well through most of it by filter search, that returns those records, where certain field has the exact specified value, and both the field and the value have predetermined meaning. 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 e-mail 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 proper 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 work on lightweight ontologies [34], such as the "get specific" algorithm [35] for classification of documents in hierarchies or S-Match [33] and minimal S-Match [29] for matching of ontologies. 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 [38]: 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, because application developers have no incentive to build applications which has no data to work with and users have no incentive to annotate data unless there is a "killer app". We can come closer to solving the chicken-and-egg problem by providing application developers with a solution which is easy to use in applications. Through developers using our solution we lower the cost of annotation for users, making it "a by-product of normal computer use".

Expecting the users to write in formal language is unrealistic and while the coding standard determines the semantics for automatically produced metadata, 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 applications which rely on semantics and logical reasoning to reason about the data and thus give us the ability to leverage semantic services on our data. To do such translation one can use modern natural language processing (NLP) tools. However, they have evolved over the domain of newswire or similar text and the language used in natural language metadata differs from the one used in normal texts, such as news stories and books.

We briefly describe three applications which provide semantic services before discussing our natural language metadata translation solution that enables such semantic services.

Semantic Matching One can see semantic matching as an operator that takes two tree-like structures made of labeled nodes (such as classifications or schemas) and produces links between those tree nodes that correspond semantically to each other. Semantic matching employs two key ideas: a) it produces links with semantic relations such as *equivalence* or *more general*; and b) it calculates them by analysing the meaning (concepts) encoded in the labels of the input trees [33].

However, semantic matching algorithms need to create the formal representation of the concept of each tree node label. Most often the tree node labels are written in natural language, and, therefore, as a first step towards reasoning and calculation of correspondences between the tree nodes, the algorithm needs to translate natural language labels into their formal counterparts, for example, in case of S-Match [33] into propositional description logic formulas.

Semantic Classification Hierarchical classifications represent a natural way of organizing knowledge. However, keeping them up-to-date requires putting new information items (for example documents) into the proper places in the hierarchy. The "get-specific" algorithm [35] addresses this problem. This algorithm follows a knowledge-centric approach and first converts a natural language classification into a formal classification, which uses a concept language to express the meaning of labels. 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 handles arbitrary sequences of characters and calculates string similarity, to a semantic search, which handles concepts and calculates semantic relatedness [28]. However, everybody writes documents and search terms in a natural language, where concepts need to be identified first to enable semantic search.

Once again we face 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 suitable for the solution we propose.

Applications which use a formal counterpart of a natural language metadata, such as the ones we described, motivated our studies. These applications can benefit from an improved understanding of natural language metadata. They often operate on atomic concepts, such as the one described by the word "apple" and complex concepts, such as the one described by the phrase "green apples and red oranges". Many of them need the same steps of processing:

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

1.2 The Problem

As outlined in the introduction:

> The main problem is to provide an easy way to supply accurate semantics to (logic-powered) applications that work with (meta) data expressed using (a subset of) natural language.

To explain the problem better, we descend one level of details lower. The main problem contains several sub-problems:

- Identifying, studying and describing the subset of natural language that the target applications use;
- Keeping a balance between expressivity and computational complexity of the language subset and the logic formalism, while choosing a subset of the language to process and a logic formalism with its expressivity;
- Creating a language-to-logic processing architecture and algorithms, adapting the state of the art natural language processing algorithms as much as possible and creating new ones to substitute the unadaptable or missing ones, keeping in mind the cost of adaptation, lack of linguistic resources and computational complexity of the algorithms;
- Exploiting the user availability in some scenarios without overloading or intimidating the user with new and complex tasks;
- Evaluating the whole translation task as opposed to evaluating its separate steps.

1.3 Challenges and Objectives

1.3.1 Challenges

Semantically aware algorithms based on logics show promising results in such area as semantic matching. We believe other application areas like search and document classification can also benefit from employing semantics. However, high cost of producing semantically annotated data and the problem of precise natural language to logic translation prevents this approach from scaling.

Many users do not annotate their data because there are few convenient annotation tools and, more importantly, there are few tools for extracting added value out of annotated data. Software relying on semantically rich data does not appear because of the absence of critical mass of semantically annotated data. This resembles a vicious circle or a chicken-and-egg problem. One of the proposed solutions is to develop tools which add semantics to the data for free, as “a by-product of normal computer use”.

There are many discussions about bootstrapping semantically rich applications. One can broadly divide presented approaches into two high-level groups. The first one is a more traditional, academic approach of bootstrapping “bottom-up”. The essence of this approach is to embed semantic annotations right into the data. The difficulties of this approach are well-known. First, a critical amount of knowledge should be captured into knowledge bases. Second, a significant number of tools, able to use knowledge bases and create semantically annotated content should be created. Third, these tools should reach the users and become widely used. Usually users should learn to use new tools. Relaxing this requirement might speed up the adoption.

The second group contains more recent approaches. They are “top-down” approaches which rely on analysing existing information using natural language processing technology. These approaches have their costs too: high computational costs of natural language processing techniques and their precision and reliability. While the precision of some steps of these techniques is above 90%, for many this boundary remains to be crossed and the overall performance is far from satisfactory.

Another challenge to face is connected with the use of logics at the core of semantically rich applications. Formal logics is a well-developed and flexible instrument which one can use to power applications involving semantics. There are mature reasoners for different logic formalisms. However, it is almost impossible to imagine an average user typing in data using some logic formalism.

One more challenge not to overlook is the complexity of most logic formalisms. While being tractable on a case-by-case basis, while reasoners are becoming more and more advanced, while they handle more and more expressive formalisms and their robustness increases, the computational complexity of most of these formalisms remains high and prevents their application on a large-scale and in real-time.

Existing logical formalisms cover a wide spectrum and vary from ones having limited expressivity to very rich languages. The computational complexity correlates with the expressivity of the formalism. This introduces a challenge of keeping a balance between expressivity and computational complexity of the formalism.

While formal logic has many advantages over natural language, it is artificial and just seeing logical formalisms intimidates the user. A natural language interface remains much more suitable and easier to use for most tasks that information workers handle daily. However, using natural language interface introduces many challenges. Natural language is often ambiguous and its processing, especially at high levels, has significant computational costs.

Using natural language to interface with a semantically enabled application helps the users, while using logics in the core of the application is a way to power the application with semantics. However, one needs to develop a good natural language to logic translation to use logic in semantically enabled applications with natural language interface.

Having semantically annotated data as a "by-product of a normal computer use" remains an ideal yet to reach. On the road to this ideal, one has to involve the user in the process of creating semantically annotated data. A challenge here is to avoid overloading the user and finding out those points, where the user intervention will have the most positive and significant impact on the quality of the semantic annotations.

1.3.2 Objectives

Following our challenges, we set up several objectives.

Addressing the challenge of computational complexity of the natural language, an objective is to choose and describe suitable subset of the natural language. It should be a subset of the natural language and not an artificial creation, to avoid posing the requirement of learning another formalism. This subset should allow a natural use of language, it should not be restrictive. The chosen subset should be expressive enough, allowing users to write in a language they already use for similar tasks.

Addressing the challenge of computational and conceptual complexity of various logical formalisms, an objective is to choose a minimal logical formalism still enabling a sufficient number of motivating applications. This logical formalism should be simple enough to have low computational and conceptual complexity, but sufficiently expressive to power several applications. The formalism should allow expressing most concepts and structures in a chosen subset of the language.

Addressing the challenge of translation from natural to formal language, an objective is to develop a translation architecture, which motivating applications can exploit. The architecture should be flexible to allow tuning for a specific application. For example, the architecture should allow modifications of the modules or modality of the translation, such as user-assisted, fully or semi-automatic processing.

Addressing the challenge of constant changes in the language and in the requirements of target applications, the solution should allow modifications on the language side, as well as on the logic side. It should be adaptable to specific constructs of the input language and the output formalism of the target application.

Other objectives of the thesis follow from already established ones and are required to complete the picture. Therefore we include as an objective to develop the algorithms and the models to solve natural language processing problems specific to natural language metadata. Logical consequence of the algorithm development is an objective to evaluate the proposed solution.

1.4 Proposed Solution

Users daily create new documents, including text documents and multimedia documents like photos, illustrations and video clips. For future identification and reuse, users label the created artefacts using document titles, folder and file names, tags, subject and category. These mentioned examples of natural language labels share many features in common: they are widely used; they are short; they have simple grammar; they *describe objects*. We call these short text labels *descriptive phrases*. This thesis defines descriptive phrases, shows their properties as they are exhibited by several collected and annotated datasets of metadata and provides lightweight grammars for parsing descriptive phrases.

Descriptive phrases can bridge the “top-down” and “bottom-up” approaches and enable applications to use semantics more easily than currently possible. Descriptive phrases can be a computationally simple and powerful tool for solving the problem of producing semantically annotated data. This can improve applications like semantic matching, semantic search and automatic document classification. This thesis shows the use of descriptive phrases in semantic matching for translating short natural language labels to logics.

Descriptive phrases are short noun phrases joined with conjunctions and prepositions. However, they are more complicated in many aspects. Usually they are ambiguous on many levels of NLP. Often they lack context or context is indirectly expressed and loosely connected with the phrase instance. For example, the context for an image title can be other image titles in the same folder or on the same page, or the surrounding text. They show different statistical features than normal text, such as news stories and books. They contain less information than traditional full-fledged phrases. This thesis provides an architecture and algorithms addressing these issues.

On one side, we see descriptive phrases as a natural language tool for describing objects. On the other side, we have a well-known logic formalism serving the same purpose of describing objects called description logics. The connection between descriptive phrases and description logics is very important. These formalisms are expressive and there are different dialects of description logics with varying degrees of computational complexity. Moreover, the availability of mature reasoners for description logics makes this connection even more desirable for applications. In this thesis we select an easily tractable subset of description logics as a target language for natural language to logics translation.

Semantically aware applications are supposed to increase the quality of information processing. If we look at different stages of information processing we see that in some of them the user takes the role of consumer and in some others the role of creator. When looking at ways to empower the applications semantically, this dichotomy is of use.

On one hand, we have the user-consumer. In this role the user consumes the information that exists. The task is to enrich existing information with semantic mark-up. This mark-up, in turn, enables the applications to make use of the semantics of the information presented in a form of a natural language.

On the other hand, we have the user-creator. In this role the user creates new information. Creating content is a complex task. One of the difficulties writers have to care about is the clarity of their content. Resolving ambiguity is not easy for humans and is a very hard task for computers. Smart semantically aware applications will take advantage of the user availability during the information creation. They will exploit user availability to make the semantic mark-up more precise and less ambiguous.

The architecture we propose for the tools which need semantic annotations reflects the dichotomy we described. The applications should consider the case of available data without annotation and should care about "putting the user in the loop" during creation of semantically annotated content. Putting the user in the right place of the loop and using the right degree of involvement is a step towards having semantically annotated data as a "by-product of a normal computer use".

Whether the solution is user-assisted or fully automated, one needs algorithms to process the proposed subset of a language into the proposed logical formalism. Given the specificity of the chosen subset of natural language, the history and the state of the art in natural language processing, for some problems we can adapt existing algorithms, while for the others we develop new algorithms. To make our solution complete and practical, we propose a set of new and adapted algorithms and language models for the respective modules of the proposed architecture to tackle the natural language processing problems specific to natural language metadata.

To summarize, the proposed solution is the architecture with a set of algorithms to translate descriptive phrases into propositional description logics formulas.

1.5 Contributions of the Thesis

This PhD thesis contains the following contributions:

- *Descriptive phrases and natural language metadata.* The thesis portrays descriptive phrases and their properties, by providing an analysis of several collected and annotated sets of natural language metadata.

- *Architecture and implementation.* The thesis provides a modular architecture for understanding natural language metadata, algorithms for each module and their implementations.

- *Robust processing.* The thesis shows how the proposed architecture can be used to robustly enrich with semantics generic natural language texts.

- *User involvement.* The thesis shows the steps in processing where user involvement is the most efficient and for one of them proposes an aid, word sense summarization algorithm, for the complex and cognitively demanding word sense disambiguation task.

- *Word Sense Summarization.* The thesis proposes an algorithm to summarize word senses, contained in lexical databases such as WordNet. The thesis provides an evaluation of this algorithm.

- *Evaluation.* The thesis provides an evaluation of the proposed architecture.

1.6 Derivative Works

The following publications have been derived out of the contents of this thesis:

- Aliaksandr Autayeu, Fausto Giunchiglia, Pierre Andrews, and Qi Ju. Lightweight parsing of natural language metadata. In Proceedings of First Natural Language Processing for Digital Libraries Workshop, 2009.

- Aliaksandr Autayeu, Vincenzo Maltese, and Pierre Andrews. Recommendations for qualitative ontology matching evaluations. In Proceedings of Ontology Matching Workshop, 8th International Semantic Web Conference, 2009.

- Fausto Giunchiglia, Vincenzo Maltese, and Aliaksandr Autayeu. Computing minimal mappings. In Proceedings of Ontology Matching Workshop, 8th International Semantic Web Conference, 2009

The following publications have been submitted using the contents of this thesis:

- Aliaksandr Autayeu, Fausto Giunchiglia, and Pierre Andrews. Lightweight Parsing of Classifications into Lightweight Ontologies. In Proceedings of European Conference on Digital Libraries, 2010

- Aliaksandr Autayeu, Fausto Giunchiglia, and Pierre Andrews. Understanding Natural Language Metadata. IEEE Internet Computing, 2010

The implementation of the solution proposed in this thesis is being prepared for the release under an open source license as a part of S-Match semantic matching framework[1].

1.7 Structure of the Thesis

We structure the thesis as follows. In Chapter 2 we present the state of the art that includes the approaches to translation of natural language into logical formalisms and controlled languages as predominant means of bridging the gap between natural and formal languages.

Chapter 3 provides an intuitive notion of descriptive phrases, develops it into a basic definition and presents the analysis of samples of natural language metadata expressed in descriptive phrases, thus backing the initial intuition and developing the basic definition into a set of grammars, describing descriptive phrases "in vivo".

Chapter 4 presents the architecture for understanding natural language metadata with the models and the algorithms of its modules based on the results of the analysis presented in Chapter 3.

Chapter 5 presents the solution which aids the user in the word sense disambiguation task and consists of the algorithms, the user interface prototype and the evaluation of the presented algorithms.

Chapter 6 describes the evaluation of the proposed architecture as a whole using two large manually annotated datasets.

Chapter 7 shows possible applications of the proposed solution and demonstrates user-assisted and automatic processing modes of the proposed solution on the example applications.

Chapter 8 concludes the thesis by summarizing the problem and the proposed solution and outlines the future work.

[1]`http://semanticmatching.org`

Chapter 2

State of the Art

Many algorithms are based on reasoning in a formal language. However, users are accustomed to natural language and it is difficult for them to use a formal one. A number of approaches have been proposed to bridge the gap between formal and natural languages, most of them are based on a controlled-language approaches. Controlled languages as a solution have been developed starting both from the language side [63] and from the logic side [7].

The existing approaches and solutions differ with respect to several parameters. These parameters include:

- the main motivating application, such as ontology authoring or question answering;
- the supported natural language, such as English or Spanish;
- the language domain, such as general texts, medical texts, aerospace communication messages;
- the breadth of language support, such as which natural constructions of the language are permitted;
- the language grammar formalism used, such as context-free grammars, transformational grammars or definite clause grammars;

- the logic formalism used for output, such as propositional logic, description logics or first-order logic;
- the degree of lexicalization, such as whether the solution includes the lexicon or not;
- the availability of a solution, ranging from a paper to a working implementation;

Controlled languages are introduced for different purposes, such as to ease the readability of the language, to reduce its complexity, to ease the translation and to represent the knowledge expressed in natural language in a machine-tractable form. In fact, questions such as "Are [the controlled language] statements translated into a logic?" already appear explicitly in discussions about controlled languages [78], and in case of a positive answer the discussion then turns to questions about expressivity of the target logical formalism and its computational complexity. The controlled languages have their niche, because despite in last decades a progress has been made in natural language processing techniques and in controlled languages, yet there are examples of problems expressed in natural language where the language itself is simple, but no natural language processing system can take such a puzzle as input, translate it into a logical formalism and solve it automatically [64].

Controlled languages, such as Attempto [23, 21, 22], have been proposed as a solution to a number of knowledge representation and interoperability tasks. Attempto has been proposed as an interface between natural language and first-order logic. Attempto has been applied as a front-end to replace first-order logic as an input language of the model generation method EP Tableaux [24, 25] and as a front-end for an ontology query language PQL [9]. It has been mapped to OWL DL [43] and vice versa [42], so far as to extending its use to the verbalization of the ontologies

[44]. It has been applied to text mining within biomedical domain [45] and has shows 56% accuracy in representing paragraph headings extracted from the biomedical literature. Attempto has well-documented syntax [41] and handles many interesting language features, such as plural ambiguities [61]. The Attempto Parsing Engine implements the language using a definite clause grammar (DCG) written in Prolog.

PENG [62] is another controlled language, defined by the authors as "a computer-processable controlled natural language designed for writing unambiguous and precise specifications". It covers a strict subset of standard English and is defined by the lexicon and the controlled grammar. Specifications written in PENG can be translated into first-order predicate logic. The difficulty of writing in a controlled language is addressed by a look-ahead editor, ECOLE [67]. PENG lacks, however, phrase level coordination for noun phrases located in subject position. In addition to providing a look-ahead editor, PENG Light, introduced in [65], has been recently proposed to annotate web pages by means of a browser extension [75]. Interestingly, in [75], authors pay particular attention to processing unknown content words proposing a special syntax to handle their addition to the vocabulary. PENG Light [65] also features bi-directionality between logic and language.

CELT (Controlled English to Logic Translation) [53] is another attempt to build a controlled language interface for ontology editing. CELT converts controlled English to KIF [27] formulas using ontologies built with the Suggested Upper Merged Ontology (SUMO) [51]. It uses WordNet [18] as a source of base lexicon and word sense preference, Discourse Representation Theory to translate multiple sentences, and definite clause grammar to parse individual sentences. CELT is domain-independent and is supposed to be customized for particular domains by providing domain-specific ontologies and lexicons. However, ontologies themselves are normally used to capture domain knowledge, so in this light one might see the need of customization of CELT by providing domain-specific ontologies as creating a vicious circle. The authors continue their work in [52] by proposing PhraseBank – an extension to the lexicon of the language.

Lite Natural Language [7, 6] presents an attempt to bridge language and logic by creating a controlled natural language using a Categorial Grammar on the language side and a dialect of description logics, DL-Lite, on the formal side. The authors' approach to building the connection between logic and language is interesting on its own, because authors progress explicitly from the task of querying an ontology to identifying the logic formalism sufficient to fulfill the task to identifying which subset of language to use. The authors choose a subset of language containing those sentences, whose meaning representation could be expressed by DL-Lite. In [6] authors compare their work with other tractable subsets of English.

MetaLog [46] is an attempt to construct a Pseudo Natural Language (PNL) interface for accessing the Semantic Web by providing the query layer on top of RDF.

Rabbit [37, 17] is a controlled language that can be translated into OWL DL and provides easy access to the precision of a logical formalism to domain experts without the need to descend into low-level language syntax. Differently from some other controlled languages applied for similar purposes, such as SOS [13], which took origin from Attempto [23], Rabbit was developed independently and has some differences [66] with Attempto and SOS, the most noticeable of which is meta-level approach to axiom rendering (versus object-level in Attempto and SOS). Another interesting point in this approach is a methodological difference: as a starting point authors have chosen and intensively involved domain experts in the controlled language construction process. This is accomplished by a Protégé [2] plugin ROO [16]. GATE [15] performs lower-level language parsing tasks to power

Rabbit with required language processing facilities.

Sydney OWL Syntax (SOS) [13] is a controlled language establishing a bidirectional mapping between a subset of English and OWL 1.1. Its characteristics follow from several design choices made by its authors: prefer natural language versus closeness to OWL; prefer determinism and allow only one SOS syntax form for each OWL form; allow few explicit references to OWL constructs; use linguistic knowledge to some extent, but do not go as far as translating ontology as a whole at the expense of introducing anaphora resolution; use variables, but minimise their use. As a grammar formalism SOS uses definite clause grammar.

Controlled languages have also been proposed to bridge the gap between formal and natural languages in [26] by means of use in annotation, namely, the authors propose to *manually* annotate web pages, rightfully admitting that their proposal introduces a chicken-and-egg problem.

These, as well as a number of other proposals based on a controlled language approach [69, 68, 17], require users to learn the rules and the semantics of a subset of English to use controlled language efficiently. Moreover, users need to have some basic understanding of underlying logic to provide a meaningful input. The difficulty of writing in a controlled language can be illustrated (and tackled to some extent) by the existence of editors, such as ECOLE [67], aiding the user in editing the controlled language.

Controlled natural languages have been proposed as an interface for ontology authoring also in [8, 13]. The approach of [8] 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 [13] enforces a one-to-one correspondence between the controlled language and the ontology language and the authors prefer to leave only one of alternative language expressions for each OWL form. On the contrary, the authors in [17], following a practical experience, tailored their controlled language to the specific constructs and the errors of their users.

Many controlled languages have been proposed for different purposes, including language to logic translation. For example, author of [54] mentions 41 controlled language, originating from different natural languages. However, most of them have been critiqued for the lack of documentation and genre limitations. Few have available working and downloadable implementations. Contrary to the genre limitations of the most of them, the author critiques “the most expressive one” for its allowance of semantic ambiguity. Author concludes the criticism with outlining the emergence of two strategies: more formalistic, more precise languages and more expressive, but less precise languages and hopes for their convergence.

For querying purposes, [74] 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.

We also note few earlier approaches of translating structured language resources, such as thesauri, into formalisms such as RDF-S and OWL for the needs of Semantic Web. The GenTax approach [40, 39] of automatically translating hierarchical classifications into OWL ontologies is more interesting, because contrary to the others, it does not use a controlled language and its problem domain is similar to ours. 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.

The authors of [73] propose a methodology for converting thesauri to RDF and OWL. This methodology contains 4 steps, syntactic as well as semantic, and basically represents a set of guidelines covering the conversion process. A similar approach is used by the authors in [72] to convert thesauri to SKOS [1]. These approaches do not consider linguistic properties of terms being converted, they are treated just as (atomic) terms, without ascending to the (atomic or complex) concept level. Very early approaches to thesauri to ontology conversion are explored in [76] and [77].

As a part of solving the question answering and explanation problem, the authors of [49, 50] consider transformation of WordNet glosses to logic forms, which they see as an intermediate form between the syntactic parse and the deep semantic form. The authors take into account syntax-based relationships such as: 1) syntactic subjects, 2) syntactic objects, 3) prepositional attachments, 4) complex nominals, and 5) adjectival and adverbial adjunctsand ignore some linguistic phenomena such as plurals and sets, quantifiers, and few others. The authors use the output of a syntactic parser and the set of rules to perform the transformation. The weak point of the approach is the amount of rules needed to cover the language. The topic is further developed in [56] and in [57].

Differently from the mentioned above approaches, our work does not impose the requirement of having an ontology, users are not required to learn a syntax of a controlled language and are not restricted by it, and we do not restrict our consideration to a specific domain. We also encourage the cooperation between the user and the machine and try to involve the user in solving the problem by providing a user interface prototype. We do not consider covering a wide general subset of language, instead we start from a specific subset of it, described by several datasets, relevant to our task. Our approach, being modular, permits translation by adopting a controlled-language style parsing based on a manually created grammar (which is supplied by default), as well as the other parsing approaches, such as based on syntactic and dependency parsers.

Chapter 3

Descriptive Phrases

3.1 What are Descriptive Phrases?

We are surrounded by various physical objects and many times a day we refer them. We use concrete and abstract references, we refer to specific objects and sets of objects. And for at least the first reference we use a name for an object. We say: “Please, give me a red apple”. Few objects have their own, proper name, and mostly we refer to an object by using its class name (noun), if necessary augmenting it with a specifier (demonstrative pronoun): “Please, give me that red apple”. Often enough we use more complex combinations and refer to a set or sets of objects. We ask: “I would like some red and green apples.”

To refer to objects we describe them using a specific type of natural language phrase: noun phrase. A basic syntax of a noun phrase can be expressed by the following syntax shown in Figure 3.1.

```
NP := [DT] (JJ)* (NN)* NN(s)
```

Figure 3.1: Basic Noun Phrase Syntax.

In Figure 3.1 a noun phrase NP starts with an optional determiner [DT], followed by zero or more adjectives (JJ)*, followed by zero or more nouns (NN)* and finished by a noun, possible in plural form NN(s). This syntax allows to create quite expressive phrases such as the example in Figure 3.2.

DT	JJ	NN	NN
a	tasty	apple	juice

Figure 3.2: Basic Noun Phrase Example.

Often this is not enough and we combine such phrases to describe a desired combination of (sets of) objects. By adding a new syntax rule and allowing combinations of phrases we are able to say much more. Figure 3.3 shows this additional rule, which allows combining phrases with conjunctions (CC) or prepositions (IN). We call such phrases *descriptive phrases* (DP), because they *describe objects and sets of objects.*

```
DP := DP CC DP | DP IN DP
DP := [DT] (JJ)* (NN)* NN(s)
```

Figure 3.3: Basic Descriptive Phrase Syntax.

Descriptive phrases allow to describe complex combinations of objects and sets of objects, such as demonstrated by the examples in Figure 3.4.

DT	NN	CC	DT	JJ	NN	NN
an	apple	and	a	tasty	apple	juice
DT	JJ	NN	NN	CC	DT	NN
a	tasty	apple	juice	in	the	glass

Figure 3.4: Descriptive Phrase Examples.

3.2 Why Descriptive Phrases?

In the real world we use descriptive phrases to describe (sets of) objects and in the virtual world they come in handy when we need to define a set of documents about these objects. For example, when we search for "car engine" we want to get information items about car engines, such as documents, web pages, and images. Descriptive phrases is a natural language instrument to describe sets of (documents about) objects.

Formal instruments to represent sets of objects can be found in Description Logics. Description Logics (DLs) [4] are a set of logic formalisms that can be used to structure the domain of interest with concepts and roles. Concepts stand for sets of objects and roles stand for binary relations between (instances of) concepts. Concepts can be atomic and complex. Complex concepts are build out of atomic ones using constructs such as conjunction (&) and disjunction (|). Descriptive phrases, as they are used in natural language metadata, represent a static view of the world. They describe only (sets of) objects and as we will see in Section 4.2.3, because of the absence of verbs, concepts alone are sufficient and, by agreeing with some approximations in the processing of prepositions, we can choose a description logic formalism without roles.

Reasoning about sets of documents defined using a logic formalism is a convenient tool for many algorithms which work with (sets of) information items. As a logic formalism our "guinea pig" algorithms of semantic matching, search and classification, introduced in Section 1.1, use propositional Description Logics language L^C, introduced in [30].

The connection between Description Logics and natural language has been already noted (see Chapter 15 in [4]). Descriptive phrases are interesting, because on one hand, we have descriptive phrases as a natural language tool for describing sets of (documents about) objects and on the other hand we have propositional Description Logics language L^C which, while being propositional in nature, has set-theoretic semantics and its formulas describe sets of documents. According to the set-theoretic semantics of L^C, the interpretation of a concept is the set of documents about this concept. For example, the interpretation of a concept lexically expressed with the word “apple” is the set of documents about apples, and not the set of apples.

Descriptive phrases, as we will see in Section 3.3, form a small, but expressive enough subset of language to be tractable with simple and fast tools, such as short rule-based grammars, presented in Chapter 4. On the other hand, Description Logics contain many tractable subsets of different complexity and L^C in particular is simple and computationally efficient [30]. Studying descriptive phrases allows us to understand better their semantics, to adapt modern natural language processing tools and to develop a natural language metadata understanding architecture with accurate translation algorithms. Thus we establish a good balance between a tractable subset of natural language, formed by descriptive phrases, and a tractable Description Logic language L^C.

3.3 Samples and Syntax

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. The semantics of such items is not formalized and one has to extract it to enable automatic processing powered by reasoning over the meaning.

Upon a distant look, natural language metadata is represented mostly by descriptive phrases. In Section 3.1 we gave a basic definition of a descriptive phrase. A closer look at the samples of natural language metadata confirms our intuition and reveals that our definition is indeed basic and to make it useful we elaborate the definition to a set of a more detailed grammars. Here we study the phenomena of descriptive phrases using natural language processing tools. 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 subset of language [10]. We show that natural language metadata deserves to be treated as a separate subset of language and that it is mostly represented by descriptive phrases.

To study the natural language metadata, we have analysed the following datasets: DMoz, eCl@ss, LCSH, NALT, UNSPSC, Yahoo! Directory. These datasets belong to the 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 3.1 summarizes some key characteristics of these datasets, and in the following we provide a more detailed description and analysis. The column "Dataset" contains the dataset names which we will use to refer to them later. The column "Labels" shows the number of natural language labels the dataset contains. The column "Sample Size" contains the number of labels in the manually annotated sample of the dataset. The column "Unique Labels (%)" shows the percentage of unique labels in the dataset. The difference between labels and unique labels is similar to the difference between tokens and types in a corpus. Namely, a unique label might have several label instances in a corpus. The column "Levels" contains the number of levels in the dataset, that is, how many labels there are in the path from the

Table 3.1: Key Datasets' Characteristics.

Dataset	Labels	Sample Size	Unique Labels (%)	Levels	Label Length, NL tokens	
					Max	Avg
DMoz	494 043	27 975	25.46	12	12	1.8
eCl@ss	14 431	3 591	94.51	4	31	4.2
LCSH	335 704	44 490	100.00	21	24	4.0
NALT	43 038	13 624	100.00	13	8	1.6
UNSPSC	19 779	5 154	100.00	4	19	3.5
Yahoo	829 081	132 350	16.70	15	18	2.0

root label of the dataset to the deepest label in the dataset. The columns "Max" and "Avg" show the maximum and average length of the dataset label, measured in tokens, respectively. The minimum length of a dataset label is zero.

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 of 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 and used to classify products and services.

LCSH[4] 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.

[1] `http://dmoz.org`

[2] 114 sites according to DMoz

[3] `http://www.eclass-online.com/`

[4] `http://www.loc.gov/cds/lcsh.html`

NALT[5] 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.

UNSPSC[6] 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[7] is a "catalog of sites created by Yahoo! editors who visit and evaluate websites and then organize them into subject-based categories and subcategories".

In our analysis we extensively use PennTreeBank part of speech (POS) tag notation [60]. This notation defines a POS tag for each word class. The tag summarizes the part of speech and the form of the word. The tags are mostly two or three letter combinations, inspired by the name of the part of speech. For example, NN stands for a singular form of a noun (NouN, such as apple), while NNS stands for a plural form of a noun (NouN, such as appleS). Table 3.2 explains the tags we use the most. When we speak about POS tag pattern, we mean a sequence of POS tags, such as DT JJ NN NN, which originates from a phrase, such as "a tasty apple juice", as shown in Figure 3.2. Each phrase has only one POS tag pattern corresponding to it and each POS tag pattern has many phrases corresponding to it. This relation between phrases and POS tag patterns turns POS tag patterns into a powerful instrument of language analysis.

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

Table 3.2: Excerpt of PennTreeBank Tag Notation.

Tag	Part of Speech	Examples
CC	Coordinating conjunction	and, or
CD	Cardinal number	1, 14
DT	Determiner	a, the
FW	Foreign word	noir, persona non grata
IN	Preposition or subordinating conjunction	in, for
JJ	Adjective	red, soft
JJR	Adjective, comparative	better, more
JJS	Adjective, superlative	best, fastest
NN	Noun, singular or mass	apple, ox
NNS	Noun, plural	apples, oxen
NNP	Proper noun, singular	George Bush, John
NNPS	Proper noun, plural	Smiths
POS	Possessive ending	's, '
PP$	Possessive pronoun	theirs, ours
RB	Adverb	deeply, softly
VB	Verb, base form	be, go
VBD	Verb, past tense	were
VBG	Verb, gerund or present participle	going
VBN	Verb, past participle	granted

3.3.1 DMoz

```
...
        Business
            Arts and Entertainment
                Agents and Agencies
                    Film, Video, and Television Production
...
```

Figure 3.5: DMoz fragment.

We show the fragment of the DMoz dataset in Figure 3.5 to exemplify typical labels of this dataset. We have analyzed a DMoz dataset language patterns using the following sequence of steps:

1. Tag the dataset using OpenNLP POS tagger trained on an annotated sample of the dataset using PennTreeBank POS tags [60] and extended context, including the labels in the upper levels of hierarchy. This model tags 99.67% of tokens or 96.64% of labels correctly, as per 10-fold cross-validation procedure performed on an annotated sample.

2. Collect obtained POS tag patterns, sort them by frequency and analyze them manually, thus extracting the information about the language features used in this dataset and building the basis for both future grammar construction and translation procedure.

3. Split the patterns into two groups. The first group contains proper name labels in which all tokens have NNP tag or NNPS tag. The second group contains common labels, in which the tokens have other tags. This division was proposed first in [79] to simplify and speed up the processing by skipping some processing steps for proper name labels.

4. Analyze the common labels patterns to identify incorrectly tagged proper name patterns and exclude them from further analysis as mistakenly classified.

5. Analyze remaining common labels pattern group.

It should be noted that while tagger precision is high, it is not 100% and therefore, in some numbers and examples obtained with this model and presented here there is a slight margin for error. For example, as 0.33% of tokens might be tagged incorrectly, the shares reported in Table 3.4 might vary slightly.

Table 3.3: DMoz Labels Lengths Distribution. Top 5 Rows.

Token count	Label count	Share of labels (%)
1	251 140	50.8336
2	86 454	17.4993
3	136 425	27.6140
4	12 232	2.4759
5	3 762	0.7615

Table 3.4: DMoz Common Labels Lengths Distribution. Top 5 Rows.

Token count	Label count	Share of labels (%)
1	160 880	49.5968
2	49 035	15.1167
3	108 838	33.5530
4	3 821	01.1780
5	607	00.1871

Tokenization

The analyzed dataset consists of 494 043 labels, of them 125 797 (25.46%) are unique labels. More then a half of labels consists of 1 token. Table 3.3 shows the top 5 rows of the distribution of the label lengths for the complete dataset, while Table A.1 shows the complete version of the distribution.

In the common labels group the token count per label distribution is slightly different and Table 3.4 displays the top 5 rows of it, while Table A.2 shows the complete version.

POS Tags

We analyzed common labels group which consists of 310 710 labels (62.89% of all labels) or 578 311 tokens. Table 3.5 shows the top of the POS tag distribution among common labels, while Table A.3 shows the complete version of the distribution.

Table 3.5: DMoz Common Labels POS Distribution. Top 5 Rows.

POS Tag	Token count	Share of tokens (%)
NN	298 342	51.59
NNS	132 797	22.96
CC	107 006	18.50
JJ	36 888	6.38
,	2 173	0.38

Table 3.6: Top 5 DMoz POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
104 695	33.70	NN	Compensation
62 625	20.16	NN CC NN	Pregnancy and Birth
44 847	14.43	NNS	Sidecars
21 854	7.03	NNS CC NNS	Invitations and Announcements
13 047	4.20	NN NNS	Restaurant Chains

POS Tag Patterns

The final result of the analysis is a set of 232 POS tag patterns, which describes common labels used in DMOZ directory. These 232 patterns cover 310 710 labels, or 62.89% of all labels. Table 3.6 shows top 5 of the 20 patterns which cover almost 99% of common DMOZ labels, while Table A.4 shows all 20 patterns.

Qualitative Analysis

We have analysed in details two groups of patterns: discarded patterns and common patterns. The first group contains patterns which were discarded for various reasons on different steps of analysis. Analysing patterns gives us insight into their semantics, which in turn, allows translating them into logics better.

Discarded Patterns First group of patterns which was discarded consists of patterns made from proper nouns only. They were discarded automatically as they contain only NNP or NNPS tags and can be easily detected. We think that because of the 96.64% precision per label (PPL) accuracy of the tagger the mistake margin of discarding some useful common labels should be acceptable.

The second group of discarded patterns was a group of patterns where some of the proper noun tokens were mistakenly tagged as common nouns. In this group we can highlight the following subgroups, with pattern and label examples:

- **NNP-patterns** containing a lot of NNP tags with few common noun (NN) tags:
  NN NNP NNP, like "Beta Kappa Phi";
- **proper noun patterns mistakenly tagged as common**: NN NN NN CD, like "Combat Flight Simulator 3";
- **mixed labels in forward order**, containing proper name and noun, many of them ending with "Series": NNP NNP NN, like "Air Warrior Series", "Broken Sword Series";
- **mixed labels in backwards order**, containing noun and proper name, many of them being organizations like "Church", "Union", "College", "Institute": NN NN IN NNP, like "Art Institute of Colorado", "Art Academy of Cincinnati", "Baptist College of Florida';
- patterns with **mixed proper and common labels**: NN CC NN NN NN, like "Network and Operating System Management", "Life or Something Like It";
- patterns with **commas**, most of them names: NNS , NNP NNP, like "Vives, Juan Luis", "Vries, Hugo de";

- patterns with **other punctuation**, most of them movie or game titles: NN NNS : NNP NNP, like “Deception III – Dark Delusion”, “Circus Maximus – Chariot Wars”, “Ice Age – The Meltdown”;

Some of these patterns tend to group almost exclusively proper names of a specific kind, like organizations, person names, movie titles. Others contain several kinds of proper names. Although currently simply discarded, these patterns potentially may be used to correct the POS tagger.

The third is a broader group of patterns discarded after retagging and analysis. It includes several interesting subgroups of patterns:

- structural patterns, or **facet indicators**: IN NN, like “By Topic”, “By Movement”, “By Composer”; IN NN IN NN, like “By Source of Exposure”, “By Country of Service”; IN NN CC NN, like “By Province or Territory”, “By Room or Item”; IN NN NN, like “By Metro Area”, “By Age Group”; IN NN NN CC NN, like “By Metro Area and Region”; IN NNS, like “For Kids”, “By Kids”, “For Professionals”; IN JJ NNS, like “By Non-Native Artists”, “For Specific Publications”;

- **patterns with TO**, mostly movie titles: NN TO NN, like “Road to Perdition”, “Heavens to Betsy”;

- “Series” patterns: NNP NN, like “Tetsuo Series”, “Supercross Series”, “DrumMania Series”; NNP NNP NN, like “Ace Combat Series”, “Airborne Assault Series”

- **person names and movie titles in backward order**: NN , NN, like “Troup, Bobby”, “Faculty, The”; NNS , NN, like “Parks, Tim”, “Seahorses, The”; NN NN , NN, like “Signing Game, The”, “Wedding Present, The”;

- **movie titles**: NN DT NN, like “Cracking the Conspiracy”, “Wearing the Claw”;
- **“Based” labels**: NN VBN, like “Browser Based”, “Fee Based”, “Home Based”; NN VBN NN, like “Computer Based Training”, “Evidence Based Healthcare”; NN NN VBN, like “Scheduling Program Based”;
- **organizations in backward order**: NN , NN IN, like “Education, Faculty of”, “Engineering, College of”; JJ NNS , NN IN, like “Social Sciences, Faculty of”; NN NN , NN IN, like “Business Administration, Faculty of”, “Hotel Administration, School of”; NNS CC NN , NN IN, like “Arts and Science, School of”;
- **personalized organization names**. These labels stand somewhat in between proper “proper names” and “common names”, because on one hand they contain a proper name, but on the other hand they contain quite a lot of interpretable meaning: NNP NNP NNP NN CC NN, like “Korea University of Technology and Education”, “American Institute of Business and Economics”; NN NN IN NN CC JJR NN, like “Guildford College of Further and Higher Education”;
  NN NN NN IN NN CC NN, like “Oak Ridge Institute for Science and Education”;
- **mistakenly tagged** patterns: JJ CC JJ NNP NN, like “Radiological and Environmental Sciences Laboratory”; NN CC JJ NNP NN, like “Life and Environmental Sciences Division”; NN , NN , CC NNP NNS, like “Literature, Science, and the Arts”;
- **mistyped labels**, like those missing comma or ungrammatical:
  NNS NNS CC NN NN CC NN NNS, like “Faucets Fittings and Accessories Trim and Flush Valves”; NN NN CC NNS, like “Peace Love

and Pitbulls";

Some of these pattern groups encode special meaning or serve for structural purposes. We can use these patterns to recognize special kinds of labels. For example, structural patterns resemble facet names or facet indicators.

We encounter patterns in backward order, where the order of words is reversed. Compare the label "Faculty of Education" in normal, forward order with the same label in backward order: "Education, Faculty of". Other examples include movies names. Compare normal, forward word order in the "The Matrix" with the backward word order in the "Matrix, The".

Group of patterns covering organizations in backward order might be either included or excluded, depending on the requirements. Inclusion would increase language coverage; exclusion would increase homogeneity of allowed labels, that is, all labels will have more similar translation procedure.

We currently tag personalized organization names as proper nouns. However, we might treat them as common nouns, because it is possible to deduce most of their meaning in the same manner as with common nouns. The best way of tagging them depends on whether we want to match later pairs like "College of Liberal Arts" and "Texas College of Liberal Arts".

We currently tag the word "Based" in "Based" labels as VBN (verb, past participle), while JJ (adjective) might also be an option, if we consider semantic tagging approach. PennTreeBank tagging guidelines adopted syntactic approach, which we followed. They tag a word according to its syntactic function, while we are mainly interested in semantics. Following syntactic tagging approach makes our POS tagger more compatible with available parsers, and simultaneously makes the translation procedure, based on these grammars, a bit more complicated.

Common Label Patterns A majority of common label patterns preserve semantics and can be quite easily converted into logics. We would like to make some notes about the following groups of patterns:

- **Patterns with multiwords**. Some labels contain multiwords, like "Bed and Breakfast", "Home Theatre", which after recognition usually fall under different pattern, usually a shorter one, like NN or NNS and therefore, it is simpler to translate labels than it seems initially.

- **Potentially ambiguous patterns**. They usually contain CC, are quite long and seem to be, or actually are, ambiguous and need additional disambiguation efforts to be correctly translated into logics. We selected a set of 958 potentially ambiguous labels, manually disambiguated and analyzed them. The result is a set of 113 patterns with various degree of ambiguity. We present here the patterns having more than 1 instance label:

  - **Unambiguous patterns**. Table 3.7 shows the top 5 of 20 patterns which disambiguate always the same way, while Table A.5 shows all 20 patterns. In this table and the following tables we use propositional description logic formulas notation to show disambiguation option. This notation includes symbol | indicating logical disjunction (OR) and symbol & indicating logical conjunction (AND).

  - **Ambiguous patterns**. Table 3.8 shows the top 5 of 22 patterns which have 2 disambiguation options, while Table A.6 shows all patterns. In Table 3.8, for each pattern the column "Disambiguations" shows possible disambiguation options, followed by the share of each disambiguation option.

  - **Highly ambiguous patterns**. The pattern in Table 3.9 has

Table 3.7: Unambiguous DMoz POS Tag Patterns. Top 5 Rows.

POS Tag Pattern	Label Count	Share (%)	Disambiguation
NNS CC NN NNS	97	10.13	NNS | (NN & NNS)
NNS CC NN NN	21	2.19	NNS | (NN & NN)
NNS CC JJ NNS	15	1.57	NNS | (JJ & NNS)
JJ CC NN NNS	6	0.63	(JJ | NN) & NNS
NNS CC JJ NN	6	0.63	NNS | (JJ & NN)

Table 3.8: Ambiguous DMoz POS Tag Patterns. Top 5 Rows.

POS Tag Pattern	Label Count	Share (%)	Disambiguations	Share (%) / Share (%)
NN CC NN NN	165	17.22	NN | (NN & NN)	41.82
			(NN | NN) & NN	58.18
NN CC NN NNS	139	14.51	NN | (NN & NNS)	15.11
			(NN | NN) & NNS	84.89
NN NNS CC NNS	86	8.98	(NN & NNS) | NNS	13.95
			NN & (NNS | NNS)	86.05
NN NN CC NN	61	6.37	NN & (NN | NN)	73.77
			(NN & NN) | NN	26.23
JJ NNS CC NNS	51	5.32	JJ & (NNS | NNS)	88.24
			(JJ & NNS) | NNS	11.76

Table 3.9: Highly ambiguous DMoz POS Tag Patterns.

POS Tag Pattern	Label Count	Share (%)	Disambiguations
NN NN CC NN NNS	5	0.52	((NN & NN) | NN) & NNS
			NN & (NN | NN) & NNS
			(NN & NN) | (NN & NNS)

shown to be very ambiguous and has more than 3 disambiguation options.

- **Hanging modifiers**. Many labels are self-containing, however, there are labels which semantics become clear only in context. These ones might require special care in the translation procedure.

  - JJ is the most significant in this group, 3.15% of all common labels.
  - JJ , JJ , CC JJ. Quite frequently these modifiers repeat their constituents as their children.
  - JJ JJ
  - JJ , JJ CC JJ
  - NN CC JJ
  - JJ , JJ , JJ
  - JJ CD

- **"Wildcards"** or labels with complex semantics. These labels contain one of the words "other", "related", "its" or "not" and are not straightforwardly translatable into logics:
  - **Containing "other"**:
    * **as a first word**. Example patterns include: NN NNS, JJ NNS, NN NN, NN NN NNS, NN NNS CC NNS with labels such as "Other Themes", "Other Candidates", "Other Locations", "Other Products", "Other Media", "Other Payment Systems", "Other Times and Places"; As follows from the sibling nodes of these examples, possible logical formulas for these labels would be XX & $!(YY_1|\dots|YY_N)$, where XX is the rest of the label ("other" excluded) and YY_i are the labels of the sibling nodes of the original node.
    * **otherwise**. It is difficult to apply a single approach to these examples and these patterns need further investigation for translating them into logics properly:

        · NN CC NNS "aimster.com and others"
        · NN NN NN "Seeing Other People"
        · NN NN NN "Examining Other Beliefs"
        · NN CC NN NNS "Buddhism and Other Religions", "Cholesterol and Other Fats"

  - **Containing "related"**. We think that to translate accurately these patterns one needs to have a knowledge base with "related" relation, which will allow to decode "related" concepts. Moreover, some of the example require considering parent labels in the translation procedure to get accurate translation. Examples include:
    - ∗ JJ NNS "Related RFCs", "Related Utilities", "Related Publications", "Related Issues", "Related Movements".
    - ∗ NN CC JJ NNS "Consulting and Related Services", "Haiku and Related Forms", "Food and Related Products"
    - ∗ JJ NN CC NNS "Related Software and Services"
    - ∗ JJ NNS CC NNS "Related Products and Services", "Related Musicians and Places", "Related Firms and Organizations"
    - ∗ NN JJ "Music Related", "Pipeline Related", "Heat Related", "Health Related"
    - ∗ NN IN NN JJ NNS "College of Health Related Professions"
    - ∗ NN CC NN JJ NNS "Contraception and Abortion Related Services"
  - **Containing "its"**:
    - ∗ NN CC NNS NNS "Lightning and its Effects"

- **Common organization names**. There are labels which are organization names, however, with easily understandable semantics. Most of them start with “College of”, “Office of”, “School of”, “Department of”, “Faculty of”. Some examples:

– NN IN JJ NNS “Society of Automotive Engineers”, “School of Visual Arts”

– NN IN NNS “Faculty of Arts”, “School of Humanities”

– NN NN IN NNP “Anglican Church of Mexico”, “Anglican Mission in America”, “Art Academy of Cincinnati”, “Baptist Union of Australia”

Syntax

The analysis enabled us to describe the DMoz labels with the syntax presented in Figure 3.6 using Backus-Naur Form (BNF) notation. The presented BNF grammar covers 310 037 (99.81%) common labels. Rejected patterns constitute only 0.19% of all common labels.

In terms of POS tag patterns, the presented BNF grammar covers 211 (90.95%) of 232 common label patterns. It does not accept 21 (9.05%) patterns. Most of discarded patterns were tagged wrongly.

```
(1) NL_Label:= Phrase {Conn Phrase}
(2) Phrase:= Adjectives [Nouns] | Nouns
(3) Adjectives:= Adjective {Adjective}
(4) Nouns:= Noun {Noun}
(5) Conn:= ConjunctionConn | PrepositionConn
(6) Noun:= NN [POS] | NNS [POS]
(7) Adjective:= JJ | JJR
(8) ConjunctionConn:= CC | ,
(9) PrepositionConn:= IN
```

Figure 3.6: DMoz Labels Syntax.

Constructing a BNF grammar for this dataset and for the following ones allows us to generalize the translation procedure and accept more patterns than we have seen in our analysed samples. Moreover, a comparative analysis of grammars shows the possibility of a grammar unification, which will allow to make the translation procedure more uniform.

Figure 3.7 shows an example parse tree for a DMoz label "Massage Therapy and Body Work" with a respective pattern NN NN CC NN NN.

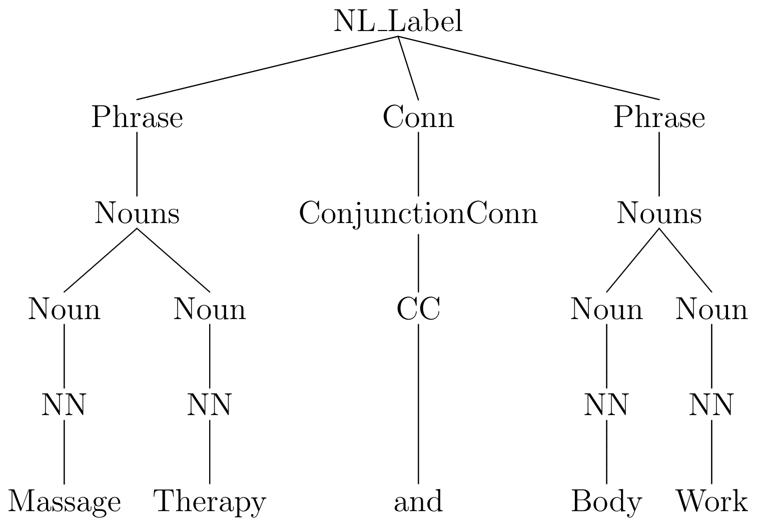


Figure 3.7: Sample DMoz Label Parse Tree.

3.3.2 eCl@ss

```
...
    Communication technology, office -
        Paper, film
            pad (writing, office)
                note paper (other, office)
...
```

Figure 3.8: eCl@ss fragment.

We show the fragment of the eCl@ss dataset in Figure 3.8 to illustrate typical labels of this dataset. To analyze the eCl@ss language patterns we use the following sequence of steps:

1. Tokenize the dataset using OpenNLP tokenizer trained on a manually tokenized random sample from eCl@ss (3591 labels, 24.88%). This tokenizer model tokenizes 92.87% (10x cross-validation) labels correctly, which is higher than the standard OpenNLP model which correctly tokenizes 79.42% of labels.

2. Tag the dataset using OpenNLP POS tagger trained on a manually tagged random sample from eCl@ss (the same sample used for tokenization). This model achieves 97.36% PPT and 90.06% PPL (10x CV), which is higher than the standard OpenNLP model with 66.32% PPT and 18.96% PPL.

3. Collect, sort by frequency and analyze manually the POS tag patterns.

It should be noted that while the POS tagger precision is high enough, it is not 100% and therefore, in the numbers presented here there is a possibility for a small error.

Table 3.10: eCl@ss Labels Lengths Distribution. Top 5 Rows.

Token count	Label count	Share of labels (%)
1	2 512	17.4529
2	3 661	25.4360
3	1 236	08.5875
4	1 260	08.7543
5	1 958	13.6038

Table 3.11: eCl@ss Labels POS Distribution. Top 5 Rows.

POS Tag	Token count	Share of tokens (%)
NN	31 888	56.3890
)	6 204	10.9708
(	5 906	10.4439
JJ	5 421	9.5862
,	3 324	5.8780

Tokenization

The analyzed dataset consists of 14 431 labels, of them 13 369 (92.64%) are unique labels. The token per label distribution is quite even among the major categories, as displayed in Table 3.10 for the top of the distribution, while Table B.1 shows the distribution completely.

POS Tags

Table 3.11 shows the top of the POS tag distribution among labels, while Table B.2 shows the complete version. High shares of round brackets "()" and commas "," indicate highly structured labels. One can notice a very small share, almost absence of proper nouns (NNP, 0.03%). A small share of coordinating conjunctions "CC" indicates that the labels should have a low ambiguity.

Table 3.12: Top 5 eCl@ss POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
2 853	19.82	NN NN	Methyl benzoylformate
2 457	17.07	NN	Acylase
583	4.05	NN NN (NN)	Laboratory app. (repair)
567	3.94	NN NN NN	Block heat exchanger
566	3.93	JJ NN	Exterior radiator

POS Tag Patterns

There are 1 496 patterns covering the complete eCl@ss dataset. The top 20 patterns in eCl@ss cover 64.70% of labels. 645 patterns have more than one label instance. Table 3.12 shows the top 5 eCl@ss patterns with examples, while Table B.3 shows all 20 top patterns.

Qualitative Analysis

Many labels are made of simple forward noun phrases, like “Block heat exchanger” or “flexible equipment wire”. In rare cases the modifiers are placed also after a noun, as well as simultaneously in front and after the noun: “Tallowamide hydrogenated”, “Alcohol C12-13 ethoxylated”, “Sliding vane rotary compressor”, “monolithic thickfilm ceramic sensor”.

Abbreviations is the most notable phenomena encountered in eCl@ss labels. Their almost complete absence in the other analysed datasets makes them even more notable. Abbreviations make 4.02% of all tokens (2 276 cases). In many cases it is possible to expand them by looking for the token which starts like the abbreviation in the labels above the current one along the path to the classification root. Such cases amount to 39.6% of all abbreviations. While a majority of abbreviations are specific to this dataset, one should notice that some of them, such as “w.” for “with” and “f.” for “for” are quite generic and could be encountered in other datasets.

Table 3.13: eCl@ss Abbreviations.

Abbreviation	Expansion (? indicates guessed)	Share of tokens (%)
lab.	laboratory	6.46
w.	? with	3.47
f.	? for	2.64
oth.	? other	0.88
techn.	? technology, technological	0.79

Table 3.13 shows some examples of abbreviations. Expansions marked by question mark were not found in dataset, but rather were suggested by common sense.

While searching for expansion along the path to the root is a first option to check, some abbreviations could be expanded by examining their siblings or by looking at other abbreviation cases from the dataset.

Round brackets, as well as commas, are also widely used in eCl@ss labels (65.37% of patterns), although their semantics is not very consistent. We identify the following types of round brackets use:

- **specification**: "Laboratory app. (repair)", "epoxy resin (transparent)", "lithography (19th century)", "Valve (pneumatic, parts)", "reducing flange (steel, alloyed)", "Screw (with head)";

- parts of **chemical slang**: "(E,E)-Potassium sorbate", "(N,N-Diethyl-3-aminopropyl) trimethoxysilane", "(Methylimino)diethane-1,2-diyl-distearate", "(1S)-(-)-Camphanoyl chloride", "(S)-Malic acid". We note that chemical slang is regular and has precise semantics which can be parsed by a special grammar, however, exploiting this requires recognizing that these labels are indeed chemical and differentiating them from other labels;

- **repetition** of the broader topic from the above levels: "Seal, sealing material (packing material)", "Blank (packing material)", "Box (packing material)";

- **specification and repetition**: "Capsule (gelatine, packing material)", "Beaker (plastic, packing material)", "Pipette (plastic, packing material)".

Often bracketed tokens repeat the label of the level above, but even in these cases the use is not consistent, although the examples of the first of the following two kinds prevail. Compare:

- **Sub-topic (topic)**: "documentation (industrial compact computer)", "software (industrial compact computer)";
- **Topic (sub-topic)**: "industrial compact computer (accessories)", "industrial compact computer (other)"

In a majority of cases, the round brackets are to be found at the end of the label. However, there are a few exceptions, such as: "Bottle (aluminum) larger than 1000 ml", "Can (coex) up to 1000 ml", "Cobalt (II) carbonate", "Diethyl (trimethylsilyl) phosphite", "Rhenium (IV) oxide".

There are many cases were round brackets are repeated. Among these cases the following categories could be identified, with the first category prevailing:

- **specification**: "Reducing piece (high pressure) (non-ferrous metal)", "T-piece (ready) (plast.)", "Pipe (round) (non-ferrous metal)", "Reducer (other) (glass)";
- **specification and repetition**: "cutting grinder (electrical) (household appliance)".

eCl@ss POS tag patterns containing commas constitute a significant (53.81%) portion of patterns. A majority (68.07%) of patterns with commas contains commas used between tokens within brackets. We identify the following semantics of the pieces within brackets separated by commas:

- modifiers, specifying **different kinds of topic outside brackets**: “Threaded flange (iron, steel)”, “cross union (steel, alloyed)”;
- **modifier and repetition**: “Box (aluminum, packing material)”, “Carrier bag (paper, packing material)”, “Gun (steam, parts)”, “fork arm (industrial truck, parts)”;

Commas, used to separate pieces outside of round brackets, differ in their semantics too. We identify the following groups here, with the first group representing most of cases:

- comma for enumeration of **largely independent pieces**: “Nonprint, Multimedia”, “Sound damper, pulsation damper”, “Machine, apparatus”, “Training, schooling”, “Gas cleaning, dedusting plant”, “Cleansing material, cleaning material”, “Spam, Canned meat”, “Fish, seafood, crustacean”;
- comma between **modifiers of a head noun**: “copying, printing line”, “Sparkling, dessert wine”;
- comma between **head noun and modifiers**: “Refrigeration, equipment”, “moistener, Finger Tip”, “Package insert, paper”, “Shelf display, wood”, “window opener, electric”.

Few patterns (1.93%) also use a dash or a backward slash as a syntax tool, mostly to separate alternatives. However, while in some cases a dash or a slash indicate alternative, in others they separate a modifier or specifier. For example:

- **alternative**: “master clock / signal clock”, “account book / journal”, “Dewatering Machine - Expander/Expeller”, “softstarter/ ACregulator", "tribologic dust measurement / filter monitor (PAT)", "controller / card (PC)";

- **modifier or specifier**: "Filter - Activated Carbon", "Heat exchangers - reboiler", "Sterilizer - Compression Still".

Syntax

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

Figure 3.9: eCl@ss Labels Syntax.

The analysis enabled us to describe the eCl@ss labels with the syntax presented in Figure 3.9 using BNF notation. The presented BNF grammar covers 92.70% of labels. It covers 1 009 patterns, which constitute 67.45% of total amount. 487 patterns (32.55%) were discarded by this grammar. We did not extend the grammar to cover these 487 patterns because they have very few label instances. Figure 3.10 shows an example parse tree for the eCl@ss label "Reducer (high pressure)" with a respective pattern NN (JJ NN).

Analysis of the top 163 patterns (with more than 1 instance label) shows the following rejection reasons, which fall into 3 major groups of patterns:

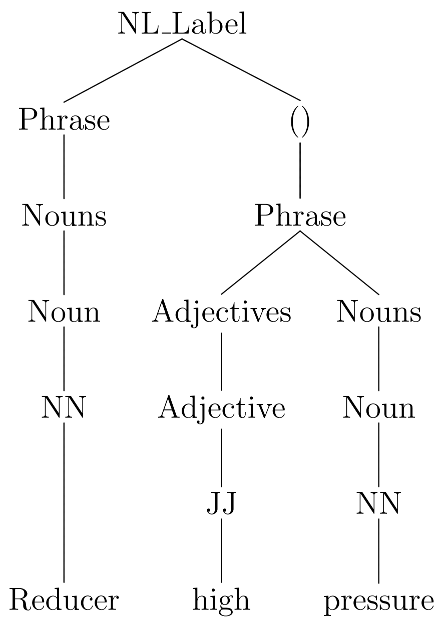


Figure 3.10: Sample eCl@ss Label Parse Tree.

- **POS tagger error** (44.17% of patterns). The tagger tagged one of the tokens incorrectly, thus creating an erroneous and unusual pattern;
- **inconsistent or complex label** (7.9% of patterns). The label is very complex and may include some erroneous syntax elements, like misplaced comma, for example: "Stand. software, (platform-independent)";
- various **BNF limitations** (47.93% of patterns). These limitations mostly arise due to inconsistent or too complex use of syntax or language:
  - **noun used among modifiers** (10.42% of patterns): "Sunflower fatty acid methyl ester". The noun was not recognized as a modifier;
  - **proper noun used as a modifier** (3.06% of patterns): "Kaplan turbine";

– **reverse modifiers** (6.7% of patterns): In these cases modifiers are put after the noun: “Tallowamide hydrogenated”, “Coconut oil alcohol ethoxylated”;

– expressions with **gerund verbs** (9.8% of patterns): In these cases verbs are used as a part of fixed expression or as modifiers: “Pipe (ready to be installed) (oth. mat.)”, “Polymer containing Si-Si chains”;

– **brackets in the middle of the label** (3.06% of patterns); as opposed to appended to the end of the label;

– **colon or slash** used to indicate alternative (3.6% of patterns). “account book / journal”;

– **prepositions with comparative adjectives** used to express complex relation (7.3% of patterns): “Acyclic monocarboxylic acids greater than C8”, “Heterocycles (unsaturated, more than 2 N) (lab)”;

3.3.3 LCSH

```
...
    Adventure and adventurers
        Escapes
            Concentration camp escapes
            Escapes, Fiction
            Prisoner-of-war escapes
...
```

Figure 3.11: LCSH fragment.

We show the fragment of the LCSH dataset in Figure 3.11 to illustrate typical labels of this dataset. To analyze the LCSH language patterns we use the following sequence of steps:

1. Tokenize the dataset using the OpenNLP tokenizer, trained on a manually tokenized random sample from LCSH (44 490 labels, 13.20%). This tokenizer model tokenizes correctly 99.40% (10x cross-validation) of labels, which is higher than the standard OpenNLP model which correctly tokenizes correctly 96.07% of labels.

2. Tag the dataset using the OpenNLP POS tagger, trained on a manually tagged random sample from LCSH (the same sample used for tokenization). This model achieves 99.45% PPT and 92.64% PPL (10x CV), which is higher than the standard OpenNLP model with 72.63% PPT and 27.18% PPL.

3. Collect, sort by frequency and analyze manually the POS tag patterns.

It should be noted that while the POS tagger precision is high enough, it is not 100% and therefore, in the numbers presented here there is a possibility for a small error.

Table 3.14: LCSH Labels Lengths Distribution. Top 5 Rows.

Token count	Label count	Share of labels (%)
1	28 110	08.3727
2	67 678	20.1583
3	48 138	14.3382
4	45 279	13.4866
5	46 800	13.9396

Table 3.15: LCSH Labels POS Distribution. Top 5 Rows.

POS Tag	Token count	Share of tokens (%)
NNP	386 302	26.1166
NN	331 775	22.4302
,	210 808	14.2520
NNS	164 186	11.1001
JJ	129 578	8.7603

Tokenization

The analyzed dataset consists of 335 856 labels, of them 335 809 (99.98%) are unique labels. The token per label distribution is quite even among the major categories, as Table 3.14 shows for the top of the distribution, while Table C.1 shows the complete version.

POS Tags

Table 3.15 shows the top of the POS tag distribution among labels, while Table C.2 shows the complete version. High shares of commas "," and round brackets "()" indicate highly structural labels. One can notice almost equal shares of NNP and NN. These resembles the clear division between proper and common name labels in DMoz, but here the picture is more complicated, as labels have more complex structure.

Table 3.16: Top 5 LCSH POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
22 192	6.61	NNP NN	Teach family
14 444	4.30	NNP NNP (NNP)	Coconucos Range (Colombia)
13 474	4.01	NNP	Myzocallis
11 211	3.34	JJ NN	Negative staining
8 771	2.61	NN NNS	Museum docents

POS Tag Patterns

There are 13 520 patterns covering the complete LCSH set. The top 20 LCSH patterns cover only 44.45% of labels (compare to the top 20 patterns in DMoz covering almost 99% of labels). Table 3.16 shows 5 of the top 20 LCSH patterns, their coverage and examples, while Table C.3 shows all 20 top patterns.

Qualitative Analysis

One noticeable thing about LCSH patterns is that they are highly structured with commas. Commas split patterns into pieces or chunks and while the number of patterns is significant (13 520), the examination of patterns at the chunk level reveals that they form 44 groups. Each group consists of patterns where each piece has the same semantics. We can use the semantics of such chunks during translation. For example, to leave out of the formula the tokens used for the disambiguation of other tokens, once we have finished disambiguation. We identified the following chunk types:

- common noun phrase (NP): “International cooperation”;
- event name (event): “Ashanti War”;
- geographical name (geo): “Tokyo”;
- geographical name with disambiguation (geo-dis): “Tokyo (Japan)”;

Table 3.17: 5 LCSH Chunk Types with Examples.

Chunk-Pattern	POS Tag pattern	Example
event,geo,time	NNP NNP NNP NNP, NNP, NNP, CD	Clydeside Apprentices' Strike, Glasgow, Scotland, 1937
event,time	NNP NNP, CD	Turco-Montenegrin Wars, 1711-1714
event,time,geo	NNP NNP, CD, NNP	World War, 1939-1945, Poland
event,time,NP	NNP NNP, CD, NNS	Sino-Japanese War, 1894-1895, Causes
event,time,NP,geo	NNP NNP, CD, NNS, NNP	Crimean War, 1853-1856, Campaigns, Romania

- time period (time): "1918-1945";
- noun phrase with disambiguation (NP-dis): "Contractions (Topology)";
- domain (domain): "in literature";
- personal name (name): "Constantine I";
- "wildcard": "Handbooks, manuals, etc.";
- "reverse" noun phrase (RNP): "Sculpture, Gothic".

These types combine into 44 groups of patterns. For example, pattern NNP, NN CC NN, CD when considered at the chunk level, consists of three chunks: NNP, NN CC NN and CD. Analysis of the labels with this pattern type reveals that at the chunk level they form a pattern "geo, NP, time". Table 3.17 shows 5 chunk pattern types with POS tag pattern and heading examples, while Table C.4 shows the complete version.

Many of the other phenomena present in LCSH, are also encountered in the previously analyzed datasets, such as:

- **"etc." wildcards**, that is, an expression, starting with few words giving a general direction and ending with the keyword "etc.", such as
  - "Cranes, derricks, etc.",
  - "Associations, institutions, etc.",
  - "Charts, diagrams, etc.",
  - and "Handbooks, manuals, etc.".

  We encounter such wildcards in different combinations with other chunks, such as:
  - combined **with RNP**: "Aesthetics, Religious aspects, Buddhism, [Christianity, etc.]";
  - combined **with NP**: "Obstetrics, Handbooks, manuals, etc.";
  - combined **with a modifier** in reverse order: "Speeches, addresses, etc., Ethiopian";
- **"other" wildcards**, such as illustrated by the label "Christianity and other religions, Greek";
- **round brackets** use for disambiguation, such as:
  - a proper noun disambiguated with another proper noun: "Whitemarsh Hall (Philadelphia, Pa.)";
  - a proper noun disambiguated with a common noun: "Maat (Egyptian deity)";
  - a common noun disambiguated with another common noun "Difference (Philosophy)";
- **domain specification**:

  - **within a chunk**: “Nude in art”, “Calvinism in literature”;
  - **in a separate chunk**: “National characteristics, Belgian, in literature”, “Textile fabrics, Medieval, in art”;

- **abbreviations**: “Otway Basin (Vic. and S. Aust.)”, “Abb’s Valley (Va. and W. Va.)”;

Syntax

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

Figure 3.12: LCSH Labels Syntax.

The analysis enabled us to describe the LCSH labels with the syntax presented in Figure 3.12 using BNF notation. The presented BNF grammar covers 99.45% of headings. It covers 12 585 patterns, which constitute 92.96% of the total amount. 953 patterns (07.04%) were discarded. We did not extend the grammar to cover these patterns, because they have few label instances. Figure 3.13 shows an example parse tree for the LCSH label "Whitemarsh Hall (Philadelphia, Pa.)" with a respective pattern NNP NNP (NNP , NNP).

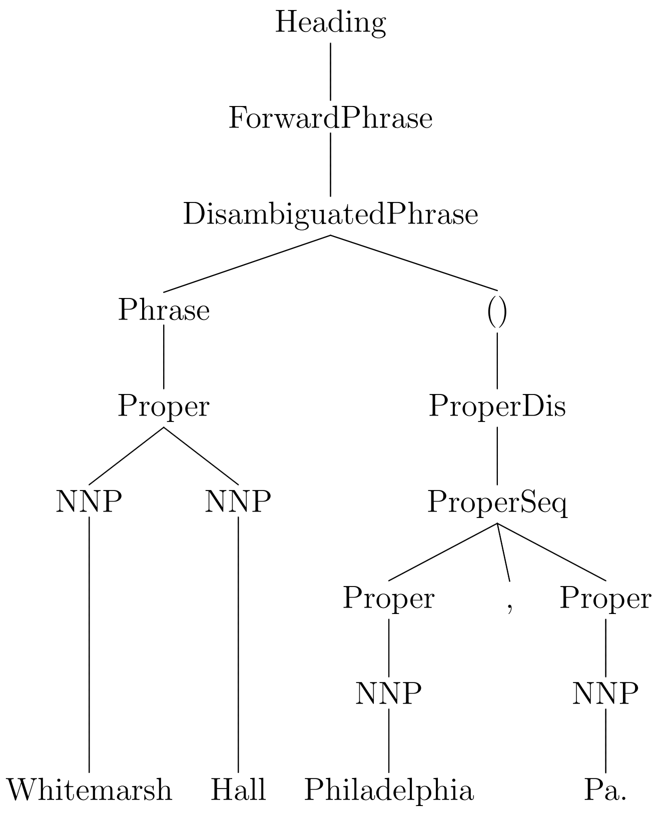


Figure 3.13: Sample LCSH Label Parse Tree.

An analysis of the top 366 patterns shows the following rejection reasons, which fall into 3 major groups of patterns:

- **POS tagger error** (49.59%). The tagger tagged one of the tokens incorrectly, thus creating an erroneous and unusual pattern;
- **inconsistent or complex heading** (2.47%). The heading is complex, includes wildcards or written in a different way than most headings. For example, "English language, Study and teaching (Elementary), Spanish, [German, etc.] speakers". This heading contains a

disambiguation element (Elementary) and a wildcard "Spanish, [German, etc.]". Usually in LCSH wildcards are used at the very end of the heading, and used for head words. This is one of two examples of a wildcard applied to a modifier;

- **BNF limitations** (47.94%). These limitations mostly originate from the inconsistent use of language, that is, some headings are written in slightly different way than most headings. Cases are not exclusive as sometimes several reasons apply:
  - **noun used among modifiers** (17.81%): "Child mental health", "Group medical practice". The noun was not recognized as a modifier;
  - **the article** (6.30%): "Language and the Internet". Article the was not recognized;
  - **disambiguation attachment** (9.59%): "Moses, (Biblical leader), In rabbinical literature". The disambiguation is separated with commas and is not attached to a token, but makes a separate "chunk". "Teniente (Firm) Strike, 1973". The disambiguation is in the middle of the proper noun tokens;
  - **general BNF deficiency** (3.29%). Here are the cases which already should be handled by the BNF, but due to some errors are not handled yet: "Drina River Valley (Bosnia and Hercegovina and Serbia)";
  - **reverse "the"** (3.01%): "State, The, in literature";
  - **proper noun used as a modifier** (2.74%): "Sharp/NNP programmable/JJ calculators/NNS";
  - **hanging modifiers** (2.47%): "Television for the hearing impaired". "hearing impaired" is named as a "hanging" modifier,

because there is no noun which they modify. Intended meaning is “hearing impaired persons”;

- **foreign words** used as modifiers (0.55%): “Foie/FW gras/FW industry/NN”;
- **mixed order** (0.55%): “Civilization, Ancient, in popular culture”. In these cases the “chunk” order is mixed, or it could be considered as a normal reverse order, but specified with a domain “in popular culture”, which should be handled in a special way then;
- remaining cases are **combinations of the above** and occupy each 0.27%;

3.3.4 NALT

```
...
        animal science
            animal production
                replacement rate
                wool production
                animal characteristics
                    animal age
                    animal performance
                        racing performance
...
```

Figure 3.14: NALT fragment.

We show the fragment of the NALT dataset in Figure 3.14 to illustrate typical labels of this dataset. To analyze the NALT language patterns we use the following sequence of steps:

1. Tokenize the dataset using the OpenNLP tokenizer with a standard model. Model trained on a manually tokenized random sample from NALT (13 624 terms out of 43 038, 31.60%). This tokenizer model tokenizes 99.93% (10x cross-validation) of the terms correctly, which is a bit lower than the OpenNLP standard model, which correctly tokenizes 99.96% of labels. We suspect that the custom tokenizer model performance is lower than the standard model performance because the NALT dataset size is smaller than that the one used for training the standard model and there are no enough differences in data for the advantages of retraining to show up.

2. Tag the dataset using the OpenNLP POS tagger, trained on a manually tagged random sample from NALT (the same sample used for tokenization). This model achieves 97.31% PPT and 96.35% PPL

Table 3.18: NALT Labels Lengths Distribution. Top 5 Rows.

Token count	Label count	Share of labels (%)
1	18 246	42.3951
2	20 533	47.7090
3	2 018	04.6889
4	1 776	04.1266
5	271	00.6297

(10x CV), which is higher than the OpenNLP standard model with 58.05% PPT and 40.46% PPL.

3. Collect, sort by frequency and analyze manually the POS tag patterns.

It should be noted that while POS tagger precision is high enough, it is not 100% and therefore, in the numbers presented here there is a possibility for a small error.

Tokenization

The analyzed dataset consists of 43 038 terms, of them 43 038 (100%) are unique labels. Many (90.10%) labels are very simple: 42.40% of them are 1-token labels and 47.71% of them are 2-token labels. 3 and 4-token labels occupy few percents with everything longer than 4 tokens occupying a fraction of a percent, as shown in Table 3.18, while Table D.1 shows the complete picture.

POS Tags

Table 3.19 shows the top of the POS tag distribution among labels, while Table D.2 shows the complete version. A majority of labels are nouns, proper, singular, and plural. They are rarely modified with adjectives. Round brackets are present, but have a very small share of about 1%.

Table 3.19: NALT Labels POS Distribution. Top 5 Rows.

POS Tag	Token count	Share of tokens (%)
NNP	49 181	65.5450
NN	15 470	20.6173
NNS	5 732	7.6392
JJ	3 520	4.6912
)	423	0.5637

Table 3.20: Top 5 NALT POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
13 356	31.03	NNP NNP	Rhode Island
12 325	28.64	NNP	Diachros
3 858	8.96	NN	thyroglobulin
2 651	6.16	NN NN	milk allergy
2 063	4.79	NNS	defoliants

Nevertheless, they are used mostly for disambiguation purposes. There are also a few cases of comma use.

POS Tag Patterns

There are 275 patterns covering the complete set of NALT terms. One should note that 121 of these patterns are encountered only once and many of them are the result of a tagger error, which means that the real amount of patterns is almost two times smaller. The top 20 NALT patterns cover 97% of terms. Table 3.20 shows top 5 NALT patterns with examples, while Table D.3 shows the top 20.

Qualitative Analysis

The label patterns in NALT were analyzed manually and with the help of previously developed BNF grammars of the other datasets. This allows the highlighting of commonalities and differences between the types of labels. We should note that the NALT terms are close enough to the Yahoo labels and single pieces of LCSH headings, as revealed by the grammars. There are only 9 purely proper noun patterns. While NALT terms rarely use proper nouns as modifiers, still there is no such clear separation between common labels and proper labels as in DMoz case. The labels are quite simple, as shown by the amount of patterns (275 in NALT vs 2 021 in Yahoo vs. 975 in DMoz and vs. 13 520 in LCSH).

Proper names constitute almost 60% of terms (the 2 top patterns). However, this figure is largely due to the taxonomic classification being a part of NALT. These proper names (species names) are the essence of this classification.

The NALT terms contain disambiguation tools similar to those used in LCSH and in Yahoo. Namely, round brackets are used as disambiguation tool. They are used to disambiguate both common and proper nouns using as a disambiguation both proper and common nouns.

It should be noted that round brackets use in NALT is not homogeneous. Sometimes they are used for purposes other than disambiguation. Namely, they are used instead of a preposition or they contain a modifier inside, like the following examples illustrate:

- **instead of a preposition**: "training (animals)", "cloning (animals)", "male infertility (plants)";
- **disambiguation**: "fruits (plant anatomy)", "starch digestion (in vivo)", "curing (crops)";
- **modifiers**: "aquariums (public)", "ponding (natural)";
- **specification**: "food packaging (tamper resistant)", "water absorption (by products or materials)".

Another disambiguation tool used in NALT is the preposition "as": "brain as food", "heart as food", "shellfish as food".

Few terms contain commas. However, there is no consistency in comma use. Namely, natural comma use is mixed with LCSH-style comma use (for specifying additional modifiers). Compare

- **natural comma use**: "leisure, recreation and tourism", "oxide, hydroxide, and oxyhydroxide minerals"
- and **use of commas to separate modifiers**: "inhalation toxicity, acute", "feeding, complementary".

The natural comma use prevails, though: 12 vs 6 label. There is also a wildcard example. Consider these terms: "Ictalurid herpes-like viruses", "Cricket paralysis-like viruses". Here "-like" expresses a wildcard.

Syntax

Yahoo (presented in Section 3.3.6) and LCSH (see Section 3.3.3) grammars both suit NALT well. Both proposed grammars cover more than 99% of terms. The LCSH one has a slightly larger coverage and covers 163 patterns (vs 159 by the Yahoo one), which constitute 59.27% of the total amount. 112 patterns (40.73%) were discarded. We did not extend the grammars to cover these patterns, because they have few label instances.

Figure 3.15 shows an example parse tree for the NALT label "valves (equipment)" with a respective pattern NNS (NN) parsed using the LCSH grammar.

All 112 rejected patterns were analyzed and the following major reasons for rejection were identified:

- **POS tagger error** (80.35%). The overwhelming majority of tagger errors were failures to recognize proper names;

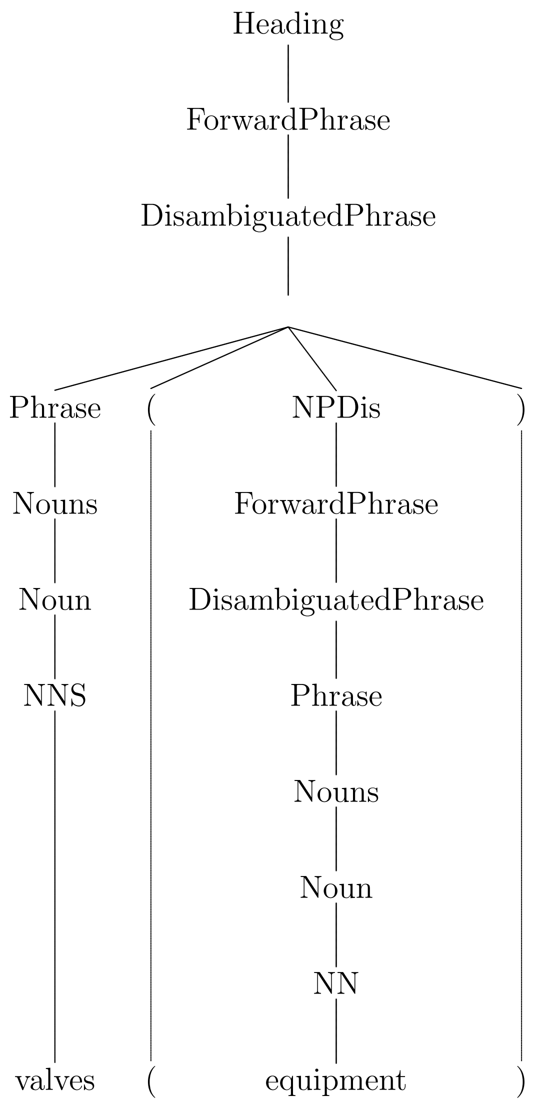


Figure 3.15: Sample NALT Label Parse Tree.

- **noun used as modifier** (8.03%). It seems that modifiers and head nouns should be handled using a different approach. It is difficult to make a decision based exclusively on POS tags;
- **proper noun used as modifier** (3.57%);
- **unusual disambiguation** (1.7%). There are two cases here, one includes double disambiguation: "malate dehydrogenase (oxaloacetate-decarboxylating) (NADP)"; another has disambiguation immediately

after token: “glycogen (starch) synthase”;

- **wildcard**. (1 case). Only one type of wildcard is present in NALT terms. These are few terms like “Ictalurid herpes-like viruses”, “Cricket paralysis-like viruses” where “-like” expresses a wildcard.

3.3.5 UNSPSC

```
...
    Industrial Cleaning Services
        Decontamination services
            Hazardous material decontamination
                Radioactive decontamination services
                Asbestos decontamination or removal
...
```

Figure 3.16: UNSPSC fragment.

We show the fragment of the UNSPSC dataset in Figure 3.16 to illustrate typical labels of this dataset. To analyze the UNSPSC language patterns we use the following sequence of steps:

1. Tokenize the dataset using the OpenNLP tokenizer with a standard model, which gives 100% precision when tested on a manually tokenized random sample from the UNSPSC (5 154 labels, 26.05% of 19 779). The tokenizer model, trained on this annotated sample also gives 100% precision in 10x cross-validation.

2. Tag the dataset using the OpenNLP POS tagger, trained on a manually tagged random sample from the UNSPSC (the same sample used for tokenization). This model achieves 97.56% PPT and 92.32% PPL (10x CV), which is higher than the OpenNLP standard model with 74.71% PPT and 32.89% PPL.

3. Collect, sort by frequency and analyze manually the POS tag patterns.

It should be noted that while POS tagger precision is high enough, it is not 100% and therefore, in the numbers presented here there is a possibility for a small error.

Table 3.21: UNSPSC Labels Lengths Distribution. Top 5 Rows.

Token count	Label count	Share of labels (%)
1	1 932	09.7679
2	6 328	31.9935
3	4 018	20.3145
4	3 191	16.1333
5	1 965	09.9348

Table 3.22: UNSPSC Labels POS Distribution. Top 5 Rows.

POS Tag	Token count	Share of tokens (%)
NN	31569	47.9772
NNS	18659	28.3571
JJ	9350	14.2097
CC	5172	7.8602
IN	655	0.9954

Tokenization

The analyzed dataset consists of 19 779 labels, of them 19 779 (100%) are unique labels. The token per label distribution is quite even among the major categories, as displayed in Table 3.21 for the top of the distribution, while Table E.1 shows the complete version.

POS Tags

Table 3.22 shows the top of the POS tag distribution, while Table E.2 shows the complete version. Notice the very low share of proper nouns. Basically, the labels are composed of the first 4 grammatical categories: nouns (NN, NNS), adjectives (JJ) and coordinating conjunctions (CC). A significant share of coordinating conjunctions together with high shares of 4-token (and longer) labels indicates highly ambiguous labels.

Table 3.23: Top 5 UNSPSC POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
3 347	16.92	NN NNS	Sheet lifters
1 662	8.40	NN NN NNS	Slickline paraffin scrappers
1 511	7.64	NN NN	Play sand
1 046	5.29	NNS	Levels
1 009	5.10	JJ NNS	Brominated retardants

POS Tag Patterns

There are 1 356 patterns covering the complete UNSPSC label set and 786 of them have only one label instance. The pattern distribution is quite even, as opposed to the DMoz case. The top 20 patterns in DMoz cover almost 99% of labels, while the top 20 UNSPSC patterns cover only 69.97% of labels. Table 3.23 shows the top 5 UNSPSC patterns with examples, while Table E.3 shows the complete version.

Qualitative Analysis

The UNSPSC labels do not use syntactic tools like commas or brackets for structural purposes. Most labels are quite simple descriptive phrases. The labels contain a significant amount of coordinating conjunctions. 950 patterns (out of 1 356) contain at least one conjunction. Consider, for example, "Manufacturing Components and Supplies" and "Seating parts or accessories".

One should note the labels containing potential wildcards. Wildcards in UNSPSC are identified by specific words, rather than by POS tag pattern alone. Such words include:

- "accessories",
- "supplies",

- “components”,
- “subsystems”,
- “parts”,
- “the like”.

The labels containing such words need special processing during the translation procedure. Consider the following labels: “Bulletin boards or accessories”, “Seating parts or accessories”, “Waste containers and accessories”. In these cases the word “accessories” is modified by the whole preceding chunk. In some cases wildcards are combined in one label: “Resuscitator components or accessories”.

In general, labels containing “-like” or “the like” need a special knowledge base or ontology to expand “like” correctly: “Machinery for working wood and stone and ceramic and the like”. However, this is an outlier, being the only example in this dataset.

There are labels containing tokens almost explicitly specifying set relations, like “other than” and “including” or “not including”. Consider the following examples containing:

- **“other than”**: “Taxes other than income tax”, “Fabrics of vegetable material other than cotton”, “Residues other than animal feed”;
- **“including” or “not including”**: “Point of sale materials not including printed materials”, “Chemicals including Bio Chemicals and Gas Materials”, “Sales marketing agencies including print”.

There are many cases where a would be adjective is missing a dash, showing a deviation from a more common writing with dash or in a single word: “Adjustable pre set capacitors”, “Pre ignition knock sensor”, “Pre school educational services”. To recognize some of these cases one might first check in the dictionary the presence of the alternative spelling.

Many labels contain acronyms without any syntactic tool being used to indicate them. This is contrary to the common practice of putting acronym in round brackets. Acronyms are derived from single nouns as well as from their plural forms. Acronyms usually follow the tokens they represent. These cases should be recognized and handled properly. Consider the following examples:

- **acronym follows** the tokens and their initial letters: “Light emitting diodes LEDs”, “Central processing unit CPU processors”, “Personal computer PC application design”;
- **acronym does not correspond completely** to the initial letters of the tokens: “Osmium Os”;
- **acronym contains initial letters of word components**: “Infrared IR sensors”, “Polyvinyl Chloride PVC”, “Polypropylene PP”, “Polyethersulfone PES”, “Slow continuous ultrafiltration SCUF units or related products”;
- **acronym follows later**, not immediately after the abbreviated tokens: “Light rail vehicle transport LRV services”;
- **acronym does not correspond** to the letters or word components: “Acrylonitrile butadiene NBR”, “Electrocardiography EKG units and related products”;
- **acronym precedes** abbreviated tokens: “VPN virtual private network managed network services”, “ATM asynchronous transfer mode managed network services”.

Acronym introduction cases should not be mistaken with cases where the acronym is simply used: “Programming for HTML”, “ERP or database applications programming services”.

```
 (1) NL_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 | ProperName
 (7) Adjective:= JJ | JJR | CD | VBG
 (8) ConjunctionConn:= CC | ,
 (9) PrepositionConn:= IN
(10) ProperName:= NNP {NNP}
```

Figure 3.17: UNSPSC Labels Syntax.

One needs to mentions many labels, containing tokens "for the physically challenged". The semantics of these labels could be difficult to translate in propositional description logics exactly. Consider the examples: "Gardening tools for the physically challenged", "Independent living aids for the physically challenged". For example, the first label speaks about a category of gardening tools, a specially modified ones, which is not indicated in the label directly, but implied by the presence of "for the physically challenged".

Syntax

The analysis enabled us to describe the UNSPSC labels with the syntax presented in Figure 3.17 using BNF notation. The presented grammar covers 90.42% of labels. It covers 1 024 patterns, which constitute 75.52% of total amount. 332 patterns (24.48%) were discarded. Figure 3.18 shows an example parse tree for the UNSPSC label "Seismic magnetic systems" with a respective pattern JJ JJ NNS.

The discarded patterns analysis shows that noun modifiers remain ma-

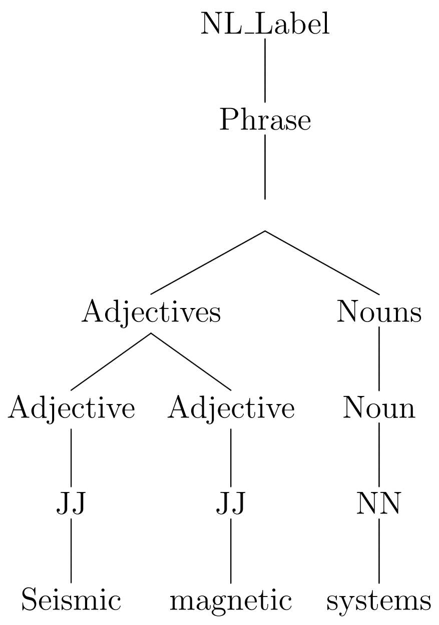


Figure 3.18: Sample UNSPSC Label Parse Tree.

jor rejection reason. Examples of these cases include: “Photo sensitive transistors” with pattern “NN JJ NNS”, “Feng shui instructional materials” with pattern “NN NN JJ NNS”. In these cases there is a noun used among modifiers, which gets confused with noun as a head of the label.

3.3.6 Yahoo! Directory

```
...
    Government
        U.S. Government
            Politics
                Interest Groups
                    Political Action Committees (PACs)
                        MoveOn.org
...
```

Figure 3.19: Yahoo! Directory fragment.

We show the fragment of the Yahoo! Directory dataset in Figure 3.19 to illustrate typical labels of this dataset. To analyze the Yahoo! Directory language patterns we use the following sequence of steps:

1. Tokenize the dataset using the OpenNLP tokenizer, trained on a manually tokenized random sample from the Yahoo (132 350 labels of 829 081, 15.9%). This tokenizer model tokenizes 99.88% (10x cross-validation) labels correctly, which is higher than the OpenNLP standard model, which correctly tokenizes 99.77% of labels.

2. Tag the dataset using the OpenNLP POS tagger trained on a manually tagged random sample from the Yahoo (the same sample used for tokenization). This model achieves 98.14% PPT and 97.90% PPL (10x CV), which is higher than the OpenNLP standard model with 61.67% PPT and 47.44% PPL.

3. Collect, sort by frequency and analyze manually the POS tag patterns.

It should be noted that while POS tagger precision is high enough, it is not 100% and therefore, in the numbers presented here there is a possibility for a small error.

Table 3.24: Yahoo Labels Lengths Distribution. Top 5 Rows.

Token count	Label count	Share of labels (%)
1	432 092	52.1170
2	141 905	17.1159
3	206 726	24.9344
4	25 050	03.0214
5	5 722	00.6902

Table 3.25: Yahoo Labels POS Distribution. Top 5 Rows.

POS Tag	Token count	Share of tokens (%)
NN	610235	38.5370
NNS	338313	21.3648
NNP	270046	17.0537
CC	188653	11.9136
JJ	113685	7.1793

Tokenization

The analyzed dataset consists of 829 081 labels, of them 96 626 (11.65%) are unique labels. More than a half of labels are extremely simple 1-token labels and majority of labels does not exceed 3 tokens, as displayed in Table 3.24 for the top of the distribution, while Table F.1 shows the complete version.

POS Tags

Table 3.25 shows the top of the POS tag distribution among labels, while Table F.2 shows the complete version. Majority of labels are nouns, singular, plural and proper. They are rarely modified with adjectives. Round brackets and commas are present, but have very small share of less that 1%. Nevertheless, they are used in some labels for structural purposes (commas) and disambiguation (brackets).

Table 3.26: Top 5 Yahoo POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
211 753	25.54	NN	Slowpitch
136 156	16.42	NNS	Sidecars
84 762	10.22	NN CC NN	Support and Assistance
52 316	6.31	NNP	Hitwise
38 395	4.63	JJ NN	High Jump

POS Tag Patterns

There are 2 021 patterns covering complete set of Yahoo labels. One should note that 1 190 of these patterns have only one instance label and many of them are result of a tagger error, which means that real amount of patterns is smaller. The pattern distribution resembles that of the DMoz case, while having a bit longer “tail” of 18%. The top 20 Yahoo patterns cover almost 94% of all labels. Table 3.26 shows top 5 Yahoo patterns with examples, while Table F.3 shows the complete version.

Qualitative Analysis

The label patterns in the Yahoo were analyzed manually and with the help of BNF grammars, developed for the previous datasets. This allows to highlight commonalities and differences between the types of labels. We should note that Yahoo labels, while being closer in structure to DMoz labels, do not have such clear separation between proper and common labels. There are only 27 purely proper noun patterns. Yahoo labels use proper nouns as modifiers more frequently in comparison with DMoz. The labels are more complex than in DMoz and less complex than in LCSH, as shown also by the amount of patterns (2 021 in Yahoo vs. 975 in DMoz and vs. 13 520 in LCSH). The analysis of syntax elements usage shows less consistency in Yahoo labels, compared to both DMoz and LCSH.

The comma use in Yahoo labels is much closer to the one in DMoz ("natural" use of comma to separate modifiers), rather than the one in LCSH (for structural purposes, to separate facets or pieces of a heading). Out of the 239 analyzed cases of patterns with commas from the Yahoo directory, only 34 preserved semantics if comma-separated parts were processed independently. The remaining patterns contained 70 erroneously tagged patterns and 132 patterns where basically comma separated modifiers of the head noun, thus making the comma-separated parts dependent and required processing as a single piece. Example of the independent comma-separated parts: "Rocks, Gems, and Minerals". The example of the dependent comma-separated parts: "Classic, Exotic, and Sports Cars".

The Yahoo labels contain disambiguation tools similar to those used in LCSH. Namely, round brackets are used as disambiguation tool. However, the round brackets are also used to indicate point in time or period of time. This could also be seen as a disambiguation. However, this usage is less consistent compared to LCSH, because sometimes time is indicated in brackets (like disambiguation), while there are also cases when time is indicated in front of the label, in the first token or at the end of the label, in the last token. The dash is another syntactic tool used for disambiguation in Yahoo labels. Examples:

- **Period of time** in brackets: "Artaud, Antonin (1896-1948)".
- **Disambiguation** in brackets: "Taloyoak (Spence Bay)".
- **Time in front** of a label, in the first token: "2008 U.S. Presidential Election Issues".
- **Time at the end** of a label, in the last token: "Hurricane Flossie 2007".
- **Dash as a disambiguation** tool: "2002 World Masters Games –

Melbourne".

- **Brackets and dash as a disambiguation** tool: "Paul, Ron (R) – 14th District".

Among Yahoo labels we should note the following types of labels, which somehow stand out:

- **hanging modifiers**, like "Oriental and European". Hanging modifiers are adjectives, which usually modify some noun higher in the tree structure;
- **facet labels**, like "By Instrument", "By Month", "By Genre and Subject", "By Province or Territory". These usually structure labels lying below them into facets;
- **verbalized facet labels**, like "Browse by Country", "Browse By Region";
- **proper noun labels** with date disambiguation, mostly movie names, like "Iron Man (2008)";
- **introduced abbreviations** in round brackets, for example "Internet Service Providers (ISPs)";
- **combined facets**, like "2004 – Athens". These labels combine time and space;
- **double disambiguation**, like "Days of Glory (Indigenes) (2006)";
- **periods of time** specified using dash, dash and spaces, open periods of time, birth dates, imprecise periods:
  - dash: "Clare of Assisi (1194-1253)";
  - dash and spaces: "Bacon, Ernst (1898 - 1990)";

  - open periods of time: “Gell-Mann, Murray (1929-)”;
  - birth dates: “Oehlen, Albert (b. 1954);
  - imprecise periods: “Le Prince, Louis (1842 - 1890?)”;
  - early history periods of time: “Sophocles (496-406 BCE)”;
- special, **directory-specific patterns**:
  - “2000: Bush – Gore”. This pattern is used for all presidential elections mentioned in the directory.
  - political affiliation: “Huckabee, Mike (R)”;
  - political affiliation and district: “Pelosi, Nancy (D) – 8th District”;
- **etc-wildcards**: “MUDs, MUSHes, MOOs, etc.”, “MUDs, MUSHs, Etc.”;
- **square brackets** instead of round brackets for disambiguation: “Danville [Knox County]”;
- **reversed titles**: “Lesson Before Dying, A”, “Machinist, The”;
- **disambiguation brackets**:
  - after token, in the middle of the phrase: “Little Blue (Fairy) Penguins”;
  - after the phrase: “XO (One Laptop Per Child Project)”, “Zoloft (Sertraline)”;
- **semicolon** as disambiguation or abbreviation expansion tool: “TCAP: The California Arts Project”;
- **dash as combiner**: “NATO – Russia Relations”. One should note the spaces surrounding the dash;

- **duel names**: “Lewis vs. Tyson”;
- **expanded abbreviations**: “POP (Post Office Protocol)”;
- **paired proper names**: “PowerShot S50 and S45”, “Polish, Mark and Michael”, “Maysles, Albert and David”, “Hughes, Albert and Allen”;

Syntax

```
 (1) ForwardPhrase:= [VB] [IN] DisPhrase {Conn } DisPhrase
 (2) DisPhrase:= Phrase ["(" ProperDis | NounDis ")"]
                       ["(" TimePeriod ")"] [":" Phrase]
 (3) Phrase:=[DT] Adjectives [Nouns] | [ProperName] Nouns
 (4) Adjectives:= Adjective|CD {[CC] Adjective}
 (5) Nouns:= Noun {Noun}
 (6) Conn:= ConjunctionConn | PrepositionConn
 (7) Noun:= NN [POS] | NNS [POS]
 (8) Adjective:= JJ
 (9) ConjunctionConn:= CC | ","
(10) PrepositionConn:= IN | TO
(11) ProperName:= NNP {NNP|POS}
(12) NounDis:= TimePeriod|Nouns|Adjectives [Nouns]
(13) ProperDis:= ProperSeq [CC ProperSeq]
(14) TimePeriod:= [NN] CD ["-"] [CD] [NN]
(15) ProperSeq:= ProperName ["," ProperName]
```

Figure 3.20: Yahoo Labels Syntax.

The analysis enabled us to describe the Yahoo labels with the syntax presented in Figure 3.20 using BNF notation. The presented grammar covers 99.46% of labels. It covers 1 320 patterns, which constitute 65.31% of total amount. 701 patterns (34.69%) were discarded.

Figure 3.21 shows an example parse tree for the Yahoo label "Artaud, Antonin (1896-1948)" with a respective pattern NNP NNP NNP (CD).

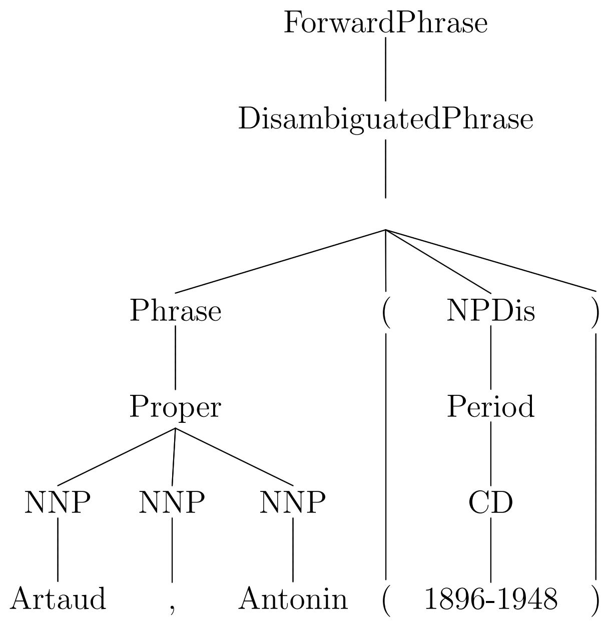


Figure 3.21: Sample Yahoo Label Parse Tree.

Top 213 rejected patterns (those with more than 1 instance) were analyzed and the following major reasons for rejection were identified:

- **POS tagger error** (70.9%). The overwhelming majority of tagger errors were failures to recognize proper names;
- **noun used as modifier** (10.3%). It seems that modifiers and head nouns should be handled on a higher level. It is difficult to make a decision based exclusively on POS tags;
- **unrecognized date** (5.6%). This is a deficiency of a current BNF;
- **disambiguation immediately after token** (3.7%). Usually disambiguation follows the label, however there are also these exceptions;
- **proper noun used as modifier** (0.9%);

- **etc-wildcard**. Only one type of wildcard is present in Yahoo labels. These are labels like “MUDs, MUSHes, MOOs, etc.” where “etc.” expresses a wildcard;

Chapter 4

Metadata Processing Architecture

4.1 Overview

Natural language metadata, being a subset of natural language, is ambiguous and hard to reason about (see Chapter 3. These problems of ambiguity and complexity need to be addressed to enable metadata use in semantic applications, such as the ones we introduced in Section 1.1 and review in Chapter 7. One of the approaches to this problem, described in [34], is to translate the natural language metadata into a propositional Description Logic language L^C to reason about sets of information items (for example documents) precisely described by L^C formulas.

We overview the typical processing steps that a target application needs to perform by considering the example from [79]: "Bank and personal details of George Bush". We identify several key steps of the translation process and highlight processing problems, as well as illustrate them with additional examples in Table 4.1.

We consider *atomic concepts* as the basic building blocks of the L^C formulas. Any controlled vocabulary containing word senses (for example, WordNet [18]) can provide such atomic concepts. A sense of a word usually represents an atomic concept. However, a concept can be lexicalized as more than one word, as often dictionaries contain multiword expressions

Table 4.1: Input Examples.

#	Label	Comments
1	Packing material	basic label
2	Multi-media service center	“multimedia” is usually written together
3	Automation, electrical-engineering, PLT	“electrical engineering” is more common
4	Electrical Cable and Accessories	label with the conjunction “and”
5	George Bush	simple NE label
6	Might and Magic Games	contains NE “Might and Magic” with a common noun “games”
7	Brain and Computer Science	double multiword example: the token “science” is in both “brain science” and “computer science”
8	Accurate Accounting and Timely Data Entry	unwanted multiword example: “accounting entry”
9	Haiku and Related Forms	wildcard example: notice “related”
10	Economics, Examinations, questions, etc.	wildcard example: notice “etc.”
11	Mug’s game	idiom or ambiguous proper noun: a small company name, which sells hand-painted mugs or a movie name “A Mug’s Game” by David Blair

which nevertheless describe a single concept. Take for example “computer science” and “Mount Fuji”. Atomic concepts can be roughly divided into two large groups: common nouns and adjectives, and proper nouns, also known as named entities (NEs).

First, we identify potential atomic concepts in the label. In our example, 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 the controlled vocabulary. Examples 2 and 3 in Table 4.1 illustrate more difficult atomic concept recognition cases, complicated by the tokenization problems. Examples 5 and 6 shows complications brought by the need to recognize named entities. Examples 7 and 8 demonstrate the obstacles presented by the multiword expressions, showing the cases where they should and should not be recognized. Examples 9 and 10 highlight a problem of wildcards or special constructions, which not only need not to be recognized as atomic concepts, but further treated specially.

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 ($\sqcup$). Examples 4, 7, 8 and 9 show the labels with such a conjunctions, while the conjunction in the example 6 should be treated as a part of a named entity. Examples 9 and 10 show the wildcard examples, which require special constructions in a target language, rather than just being treated as a complex concepts.

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”. Examples 3, 4, 7 and 8 show issues represented by more complex coordinations.

As a result we have the L^C formula which represents the concept and unambiguously describes 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.

4.2 NLP Pipeline

We propose a pipeline architecture, each module of which addresses closely related problems of a translation step. Figure 4.1 displays the proposed architecture with the modules and the connections between them. Description of each module follows in the sections below.

4.2.1 User Input

We introduce some optional dialog boxes that allow the pipeline to be used in two modes: fully automatic and user-assisted. The latter is introduced as a solution to inferior performance of some difficult processing steps, such as word sense disambiguation. It allows the user to introduce corrections into the decisions made by the pipeline. We propose to combine error-correcting tasks in a special Semantic Text Input Interface shown in Figure 4.2.

Thie dialogue with this interface appears in the beginning of the pipeline to collect the user input and at the end of processing for the error correction. In the first appearance the user types in the label and launches processing by making no input during a predefined period of time. In the second appearance the user can correct the results before the final translation into the logical formula.

In the automatic processing mode the dialogs do not appear and a calling program interacts with the pipeline in a traditional way, using an API. The pipeline API allows to manipulate and correct the results of the automatic processing. This might be needed in case the application encounters some specific and frequently repeating patterns in the language of its domain, which are not worth separate adaptation, but which need to be corrected

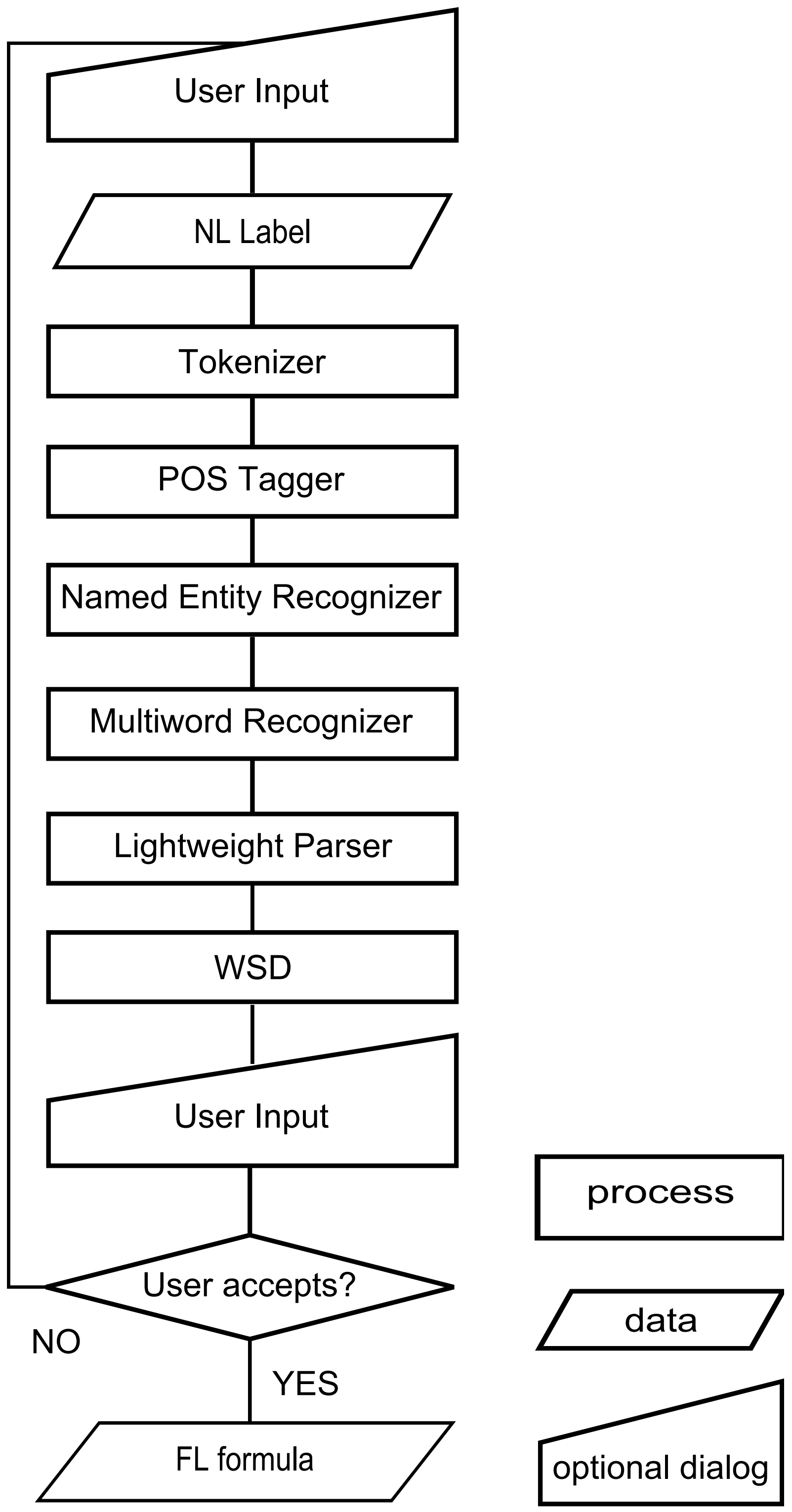
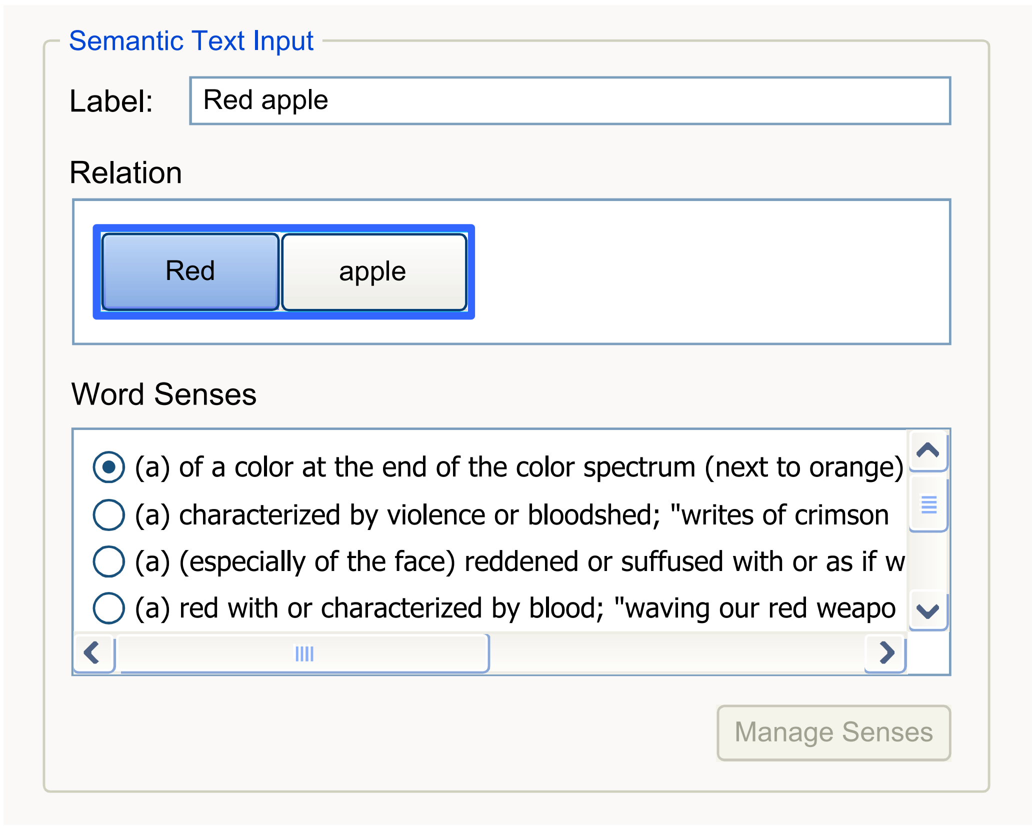


Figure 4.1: Natural Language Metadata Processing Pipeline.


Semantic Text Input

Label: Red apple

Relation

Word Senses

- [✓ (a) of a color at the end of the color spectrum (next to orange)]
- (a) characterized by violence or bloodshed; "writes of crimson
- (a) (especially of the face) reddened or suffused with or as if w
- (a) red with or characterized by blood; "waving our red weapo

Figure 4.2: Semantic Text Input Interface.

to achieve better translation results.

4.2.2 Tokenization

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 and the analysis provided in Chapter 3 helped to reveal the details. Some difficulties arise from a standard, but very frequent use of dot for abbreviations, such as described for eCl@ss in Section 3.3.2 or a frequent use commas combined with non-standard word order, as noted in Section 3.3.3. Other difficulties arise from various non-standard use of such punctuation elements as commas, round

Table 4.2: Desirable Tokenizer Output Examples.

#	Label	Desirable Tokenization
1	Packing material	packing|material
2	Multi-media service center	Multi-media|service|center
3	Automation, electrical-engineering, PLT	Automation|,|electrical|-|engineering|,|PLT
4	Electrical Cable and Accessories	Electrical|Cable|and|Accessories
5	George Bush	George|Bush
6	Might and Magic Games	Might|and|Magic|Games
7	Brain and Computer Science	Brain|and|Computer|Science
8	Accurate Accounting and Timely Data Entry	Accurate|Accounting|and|Timely|Data|Entry
9	Haiku and Related Forms	Haiku|and|Related|Forms
10	Economics, Examinations, questions, etc.	Economics|,|Examinations|,|questions|,|etc.|
11	Mug’s game	Mug|’s|game

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 example eCl@ss label “Hand tools (maint.,service)”. 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. Table 4.2 shows further examples of desirable tokenizer output on our examples.

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 4.4 summarizes the results of the experiments.

We report the results using precision per token (PPT) and precision per label (PPL) measure for our datasets. Namely, we count the percentage of correctly tokenized tokens (PPT) and the percentage of correctly tokenized labels (PPL). 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 LCSH. One can also notice that the model trained only on this particularly difficult datasets also outperforms the standard OpenNLP model. In the case of eCl@ss we notice lower performance than in other cases. This is caused by the particularly unorthodox use of punctuation in combination with a relatively small size of this dataset.

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

We performed incremental training to explore the stability of the models obtained. For clarity, we report here the first 50 000 tokens of two major datasets in Figure 4.3, with a notice that the rest shows similar trend. Namely, the performance tends to stabilize and reach a plateau, except one case of eCl@ss, where the performance fluctuates around 90%. We report the results of the incremental training for other datasets in the

Table 4.3: Tokenizer Performance, Precision Per Token (%).

Model	DMoz	eCl@ss	LCSH	NALT	UNSPSC	Yahoo
DMoz	**99.91**	53.27	76.24	98.01	100.00	97.77
eCl@ss	99.49	**91.28**	97.32	99.97	99.99	99.06
LCSH	99.87	90.41	**99.71**	99.87	100.00	99.74
NALT	97.63	67.24	82.49	**100.00**	100.00	96.97
UNSPSC	94.39	43.04	36.58	97.58	**100.00**	92.99
Yahoo	99.81	54.59	87.91	98.32	100.00	**99.87**
OpenNLP	*99.79*	*79.42*	*97.20*	*99.94*	*100.00*	*99.68*
combined	*99.89*	*91.23*	*99.34*	*100.00*	*100.00*	*99.87*

Table 4.4: 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**

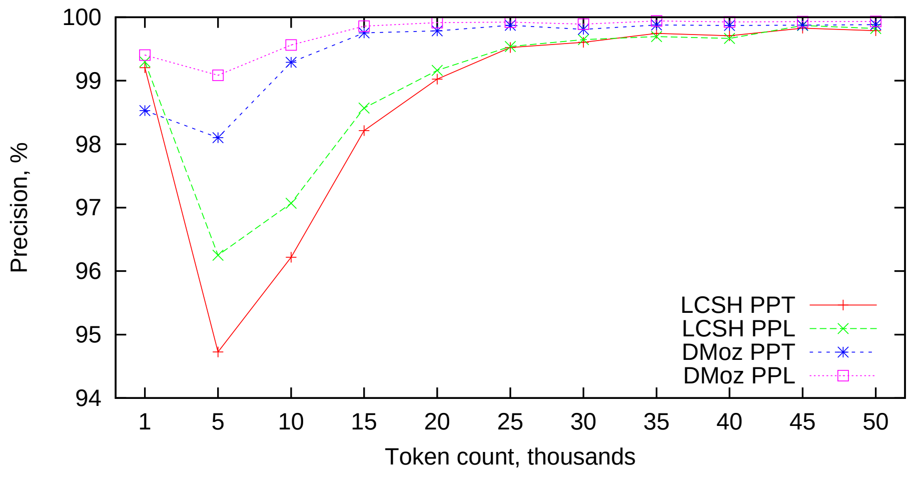


Figure 4.3: Tokenizer Incremental Training for LCSH and DMoz.

Appendix G.

4.2.3 POS Tagging

Problems

Most state of the art POS tagging algorithms are based on supervised learning approaches. To determine a part of speech of 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 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, as we notice in Section 3.3 (see Table 3.1).

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 as we have observed during the analyses presented in Section 3.3. For example, in thesauri the capitalization rule is often

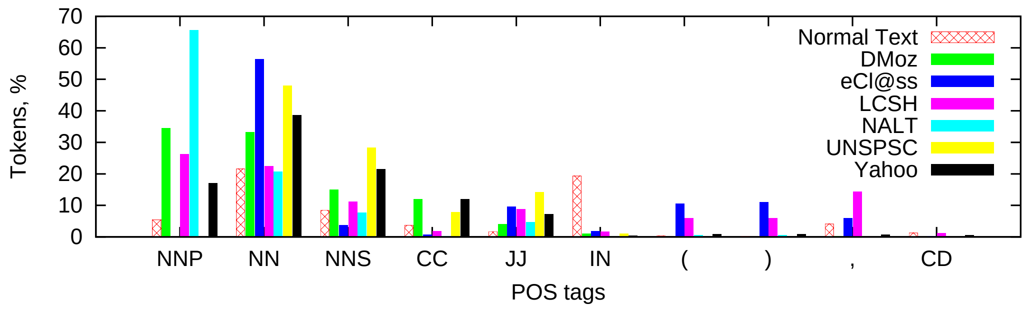


Figure 4.4: Distributions of POS Tags for Normal Text and Metadata.

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. Figure 4.4 shows the distribution of POS tags in normal text and in all our metadata datasets.

Table 4.5 shows examples of desirable POS tagger output.

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 [5, 55].

We performed experiments with the standard OpenNLP models, with a 10-fold cross-validation on each of the datasets, and tested some com-

Table 4.5: Desirable POS Tagger Output Examples.

#	Label	Desirable POS Tags
1	Packing material	packing/NN material/NN
2	Multi-media service center	Multi-media/NN service/NN center/NN
3	Automation, electrical-engineering, PLT	Automation/NN ,/, electrical/JJ -/: engineering/NN ,/, PLT/NN
4	Electrical Cable and Accessories	Electrical/JJ Cable/NN and/CC Accessories/NNS
5	George Bush	George/NNP Bush/NNP
6	Might and Magic Games	Might/NNP and/NNP Magic/NNP Games/NNS
7	Brain and Computer Science	Brain/NN and/CC Computer/NN Science/NN
8	Accurate Accounting and Timely Data Entry	Accurate/JJ Accounting/NN and/CC Timely/JJ Data/NNS Entry/NN
9	Haiku and Related Forms	Haiku/NN and/CC Related/JJ Forms/NNS
10	Economics, Examinations, questions, etc.	Economics/NN ,/, Examinations/NN ,/, questions/NN ,/, etc./FW
11	Mug's game	Mug/NNP 's/NNP game/NNP

bined models. We report the results for the major datasets in a similar way to Table 4.4, using a precision per token (PPT) measure in Table 4.6 and a precision per label (PPL) measure in Table 4.7. 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 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

Table 4.6: 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

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 4.7, we can notice that the data confirms the trend reported by Table 4.6 with even more drastic performance improvements ranging in *all* 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

Table 4.7: 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**

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 here the first 50 000 tokens of two major datasets in Figure 4.5, 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 [79]. We report the results of the incremental training for other datasets in the Appendix H.

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)

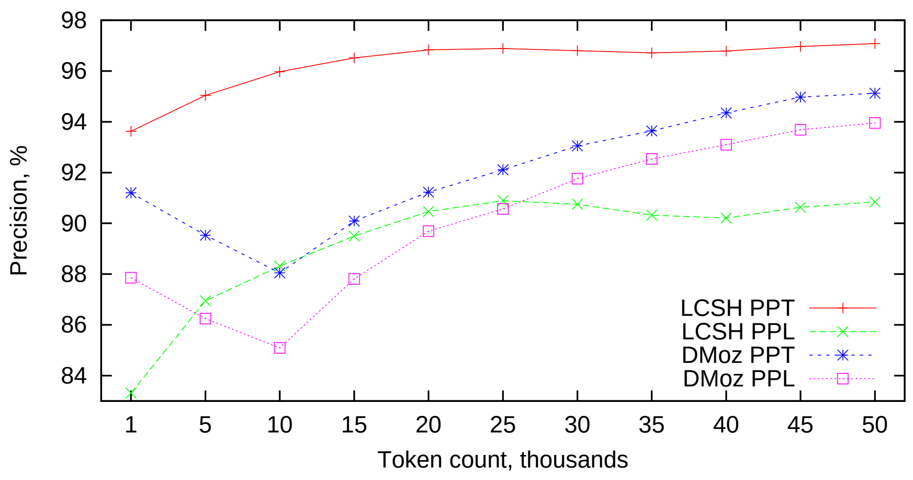


Figure 4.5: POS Tagger Incremental Training for LCSH and DMoz.

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.

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.

4.2.4 Named Entity Recognition

Problems

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

- we need to identify them, because they require different processing than common nouns;
- we need to classify them, because different classes of NEs require dif-

ferent processing;

- we need to disambiguate them.

As for tokenization and POS tagging, our analysis presented in Section 3.3, reveals that 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 4.8 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 dataset (LCSH), 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, presented in Section 3.3, allowed us to see that the assumption made in [79] about labels

Table 4.8: Named Entities Characteristics.

Dataset	% of labels with		NEs	of them (%)			
	NEs	NEs only	Count	LOC	ORG	PERS	MISC
DMoz	36.29	34.80	10 244	75.82	9.21	7.34	7.63
eCl@ss	0.39	0.00	14	0.00	0.00	100.00	0.00
LCSH	37.55	0.75	24 836	79.51	12.46	2.15	5.89
NALT	1.75	1.59	242	92.15	5.37	0.41	2.07
UNSPSC	0.06	0.00	3	0.00	33.33	66.67	0.00
Yahoo	17.97	16.68	24 668	67.54	15.56	8.27	8.63

being either named entity 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 case mixed labels reach 98.01%. 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 4.9 shows examples of desirable NE recognizer output.

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 attain comparable performances. We have chosen the NER algorithm from OpenNLP and we also report the performance of the Stanford NER algorithm [19].

As our annotation and classification scheme we used the CONLL shared task [59] classes of NEs. We identify three major named entity types LOCation, ORGanization, PERSon and a fourth “catch-all” type MISCella-

Table 4.9: Desirable NE recognizer Output Examples.

#	Label	Desirable NER Tags
1	Packing material	packing/O material/O
2	Multi-media service center	Multi-media/O service/O center/O
3	Automation, electrical-engineering, PLT	Automation/O ,/O electrical/O -/O engineering/O ,/O PLT/O
4	Electrical Cable and Accessories	Electrical/O Cable/O and/O Accessories/O
5	George Bush	George/B-PERS Bush/I-PERS
6	Might and Magic Games	Might/B-MISC and/I-MISC Magic/I-MISC Games/O
7	Brain and Computer Science	Brain/O and/O Computer/O Science/O
8	Accurate Accounting and Timely Data Entry	Accurate/O Accounting/O and/O Timely/O Data/O Entry/O
9	Haiku and Related Forms	Haiku/O and/O Related/O Forms/O
10	Economics, Examinations, questions, etc.	Economics/O ,/O Examinations/O ,/O questions/O ,/O etc./O
11	Mug's game	Mug/B-ORG 's/I-ORG game/I-ORG

neous.

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 4.10 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 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% [14]. 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 [79].

Table 4.10: NE Recognizer Performance, F-Measure (%).

Model	DMoz	LCSH	NALT	Yahoo
OpenNLP	11.76	41.68	17.10	9.30
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
combined	**56.56**	**63.38**	**37.60**	**50.78**

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.

4.2.5 Multiword Recognition

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.

We are interested in certain kinds of multiwords only, which we note below while exploring their types:

- lexicalized phrases ("have at least partially idiosyncratic syntax or semantics, or contain words which do not occur in isolation"):
  - fixed expressions, which cannot be internally modified and should be recognized as is (such as "ad hoc", "absolute zero", various idioms);
  - semi-fixed expressions:
    - non-decomposable idioms (with regard to semantic decomposability), that is multiwords whose meaning does not derive from the meaning of the distinct words that appear in them. This type of multiwords must be recognized. This type, however, can undergo some small degree of change like changing the case or becoming plural ("seventh heaven", "cloven foot" $\rightarrow$ "cloven feet", "dark horse" $\rightarrow$ "the darkest horse");
    - compound nominals, which can be internally modified (for example, "abstract art" $\rightarrow$ "abstract visual art"; "at first" $\rightarrow$ "at very first"), or externally modified ("car park" $\rightarrow$ "(car park)s", "attorney general" $\rightarrow$ "(attorney general)s"), whose meaning can be (roughly) approximated as a function of the meanings of the words appearing in the multiword ("academic requirement", "yellow gurnard"). It may be useful NOT to recognize multiwords of this category and consider each multiword's word by itself as it may improve matching results with labels which include the same multiword or expressions with the same meaning, for example, the multiword "academic requirement" can be matched with the "academic prerequisite". However, there is no simple way of detecting such multiwords;

        * proper names, which can be considered multiword, because they often contain several tokens, but should be recognized as a single concept. However, this category deserves and is treated in a separate NE recognizer module;
    - syntactically flexible expressions:
        * verb-particle constructions ("write up", "look up"). We have little interest in verbs, because verbs are rare in our target domain;
        * decomposable idioms ("sweep under the rug"). Exploration of the examples of multiwords of this kind shows that decomposable idioms tend to be verb-based, therefore of less interest to us;
        * light verbs ("make a mistake", and not "do a mistake"). Again, this category is verb-based and we have little interest in them;
- institutionalized phrases, such as "traffic light". These are syntactically and semantically compositional, but occur with markedly high frequency. These phrases are of interest to us.

In recognizing multiwords we face several problems [11], such as reduced syntactic and semantic transparency, recognizing fixed and non-modifiable expressions versus semi-fixed expressions, as well as other expression types [58]. Let us list the problems that are of particular interest to us:

- differentiating idioms from the other fixed expressions types. Idioms should be searched only literally "as is", without any modifications. That means there can be no other words between idiomatic words. For instance, the idiom "nut case" should not be recognized in the label "steel nut and aluminium case".

- recognizing syntactically flexible multiwords, such as the ones allowing plural form. On one hand, in the label “brains and computer science” we might lemmatize the tokens and use “brain|and|computer|science” sequence of tokens to create candidate expressions, which might lead to incorrect recognition. On the other hand, the labels such as “the darkest horse” need lemmatization for a multiword to be recognized.

- peculiarities of tokenization. Consider three ways of writing: “trade off” vs “trade-off” vs “tradeoff”. Often dictionaries providing multiwords contains only certain type of writing such “multiwords”. For example, WordNet [18] does not contain the first option, but does contain second and third.

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. For this category of multiwords the recognition is feasible, although somewhat complicated by the fact that many dictionaries do not provide information about expression type and in particular the syntactic flexibility of the expression. Therefore we focus on solving the following common problems associated with the multiwords present in the dictionary. These include:

- disentangling them in case several of them are present simultaneously

Table 4.11: Desirable MWE recognizer Output Examples.

#	Label	Desirable MWE Output
1	Packing material	packing material
2	Multi-media service center	Multi-media service center
3	Automation, electrical-engineering, PLT	Automation, electrical-engineering, PLT
4	Electrical Cable and Accessories	Electrical Cable and Accessories
5	George Bush	George_Bush
6	Might and Magic Games	Might_and_Magic Games
7	Brain and Computer Science	Brain_Science and Computer_Science
8	Accurate Accounting and Timely Data Entry	Accurate Accounting and Timely Data Entry
9	Haiku and Related Forms	Haiku and Related Forms
10	Economics, Examinations, questions, etc.	Economics, Examinations, questions, etc.
11	Mug's game	Mug_'s_game

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".

Table 4.11 shows examples of desirable multiword recognizer output.

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,

- presence of coordinating conjunctions,
- plural form of some of multiword tokens,
- possible multiplication of tokens, because of simultaneous presence of several multiwords in a label.

For example, we recognize both multiwords from WordNet [18] 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. Recognizing these two multiwords allows exploiting their precise meaning assigned by a human expert.

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, 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.

We evaluate this heuristic as an integral part of the translation task in the overall evaluation.

4.2.6 Lightweight Parsing

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 (see Table 3.1). 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 datasets 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 semantics of label pieces and any information which allows to derive logical connectives between the label parts, such as POS tag patterns, the syntactic structure of the label or dependencies the label pieces.

For example, knowing that the tokens in round brackets disambiguate the preceding tokens – as we note in Section 3.3.3 for 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" or "A-Z" often do (see examples for DMoz in Section 3.3.1), 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.

Let us list some of the common problems we encounter in our datasets, that we can solve by knowing a label structure:

- coordination disambiguation. Knowing that the labels such as "Examples and Use Cases" with the pattern "NNS CC NN NNS" disambiguate in a certain way allows building a correct formula. Of course, pattern-based coordination disambiguation has its limits, however, for the domain with a simple language structure, such as ours, this might be sufficient, especially given the amount of labels with a complex structure.

- identifying and extracting facets. The labels like "By Country" almost always introduce a structural pattern in the classification known as facet. These labels have a limited amount of patterns and can be effectively identified by a POS tag pattern.

- recognizing "hanging" modifiers in the label. Some labels are written in a backward fashion. In such cases the modifiers follow the noun they modify, conversely to the usual case of preceding modifiers. Compare the backward-fashion label "Proverbs, Ladino" with a traditional writing "Ladino Proverbs". Such cases occupy a limited set of POS tag patterns and can be efficiently recognized.

- recognizing "hanging" modifiers in the classification structure. The classifications we studied sometimes contain a label, usually a noun, on a level N and its modifiers one level lower: $N + 1$. Consider the following simple example classification:

  - Proverbs
    - Ladino
    - Italian
    - German

  Although recognizing such pattern requires considering preceding patterns located higher in the classification hierarchy, the patterns, triggering such behaviour constitute a limited set and could be identified.

- recognizing and processing labels with complex semantics, or "wildcards". Natural language has several tools for defining a set of objects using similarity. For instance, by providing several examples and following with a keyword, such as "etc.", "and others", "and similar". A typical example from the LCSH dataset: "Handbooks, manuals, etc."

- direct set manipulation. There are label instances which refine a set objects they define by using specific structure and keywords, similarly to the "wildcard" case. Often such refinement relies on sibling labels. For instance, the label "Other Products" defines a set of objects by relying on its siblings. As previously, such cases occupy a limited set of patterns and keywords and can be recognized, although for proper processing of such cases it might be necessary to access classification structure and sibling labels.

- recognizing advanced syntax tools, such as use of round brackets. Round brackets is one of the most frequently used syntax tools for the classifications. We identified several purposes for which round brackets are used, such as specification, disambiguation, repetition and preposition substitution. Some of them can be distinguished by a POS tag pattern, and in other cases recognizing the semantics requires accessing preceding labels. However, in all cases having a recognized structure and elements inside and outside the brackets allows disambiguating among few cases and processing the label more precisely.

- recognizing domain specifications. Some labels specify the domain of interest by using specific and recognizable patterns. Consider the examples: "Nude in art" and "Calvinism in literature".

Solution

We introduce a lightweight parser which makes the proposed solution more universal through the possibility of implementing different semantic actions and allows to use 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:

Table 4.12: Metadata Language Structure Characteristics.

Dataset	POS Tag Patterns	90% Coverage
DMoz	975	9
eCl@ss	1 496	360
LCSH	13 342	1 007
NALT	275	10
UNSPSC	1 356	182
Yahoo	2 021	15

NNP	NNP	(	NNP	)
Coconucos	Range	(	Colombia	)

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 4.12 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.

Rule-based parsers use manually created rules to encode the syntactic structure of the language. These rules are then applied to the input text to produce parse trees. In long texts parsing, these have been disregarded because of two main disadvantages: they require a lot of manual work to produce linguistic rules and they have difficulties achieving a “broad coverage” and robustness to unseen data. To tackle these problems, state of the art statistical parsers, such as [12], infer grammar from an annotated corpus of text. However, this approach requires a large annotated corpus of text and a complicated process for tuning the model parameters. Moreover, producing a corpus annotated with parse trees is a much more costly and difficult operation than doing a basic annotation, such as POS tagging.

However, the analysis, presented in Section 3.3, shows that the language used in natural language metadata (NLM) is limited to descriptive phrases, introduced in Section 3.1. Hence, we need a limited coverage, which simplifies the construction of the rules. Therefore we use a simpler approach and manually construct a grammar for parsing. This requires having only an accurate POS tagging and some structural information of the language, which are provided by the analysis we presented in Section 3.3. We use a basic descriptive phrase grammar, presented in Section 3.1, as a starting point for our grammars. Analyzing the POS tag patterns we modify this grammar to include the peculiarities of the descriptive phrases, such as combinations of noun phrases, or the use of commas and round brackets for disambiguation and specification, as illustrated by the examples in Chapter 3.

We developed a set of lightweight grammars covering each of our datasets. The grammars we constructed can be divided into two categories: "simple" ones with nine and ten rules (DMoz, eCl@ss and UNSPSC) and a "complex" ones with fifteen and seventeen rules (Yahoo, NALT and LCSH). Table 4.13 provides details about the grammar coverage.

One can note that in all cases we have a high coverage of the dataset labels, more than 90% in all cases and more than 99% in four cases. While the coverage is high, it does not reach 100%, as this is not possible with the flexibility of natural language. This opens the following possibilities for the pipeline to process that small percentage of labels which are not covered by the grammar:

- in a controlled setting it might behave like a controlled language and

Table 4.13: Metadata Language Grammar Characteristics.

Grammar	Rules	Coverage (%)		Parsing Mistakes (%)	
		Patterns	Labels	POS Tagger	Grammar Rules
DMoz	9	90.95	**99.81**	85.98	11.01
eCl@ss	9	67.45	**92.70**	44.17	47.93
LCSH	17	92.96	**99.45**	49.59	47.94
NALT	15	59.27	**99.05**	80.35	13.30
UNSPSC	10	70.58	**90.42**	25.01	65.70
Yahoo	15	65.31	**99.46**	70.90	20.50

refuse to accept a label that does not conform to the grammar, asking the user to edit it;

- rejected labels could be processed by a simpler heuristic, such as a simple “bag of words” approach, put into a log file for a later editing and conversion, or even discarded.

If we look at the pattern coverage we notice a slightly different picture. For NALT, Yahoo, eCl@ss and UNSPSC, we have only 60% to 70% coverage of the patterns. This can be explained by Table 4.12 where, for instance, only around 1% of the patterns already cover 90% of the labels in NALT. This shows how a small amount of the labels uses a large variety of language construction while most of the NLM uses highly repetitive constructs.

Our analysis shows that the main reason for the lower coverage is a less regular use of language in these four datasets as compared to the other two datasets. We have analysed the mistakes done by the parser and found that they mostly fall into two major categories: POS tagger errors and linguistic rules limitations, as shown in Table 4.13. This can be explained by the rule-based nature of our parser that makes it particularly sensitive to POS tagger errors. Other parser mistakes are due to the inconsistent (ungrammatical) or unusually complex labels, which could be seen as "outliers". For example, the "English language, Study and teaching (Elementary), Spanish, [German, etc.] speakers" label from LCSH contains both a disambiguation element "(Elementary)" and a "wildcard" construction "[German, etc.]".

Figure 4.6 shows two examples out of the grammars we produced for the LCSH and UNSPSC datasets. We use BNF for representing the grammar rules. The LCSH one starts with a top production rule `Heading`, which encodes the fact that LCSH headings are built of chunks of noun phrases, which we call `FwdPhrase`. In turn, a `FwdPhrase` may contain two phrases `DisPhrase` with disambiguation elements as in the example above. The disambiguation element may be a proper noun phrase (`ProperDis`) or a common noun phrase (`NounDis`), surrounded by round brackets. `NounDis` is usually a period of time or a type of object, like "Fictitious character" in "Rumplemayer, Fenton (Fictitious character)" while `ProperDis` is usually a sequence of geographical named entities, like "Philadelphia, Pa." in "Whitemarsh Hall (Philadelphia, Pa.)".

The core of the grammar is the `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, it could be a noun(s) modified by a proper noun, or a sequence of foreign words.

A comparative analysis of the grammars of different classifications shows that they all share the nine base rules with some minor variations. Compare the rules 4-12 of LCSH with the rules 2-10 of UNSPSC in Figure 4.6. These nine rules encode the basic noun phrase. Building on top of that, the grammars encode the differences in syntactic rules used in different classifications for disambiguation and structural purposes. For example, in LCSH, a proper noun in a disambiguation element is often further dis-

```
 1 Heading:=FwdPhrase {"," FwdPhrase}

 2 FwdPhrase:=DisPhrase
             {Conn} DisPhrase
 3 DisPhrase:=Phrase {"("ProperDis
                      | NounDis")"}
 4 Phrase:=[DT] Adjs [Nouns] |
          [Proper] Nouns | Foreigns
 5 Adjs:=Adj {[CC] Adj}
 6 Nouns:=Noun {Noun}
 7 Conn:=ConjConn | PrepConn
 8 Noun:=NN [POS] | NNS [POS] |
        Period
 9 Adj:=JJ | JJR
10 ConjConn:=CC
11 PrepConn:=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}
```

```
 1 Label:=Phrase {Conn (Phrase
           | PP$ Label)}

 2 Phrase:=Adjs [Nouns] | Nouns

 3 Adjs:=Adj {Adj}
 4 Nouns:=Noun {Noun}
 5 Conn:=ConjConn | PrepConn
 6 Noun:=NN [POS] | NNS [POS] |
        DT RB JJ | Proper
 7 Adj:=JJ | JJR | CD | VBG
 8 ConjConn:=CC | ,
 9 PrepConn:=IN | TO
10 Proper:=NNP {NNP}
```

Figure 4.6: LCSH (right) and UNSPSC (left) BNF production rules.

ambiguated with its type, as “Mountain” in: “Nittany Mountain (Pa. : Mountain)”.

Although very similar to one another, there are a few obstacles that need to be addressed before these grammars can be united into a single one. One of the most difficult of these obstacles is the semantically different use of round brackets: mostly, round brackets are used as a disambiguation tool, as illustrated by the examples mentioned above; however, we also found some examples where round brackets are used as a specification tool, as for instance in the label from eCl@ss: "epoxy resin (transparent)".

Due to these different semantics, these cases will almost certainly require different processing for a target application. For example, in translating metadata for semantic matching purposes [34], we need to translate the labels of a classification into formulas in the propositional Description Logic language L^C. In this application, the disambiguation element "(Pa. : Mountain)" of the label "Nittany Mountain (Pa. : Mountain)" can be used to choose a precise concept "Nittany Mountain" and the element itself is not included in the final formula, while in the specification case of "epoxy resin (transparent)", the specifier concept "transparent" should be included in the formula in a conjunction with a concept "epoxy resin" that is being specified.

Another obstacle is the different semantics of commas. Sometimes, a comma is used to indicate a sequence of phrases. However, there are cases where the comma separates a modifier in a phrase, written in a backward manner, such as illustrated above with a label "Proverbs, Ladino". In long texts, these differences can be disambiguated by the context, which is almost always missing for natural language metadata.

Despite these differences, our results show that simple and easily customizable grammars can be used to parse accurately most of the patterns found in the state of the art classifications, thus providing extra understanding of the NL without a loss in performance.

Let us illustrate the parsing process on one example of DMoz label, earlier displayed in Figure 3.7. We have implemented our grammar using JavaCC[1] toolkit. Figure 4.7 shows the sample debug output of the JavaCC-based parser while parsing the label "Massage Therapy and Body Work" with the pattern "NN NN CC NN NN". It starts with the initial rule

[1]`https://javacc.dev.java.net/`

"NL_Label" and here the semantic action code attached to this rule creates a new disjunction element | of the future formula, which corresponds to the "CC" in the middle of our pattern, or to the "Conn" rule in the "NL_Label" starting rule (see rule 1 and 5 in Fig.3.6). Several calls to other rules follow, and the next action is executed in the "Nouns" rule (see rule 4 in Fig.3.6) where the action creates the conjunction & to fill it with the concepts later on. Then, the next rule "Noun" consumes the first token "NN" of our pattern and adds the concept "Massage" to the conjunction. This action does the same with the second "NN" token and the concept "Therapy". The process repeats similarly for the second part of the pattern. Final formula is shown on the last line: "Massage & Therapy | Body & Work".

4.2.7 Word Sense Disambiguation

Problems

Many of our motivating applications operate on concepts, which these applications draw from a dictionary or a controlled vocabulary. While concepts are precise and unambiguous, the unit of a natural language is a word, which is often ambiguous. As we aim to translate natural language into formal concept language, we need to disambiguate ambiguous words into unambiguous concepts.

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 [48].

Speaking of our case of natural language metadata, the traditionally difficult task of word sense disambiguation is further complicated with the following circumstances:

- little context or no context at all. In word sense disambiguation task

```
Call:   NL_Label                                    | create |
  Call:   Phrase                                    |
    Call:   Nouns                                   | create &
      Call:   Noun                                  |
        Consumed token: <"NN" at line 1 column 1>   | add concept Massage
      Return: Noun                                  |
      Call:   Noun                                  |
        Consumed token: <"NN" at line 1 column 4>   | add concept Therapy
      Return: Noun                                  |
    Return: Nouns                                   |
  Return: Phrase                                    |
  Call:   Conn                                      |
    Call:   ConjunctionConn                         |
      Consumed token: <"CC" at line 1 column 7>     |
    Return: ConjunctionConn                         |
  Return: Conn                                      |
  Call:   Phrase                                    |
    Call:   Nouns                                   | create &
      Call:   Noun                                  |
        Consumed token: <"NN" at line 1 column 10>  | add concept Body
      Return: Noun                                  |
      Call:   Noun                                  |
        Consumed token: <"NN" at line 1 column 13>  | add concept Work
      Return: Noun                                  |
    Return: Nouns                                   |
  Return: Phrase                                    |
  Consumed token: <<EOF> at line 1 column 14>       |
Return: NL_Label                                    | Massage&Therapy|
                                                    | Body&Work
```

Figure 4.7: Sample DMoz Label Translation.

the context is indispensable. Even humans have difficulties understanding the word without context. Natural language metadata in many cases contains no context at all (see average label length in Table 3.1) or the amount of context available is limited to two or three words.

- absence of common topic or theme. Traditional approaches sometimes rely on the fact that normal text often has a stable topic or theme for a large enough span for the algorithm to grasp it. Natural language metadata, being a "summary" of a data, contains more topic and theme changes. For example, almost every label in web directory is a change of topic or the very least, its modification.

- hierarchical nature of some natural language metadata. Natural language metadata is sometimes organized into hierarchical structures. While this might help in some cases by allowing the algorithm developer to exploit available higher levels of hierarchy, in many cases semantics of a relation between levels is not formally defined [30] and changes even within a single dataset. This makes the algorithms which try to make use of a hierarchy less reliable in some cases.

- traditional word sense disambiguation algorithms were developed with a model of normal text in mind, which makes it difficult to apply them to a natural language metadata, which differs from normal text both quantitatively and qualitatively, as we have shown in Chapter 3.

Solution

We try to maximally exploit several peculiarities of natural language metadata while designing the word sense disambiguation algorithm:

- small size and somewhat regular structure of natural language metadata give a promise to favor heuristic-based approaches.

- natural language metadata contains a lot of nouns, which makes it promising to exploit the most popular hypernymy or “isA” relation between nouns.
- high performance of our POS tagger models makes it promising to rely on part of speech for disambiguation.

We take the set of heuristics first presented in [79], which reach comparable to the state of the art performance of 66.51% precision and adopt them to our framework. The heuristics use WordNet [18] as a source of senses and as a source of hypernymy relations. The heuristic contain several steps, executed one after another while the word is still ambiguous, that is, has more than one sense. In the following *active sense* means that sense has not been discarded yet and is considered to be a possible sense of the word.

The following steps are executed:

1. POS-based disambiguation. Determine the part of speech tag of the word and if the word has the senses of this POS, preserve them and discard other senses.
2. Hypernymy-hyponymy-based disambiguation of nouns in the label. Find hypernym or hyponym relations between active noun senses of a current word and other words in the label. If found, preserve these senses and discard other senses.
3. Distance-limited hypernymy-hyponymy-based noun disambiguation in the label. Find hypernym or hyponym relations between active noun senses of a current word and other words in the label which are not further than a threshold in the hypernym-hyponym hierarchy of senses from WordNet. If found, preserve the senses within the shortest distance and discard other senses.

4. Hypernymy-based disambiguation of nouns in the hierarchy. If the hierarchy is available for this instance of natural language metadata label, find hypernym relations between active noun senses of a current word and the words in the preceding label. If found, preserve these senses and discard other senses.

5. Distance-limited hypernymy-based disambiguation of nouns in the hierarchy. If the hierarchy is available for this instance of natural language metadata label, find hypernym relations between active noun senses of a current word and other words in the label which are not further than a threshold in the hypernym-hyponym hierarchy of senses from WordNet. If found, preserve the senses within the shortest distance and discard other senses.

6. The most frequent sense disambiguation. Preserve the first noun sense and discard others. If no noun sense is available, preserve first adjective sense and discard others.

The step 1 relies on the high precision of the POS tag information provided by our models and is executed for all parts of speech. Steps 2 through 5 rely on hypernymy relation available in WordNet for nouns and verbs and are applied to nouns. These steps might be also applied to verbs, however, as we have shown in Section 4.2.3, verbs are extremely rare in natural language metadata and we do not consider them. For steps 3 and 5 we use 2 as the threshold value. For steps 4 and 5 only hypernym relation is considered, contrary to the steps 2 and 3, because traditionally hierarchies go from less specific to more specific in labels and, consequently, in the senses of their words. Step 6 relies on the fact that senses in WordNet are ordered according to their frequency in the semantic concordance texts and picking the most frequent sense empirically increases the probability that this sense will be the actual meaning of the token.

The experiments with concept search [28] point out that executing the last step and leaving only one sense per word might be not the optimal choice for some applications. Therefore, we leave the choice between a Word Sense Disambiguation (which leaves one sense per word and executes all steps) or a Word Sense Filtering (which might leave several senses per word by skipping the last step) for the target application.

4.3 Robust NLP Pipeline

4.3.1 Problems

In addition to relying on natural language metadata, some of our motivating applications, such as concept search [28], work with documents which contain normal text. However, our motivating applications are mainly interested in concepts. The concepts are usually represented by noun phrases. Therefore, it might be useful to modify proposed solution to accommodate this additional requirement of the motivating applications.

The proposed solution was developed with the assumption of having a short text labels as its input. In practice, however, the following issues arise:

- rejecting the labels the pipeline is not able to parse is undesirable;
- restricting the user's input to the labels the pipeline is able to parse is prohibitive and leads to restricting the user's expressivity and disrupted workflow because of the need to correct the input;
- current approach of falling back to a “bag of words” parse in some cases leads to degraded performance and leaves out pieces of text which otherwise could be parsed with but little aid.

The first issue is solved by falling back to previously used approaches or using “bag of words” approach. However, the second and third issues

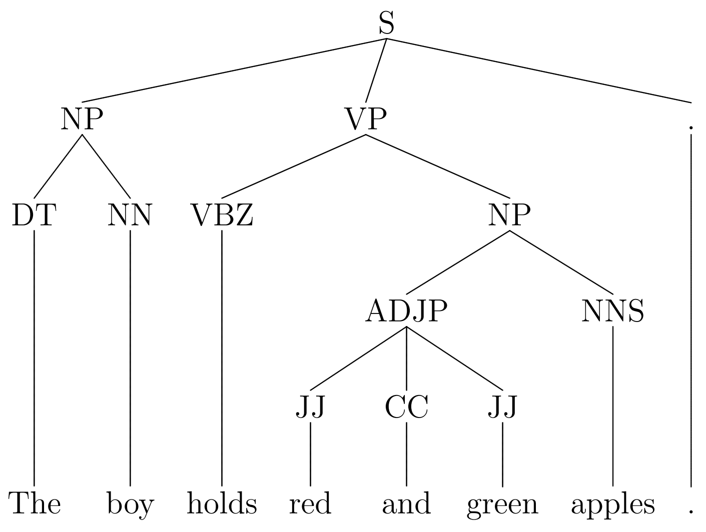


Figure 4.8: Sample Sentence.

require extending the input pipeline "understands" to include a richer subset of language. However, we must be accurate in extending it to avoid raising computational costs and complexity, usually involved with parsing normal text.

Lets consider a typical example of input and concepts of interest in this input for our motivating application. "red and green apples" is a typical short label. The pipeline parses this label into a set of logical formulas expressing complex concepts: {red&apple, green&apple}. However, in many cases these complex concepts are a part of a sentence, such as "The boy holds red and green apples." This sentence contains three concepts which are of interest to our motivating application: {boy, red&apple, green&apple}. The proposed solution, being targeted for short labels, either provides incorrect formulas for such inputs, for example, treating "holds" as a noun, or falls back to a "bag of words", missing correct formulas (for "red and green apples" and "boy") and introducing incorrect ones (for "holds").

Figure 4.8 shows the parse tree for our sample sentence. One can notice that our concepts of interest are all inside noun phrases. Noun phrases and

```
The/B-NP boy/I-NP holds/B-VP red/B-NP and/I-NP green/I-NP apples/I-NP ./O
```

Figure 4.9: Sample Chunker Output.

other types of phrases can be extracted by applying a chunker, a standard NLP component, to a sentence. Also called shallow parsers, chunkers are fairly fast and provide a shallow parse tree of a sentence. Figure 4.9 shows a chunker output for our sample sentence using BIO-notation. Namely, it identifies different chunks, or phrases, such as noun phrases (NP) and verbal phrases (VP) using begin (B-), inside (I-) and outside (O) tags combined with a chunk type.

4.3.2 Solution

We introduce an extension to the proposed solution which allows processing normal text needed by some of our motivating applications. We call this extended pipeline a "robust pipeline". Figure 4.10 shows an updated solution. First the input is processed by the extra modules we have added. They include tokenizer, POS tagger and chunker for normal text (indicated by a "generic" keyword in brackets). Then the added pipeline selector module directs the chunk types our metadata pipeline recognizes, that is NP chunks or noun phrases, for further processing into formulas. Other chunk types remain unprocessed and are discarded. Their processing can be added later if an application will expand the target language to be expressive enough to encode other chunk types, such as verbal phrases, and provide appropriate translation algorithms.

Introduced extension allows reusing standard NLP components with the proposed solution and fulfil the additional requirements of the motivating application with a minimum amount of changes, while preserving the advantages of the proposed solution, such as simplicity and processing speed.

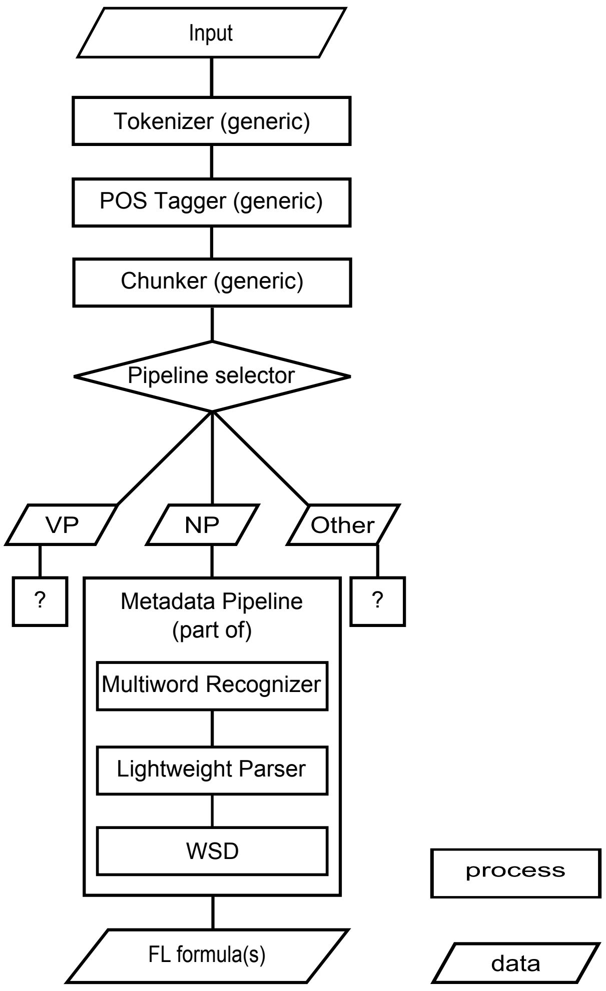


Figure 4.10: Robust Pipeline.

Chapter 5

Word Sense Summarization

5.1 Problem

Our motivating applications operate on concepts, because they are unambiguous, contrary to words of natural language. This introduces a need to establish a correspondence between a word used with a particular meaning in mind and a respective concept. Concepts are usually based on word senses drawn from a dictionary or a controlled vocabulary. Each sense is usually defined by a sentence, called a gloss, which describes the meaning of a sense and often includes examples. Lets consider the example entry from the Collins CoBuild dictionary with the first sense of the word “apple” shown in Figure 5.1. Among other pieces of information it contains the word itself, the gloss and the examples. All this takes a considerable amount of space and requires at least a moment to comprehend.

The considerable complexity of sense definitions is among the factors which make the Word Sense Disambiguation (WSD) task one of the most difficult tasks of Natural Language Processing. While the performance of the algorithm addressing this problem and presented in Section 4.2.7 reaches state-of-the-art levels, this performance is not sufficiently high and user assistance is often required to correct the errors made by the algorithm. We have presented in Chapter 4 in Figure 4.2 the prototype of the Semantic

```
apple
  apples
  1) N-VAR An apple is a round fruit with smooth green, yellow,
  or red skin and firm white flesh.
      > See also Adam's apple, Big Apple, crab apple
      I want an apple.
      ...2kg cooking apples.
      ...his ongoing search for the finest varieties of apple.
      ...a large garden with apple trees in it.
```

Figure 5.1: Sample Word Sense Definition.

Text Input Interface, a user interface which includes the section allowing to correct the disambiguation errors. While having this user interface is helpful, it does not diminish the following problems attributed to the task:

- an ambiguous word has at least two and often more senses, which makes it necessary to evaluate the minimum of two possible choices;
- glosses which define the meaning of word senses contain a minimum of a sentence, often with examples (see Figure 5.1) to illustrate the use of the word in this particular sense. Understanding the gloss itself takes time;
- the sense granularity problem introduces additional difficulties to the sense choice:
  - coarse-grained senses might urge a user to choose the sense which is not precise enough, while
  - fine-grained senses might leave a user with too many choices, leading to a similar loss of precision.

All these problems put the additional cognitive load on a user and take us further away from generating semantics "for free as a by-product of a

Table 5.1: Word Senses with Glosses and Summary Examples.

Concept	Gloss	Sense Summary
apple	fruit with red or yellow or green skin and sweet to tart crisp whitish flesh	fruit
apple	native Eurasian tree widely cultivated in many varieties for its firm rounded edible fruits	tree
java	an island in Indonesia south of Borneo; one of the world's most densely populated regions	island
java	a beverage consisting of an infusion of ground coffee beans; "he ordered a cup of coffee"	beverage

normal computer use".

5.2 Solution

We propose a novel word sense summarization algorithm with a user interface prototype as a solution which helps a user tackle the difficult word sense disambiguation task. A word sense summary is another word, a succinct representation of the meaning of the original word. Consider the examples in Table 5.1. The column "Sense Summary" demonstrates the examples of what a word sense summary can be: a succinct representation of a word sense.

Word sense summary allows a user to:

- check the accuracy of the word sense disambiguation algorithm by looking whether correct sense is chosen and to correct it if necessary or

- estimate the need to introduce a new sense when a controlled vocabulary is available and allows word sense editing.

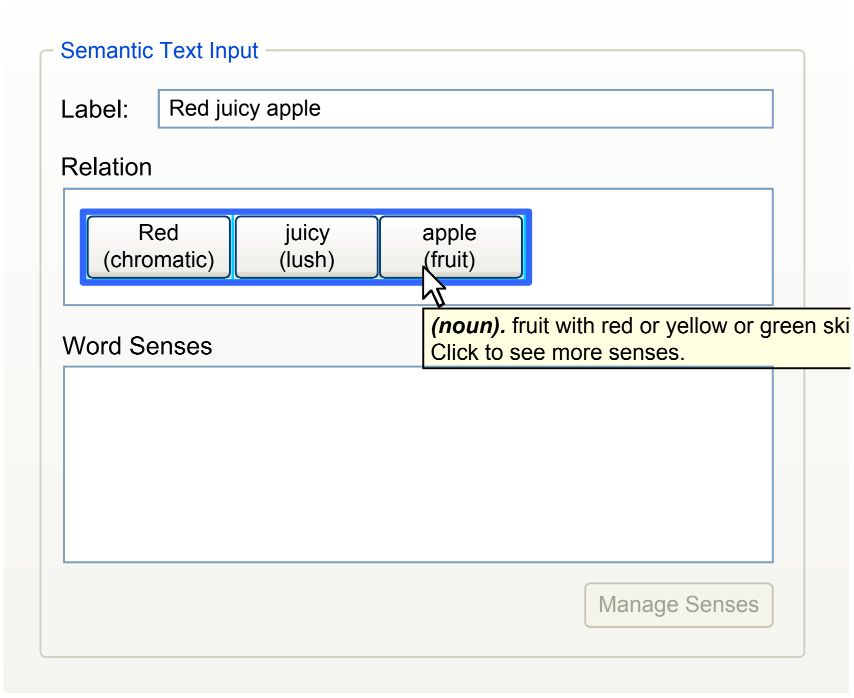


Figure 5.2: WSD Summary in Semantic Text Input Interface.

5.2.1 User Interface Prototype

Figure 5.2 shows a mockup interface, an improved version of the Semantic Text Input Interface displayed in Figure 4.2. In this interface each word, represented by a separate button in the "Relation" section, is accompanied by a word sense summary for the active sense of this word. The summary is shown in the brackets underneath the word. Thus, "chromatic" is the sense summary for the active sense of the word "red", "lush" — for "juicy" and "fruit" — for "apple". Adding the active sense summary allows users to quickly evaluate the accuracy of the word sense disambiguation algorithm and estimate the need of correcting it. It also allows to spot the exact word which needs correcting without having to click on each word to reveal its senses (which will appear in the "Word Senses" section of the interface) and having to read the active sense gloss. In fact, the "Word Senses" section is empty because the user did not click any word button.

5.2.2 Summarization Algorithm

We use WordNet [18] as a source of word senses and we use the linguistic information WordNet provides to generate the summaries for the senses of a particular word. We do the summarization differently for senses of different parts of speech, because WordNet provides different relations for every part of speech and we want to maximally exploit the information available in WordNet. For all parts of speech we consider all available senses of a word and proceed with creating a summary for each of them. The summary should be different for each sense of the word to allow distinguishing among them. We create summaries sequentially, sense by sense, starting with the first sense.

We use the following WordNet elements in the algorithms below when creating the summary for a sense of a word:

- the words in the **synset**;
- the **lemmas** of the word sense;
- the words connected to the sense via different relations such as **hypernymy** and **hyponymy**;
- the sense **gloss**, that is the explanation of the sense meaning with examples;

We call the element “unused” if it was not used yet to create a sense summary for some sense of the word in question.

Noun Sense Summarization

There are 15 776 (13.47% of 117 097) ambiguous nouns having 43 783 senses in WordNet 2.1. To create WSD summary for a sense of a noun we choose sequentially:

- **synset**: the first shortest unused lemma among the available synset words of this sense;
- **hypernym**: the first shortest unused lemma among the available hypernym synset words of this sense;
- **hyponym**: the first shortest unused lemma among the available hyponym synset words of this sense;
- **original**: if there are no hyponym synsets available, return the noun itself (original word).

There are two reasons we rely on the length and choose the shortest among available choices. The first is that often the shortest word is the simplest one. The second is to save screen space. There are few cases where several senses share the same summary (234, or 1.48% of all 15 776 ambiguous nouns). 129 (0.82%) of them have one summary for all senses. Often a summary is shorter than the original word, on average 2.31 characters shorter. In 50.25% of cases (22 001 out of 43 783 senses) the summary produced is longer than the original word, on average 4.84 characters longer. Table 5.2 shows some examples of noun summaries.

Verb Sense Summarization

There are 5 227 (45.49% of 11 488) ambiguous verbs having 18 629 senses in WordNet 2.1. To create a summary for a sense of a verb we choose sequentially:

- **synset**: the first shortest unused lemma among the available synset words of this sense;
- **hypernym**: the first shortest unused lemma among the available hypernym synset words of this sense;

Table 5.2: Noun Word Sense Summary Examples.

Sense	Gloss	Heuristic	Summary
abacus#1	a tablet placed horizontally on top of the capital of a column as an aid in supporting the architrave	hypernyms	tablet
abacus#2	a calculator that performs arithmetic functions by manually sliding counters on rods or in grooves	hypernyms	calculator
circle#1	ellipse in which the two axes are of equal length; a plane curve generated by one point moving at a constant distance from a fixed point; "he calculated the circumference of the circle"	hypernyms	oval
circle#2	an unofficial association of people or groups; "the smart set goes there"; "they were an angry lot"	synset	set
circle#3	something approximating the shape of a circle; "the chairs were arranged in a circle"	hypernyms	form
circle#4	movement once around a course; "he drove an extra lap just for insurance"	synset	lap

- **hyponym**: the first shortest unused lemma among the available hyponym synset words of this sense;

- **original**: the first word of the gloss (often well-known verb, like cause, have, be).

There are few cases where several senses share the same summary (72, or 1.38% of all 5 227 ambiguous verbs). 12 (0.23%) of them have one summary for all senses. In most cases the summary is shorter, on average 1.84 characters shorter than the original word. In 35.09% of cases (6 536 out of 18 629) the summary is a bit longer than the original word, on average 2.59 characters longer. Table 5.3 shows some examples of verb summaries.

Table 5.3: Verb Word Sense Summary Examples.

Sense	Gloss	Heuristic	Summary
abstain#1	refrain from voting	hypernyms	refrain
abstain#2	choose not to consume; "I abstain from alcohol"	synset	desist
accost#1	speak to someone	synset	address
accost#2	approach with an offer of sexual favors; "he was solicited by a prostitute"; "The young man was caught soliciting in the park"	synset	hook

Adjective Sense Summarization

There are 5 252 (23.72% of 22 141) ambiguous adjectives having 14 413 senses in WordNet 2.1. To create a summary for a sense of an adjective we choose sequentially:

- **synset**: the first shortest unused lemma among the available synset words of this sense;
- **similar**: the first shortest unused lemma among the available satellite synsets of this adjective (using similar_to relation);
- **pertainym**: the first shortest unused lemma among the available pertainym synsets;
- **see_also**: the first shortest unused lemma among the available see_also synsets;
- **antonym**: the first shortest unused lemma among the available antonym synsets;
- **gloss**: the first word of the gloss.

There are few cases where several senses share the same summary (19, or 0.36% of all 5 252 ambiguous adjectives). 6 (0.11%) of them have one summary for all senses. In majority of cases the summary is shorter, on average 2.05 characters shorter than the original word. In 43.88% of cases (6 316 out of 14 413 senses) the summary is slightly longer than the original word, on average 2.89 characters longer. Table 5.4 shows some examples of adjective summaries.

Table 5.4: Adjective Word Sense Summary Examples.

Sense	Gloss	Heuristic	Summary
young#1	(used of living things especially persons) in an early period of life or development or growth; "young people"	synset	immature
young#2	(of crops) harvested at an early stage of development; before complete maturity; "new potatoes"; "young corn"	synset	new
young#3	suggestive of youth; vigorous and fresh; "he is young for his age"	synset	youthful
young#4	being in its early stage; "a young industry"; "the day is still young"	gloss	being
young#5	not tried or tested by experience; "unseasoned artillery volunteers"; "still untested in battle"; "an illustrator untried in mural painting"; "a young hand at plowing"	synset	unseasoned

Adverb Sense Summarization

There are 751 (16.32% of 4 601) ambiguous adverbs having 1 870 senses in WordNet 2.1. To create a summary for a sense of an adverb we choose sequentially:

- **synset**: the first shortest unused lemma among the available synset words of this sense;
- **derived**: the first shortest unused lemma among the available derived synsets of this adverb (using derived_from relation);

- **antonym**: the first shortest unused lemma among the available antonym synsets;
- **gloss**: the first word of the gloss.

In 50 (6.66%) of all 751 ambiguous adverbs there are several senses of the same adverb which share the same summary. 18 (2.4%) of them have one summary for all senses. In most cases the summary is shorter, on average 2.56 characters shorter. In 33.74% of cases (631 out of 1 870 senses) the summary is slightly longer than the the original word, on average 3.24 characters longer. Table 5.5 shows some examples of adverb summaries.

Table 5.5: Adverb Word Sense Summary Examples.

Sense	Gloss	Heuristic	Summary
aboard#1	part of a group; “Bill’s been aboard for three years now”	gloss	part
aboard#2	on a ship, train, plane or other vehicle	synset	onboard
aboard#3	on first or second or third base; “Their second homer with Bob Allison aboard”	synset	on base
aboard#4	side by side; “anchored close aboard another ship”	synset	alongside

5.3 Sense Summarization Quality Evaluation

We conduct the evaluation of the word sense summary generation heuristics by creating a series of questions, designed to shed light on the different qualities of the generated summaries and asking these questions randomly to different persons. We asked the members of our research group, colleagues and friends to volunteer for this task. Most questions were answered by students. We have generated the question database and during the question-answer sessions drew questions randomly from different groups of questions in the database until we have collected the minimal required number of answers and have collected enough questions answered by more than one user.

52 users have participated in this evaluation. The users included representatives of both genders, various age groups (between 20 and 60) and cultures. Most users have academic background. While most users are non-native English speakers, they are all fluent speakers and there are native speakers as well as bilingual persons among participants.

15 users answered more than 100 questions each and the top contributor answered 700 questions. 25 users answered at least 40 questions each. 11 users answered less than 20 questions each. On average we collected 83 answers per user.

We have selected a subset of WordNet to generate test questions. Our dataset contains 9 314 summaries. Before starting the evaluation we conducted several tests with few users and almost all users complained about the questions being difficult or very difficult. We found out that the major reason of this is the limited vocabulary of our participants. This is not a surprise, given that many participants are non-native speakers. Native speakers did not report this problem.

To tackle the limited vocabulary issue we have exploited the frequency of use figures from WordNet. The frequency of use is the number of occurrences that the particular sense has in semantic concordance texts. For example, the word “water” (used in the sense “H_2O, substance”) has 744 as its frequency of use number, while the word “hypocrisy” for both of its senses has 1 as its frequency of use number. We have generated the summaries only for the words having non-zero frequency of use. This limits the questions to the most frequently used words and thus, potentially, better known words, resolving the limited vocabulary problem to some extent.

One should note that the applicability of the heuristics to the parts of speech differs because of the nature of the relations available in WordNet:

- the **synset** heuristic applies to all parts of speech;
- the **hypernym** and **hyponym** heuristics apply to both nouns and verbs;
- the **gloss** heuristic applies to verbs and adjectives because of the way glosses are written;
- the **derived** applies to adverbs only;
- the **similar_to** applies to adjectives only.

Table 5.6 and Table 5.7 provide the details of the qualitative composition of the questions dataset.

Table 5.6: Summary Questions by Heuristic.

Heuristic	Summary count	Usage count average
child	1 253	21.1748
derived	125	7.9680
gloss	1 300	13.0808
parent	3 804	24.5739
similar_to	756	6.5847
synset	2 076	11.2418

Table 5.7: Summary Questions by POS.

POS	Summary count	Usage count average
adjective	1 104	5.7101
noun	4 318	26.8256
adverb	271	16.0701
verb	3 621	11.0014

To aid users in the disambiguation task without further complicating the task, the word sense summary should be:

- **associative**: the summary should associate well with the sense it summarizes and
- **discriminative**: the summary should discriminate the senses well enough, so that the senses can not be mixed with one another.

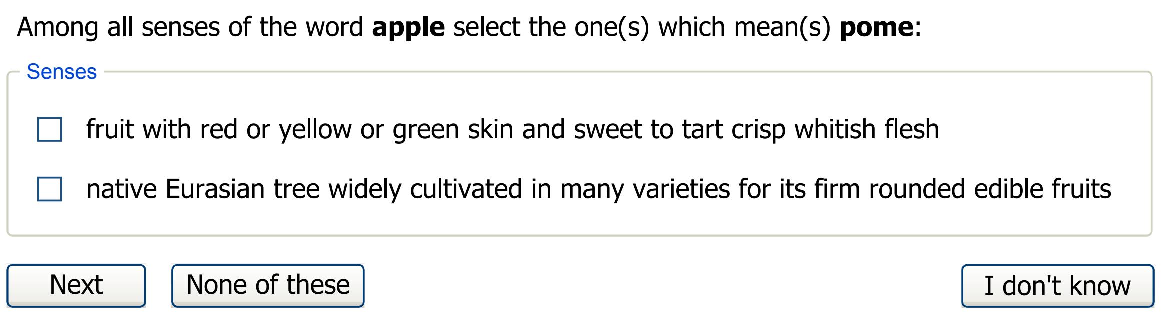
We split the word sense summary generation algorithm evaluation into two scenarios targeted at evaluating the two aspects above.

In the first scenario a user is presented with a question that contains a word, its summary and its senses, expressed by their glosses. The user is asked to select the senses corresponding to the displayed summary. The user has an option to skip the question by clicking the “I don’t know” button, if some words were not clear. This option eliminates the bias introduced by the limitations of the user vocabulary.

Figure 5.3 illustrates the question for the word “apple”. Here the “apple” is the original word which need disambiguation, presented together with all its senses in the box below. We select a sense and memorize our selection. To a user we present all senses and the summary for the memorized sense, as generated by the heuristic being evaluated. The word “pome” is a summary for the memorized sense. Ideally, the summary will help a user to identify the sense which this summary represents and a user will select the check box close to the memorized sense. Then we can compare the actual answer with the memorized sense.

In this scenario we identify the following answer categories:

- **unknown**, when the user clicked “I don’t know” button, which means the user does not know some words present on the screen;
- **none**, when the user clicked “None of these” button, which means user can not associate any sense with the summary;
- **correct**, when the user selected one sense and this sense is the mem-


Among all senses of the word **apple** select the one(s) which mean(s) **pome**:

Senses

☐ fruit with red or yellow or green skin and sweet to tart crisp whitish flesh

☐ native Eurasian tree widely cultivated in many varieties for its firm rounded edible fruits

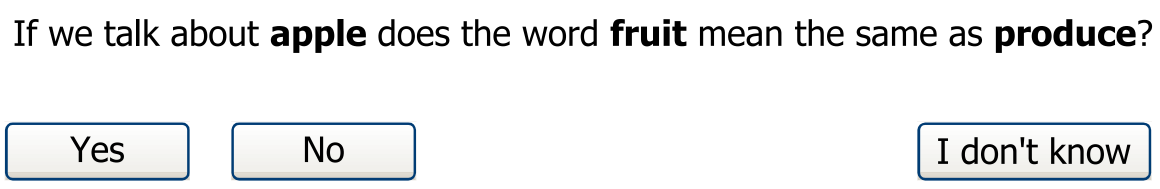
Figure 5.3: WSD Summary Evaluation Scenario 1.

orized one, which means that the summary is good, because the user was able to associate the summary with the sense for which the summary was generated;

- **semicorrect**, when the user selected more than one sense, but the memorized one is among them, which shows that the summary is potentially good, because the user was able to associate it with the memorized sense, and, probably because of too fine-grained senses was not able to make a proper distinction and selected more than one sense. The users agreement on specific questions can be used to determine whether the senses are too fine-grained. Namely, the senses could be considered too fine-grained if the users that answered the same questions agreed and selected the same set of senses;

- **incorrect**, when the user selected 1 incorrect sense, which shows that the summary is potentially bad, because the user was unable to associate the summary with the sense for which it was generated;

- **more than 1 selected sense**, when the user selected more than 1 sense, which shows cases where the senses are too fine-grained and create confusion for the user. These cases could further explored for polysemy reduction purposes using users agreement, as explained in the above point for semicorrect answers.



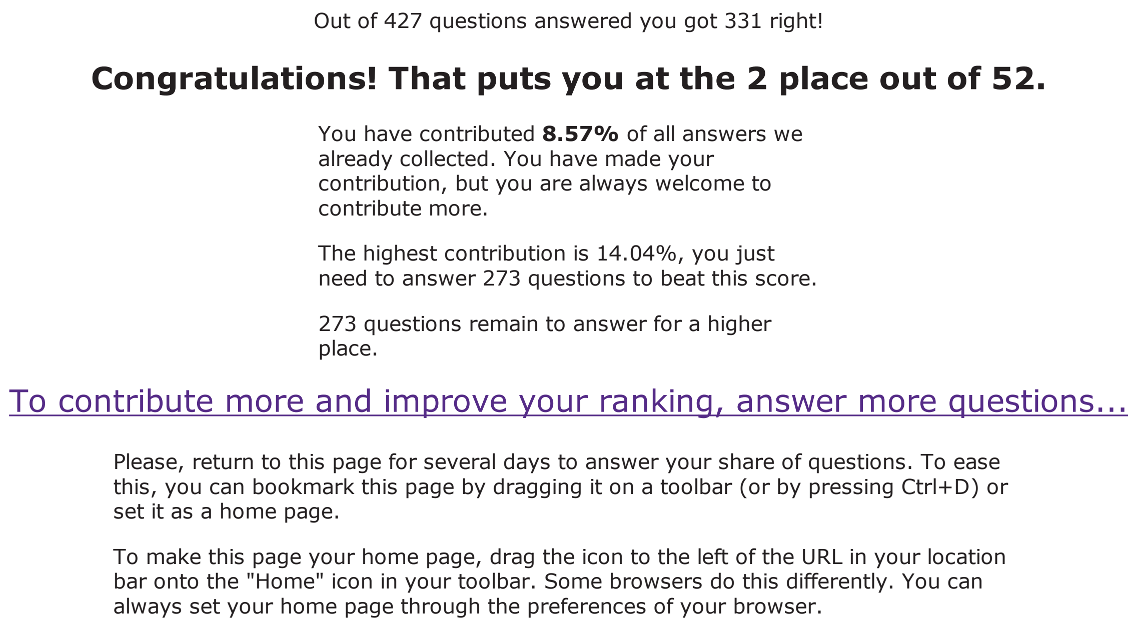
If we talk about **apple** does the word **fruit** mean the same as **produce**?

Figure 5.4: WSD Summary Evaluation Scenario 2.

In the second scenario we test the discrimination capability of the heuristics. To ease the cognitive load on a user we evaluate sense pairs, instead of presenting all senses at once. For example, we ask "If we talk about **apple**, does the word **fruit** mean the same as **produce**?" and present three answer options:

- "Yes" (incorrect), means that the user understood everything and the discrimination is bad;
- "No" (correct), means that the user understood everything and the discrimination is good;
- "I don't know" (unknown), means that the user did not understand the question or a word.

Figure 5.4 illustrates the question for the word "apple". Here the word "apple" is the original word which needs disambiguation. The word "fruit" is the summary for one sense and the word "produce" is the summary for another sense. Both summaries are generated using the same heuristic, the one being evaluated. Ideally, the summaries generated are different enough in their meaning, so that the user understands that they pertain to different senses and answers "No, it does not mean the same". We interpret that as the heuristic in question has produced summaries of sufficient discriminating power. If user answers "Yes, it does mean the same" we interpret that as the heuristic in question has failed to produce two summaries of sufficient discriminating power.



Out of 427 questions answered you got 331 right!

Congratulations! That puts you at the 2 place out of 52.

You have contributed **8.57%** of all answers we already collected. You have made your contribution, but you are always welcome to contribute more.

The highest contribution is 14.04%, you just need to answer 273 questions to beat this score.

273 questions remain to answer for a higher place.

To contribute more and improve your ranking, answer more questions...

Please, return to this page for several days to answer your share of questions. To ease this, you can bookmark this page by dragging it on a toolbar (or by pressing Ctrl+D) or set it as a home page.

To make this page your home page, drag the icon to the left of the URL in your location bar onto the "Home" icon in your toolbar. Some browsers do this differently. You can always set your home page through the preferences of your browser.

Figure 5.5: WSD Summary User Score Screen.

We split the questions into blocks of 20 questions of both types each. After a block of questions we showed the user a page, displayed in Figure 5.5, with absolute and relative contribution figures and performance to create an incentive to proceed with the task.

To measure the users agreement we handed out some questions to at least two different users. We have collected 308 at least double-rated questions for the scenario 1 (type 1 questions) and 301 at least double-rated questions for the scenario 2 (type 2 questions).

We use the generalized case to calculate raw users agreement indices [71]. As an item we consider an answer to a question. One question can have more than one answer, each answer given by a different user. It is also useful to keep in mind that type 1 questions can have multiple independent answers and therefore are more difficult for the users to agree upon. Table 5.8 provides the details on the overall proportion of agreement by the type of answer across all heuristics. The agreement of zero means the users do not agree at all and have given different answers on the same question. The agreement of one means the user agree completely and have given the same answers on the same question.

Table 5.8: Users Agreement Proportion by Question and Answer Type.

Answer Type	Type 1 Questions	Type 2 Questions
Unknown	0.18	0.225
None	0.23	n/a
Correct	**0.59**	**0.72**
semicorrect	0.04	n/a
Incorrect	0.35	0.43

Table 5.9 shows the details of the proportion of agreement for type 1 questions by different heuristics and answer types. Table 5.10 shows the details of the proportion of agreement for type 2 questions by different heuristics and the answer types.

Table 5.9: Users Agreement for Type 1 Questions.

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
unknown	0.50	0.50	0.00	0.13	0.00	0.00
none	0.50	0.00	0.00	0.09	0.40	0.00
correct	**0.37**	**0.00**	**0.53**	**0.62**	**0.61**	**0.66**
semicorrect	0.00	0.09	0.05	0.00	0.00	0.07
incorrect	0.50	0.50	0.00	0.30	0.36	0.40

Table 5.10: Users Agreement for Type 2 Questions.

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
unknown	0.18	1.00	0.25	0.25	0.00	0.00
correct	**0.82**	**0.00**	**0.72**	**0.75**	**0.60**	**0.53**
incorrect	0.00	0.00	0.47	0.38	0.66	0.50

One can see that in all cases (except hyponym heuristic in type 1 questions) users agreement for correct answers is high and it is higher for correct

Table 5.11: Noun Heuristics Comparison by Associative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
unknown	17	0	0	11	0	9
none	24	0	0	10	0	6
semicorrect	2	0	0	4	0	4
incorrect	22	0	0	14	0	17
> 1 selected sense	6	0	0	12	0	13
correct	**32**	0	0	**53**	0	**56**

answers than for the other answer types. That allows us to conclude that the users agree about the quality of generated summaries, collectively selecting the most associative and the most discriminating heuristic for each part of speech and we can order the heuristics by quality based on the users' answers.

Given that the different heuristics apply to the different parts of speech, it makes sense to view them by the part of speech they apply to. Here we describe how we order heuristics by quality on the example of noun heuristics. Table 5.11 gives the associative power figures for noun heuristics. In columns we present heuristics, in rows — answer types. We keep columns with the heuristics which do not apply to nouns (with zeros) to keep all tables uniform for the sake of easy comparison between the parts of speech. We compare the percentages in the last row, which contains correct answers, and conclude that the synset heuristic has the strongest associative power, because it gets the largest percentage of correct answers. Therefore we order the heuristics accordingly in the "By Associative Power" column of Table 5.12. We follow the same approach to order the heuristics by their discriminative power.

Appendix I provides the detailed results, organized by the part of speech and shows the associative power and the discriminating power comparisons for all heuristics. Table 5.12 summarizes the results of the evaluation by ordering the heuristics by associative power and by discriminating power.

Table 5.12: Sense Summarization Heuristics Quality.

Part of Speech	Heuristics Quality	
	By Associative Power	By Discriminating Power
noun	synset	hypernym
	hypernym	hyponym
	hyponym	synset
adjective	similar_to	similar_to
	synset	synset
verb	gloss	gloss
	synset	hyponym
	hypernym	hypernym
	hyponym	synset
adverb	synset	synset
	derived	derived

Chapter 6

Evaluation

We have evaluated the proposed solution for natural language metadata understanding using a synthetic approach. We have taken the large dataset used for evaluation of semantic matching [31], 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. We keep the dataset in two parts: “source”, combined from Google and Looksmart directories, and “target”, coming from Yahoo! directory, because these parts originate from different datasets, and this allows us to evaluate the performance on slightly different data. 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 discussed in Chapter 3 and for training discussed in Chapter 4. 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 [18] and, fi-

Table 6.1: 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

nally, created correct logical formulas for every label. For example, for the label "Acupuncture and Chinese Medicine" we have annotated tokens ("Acupuncture", "and", "Chinese", "Medicine"), POS tags (Acupuncture/NN, and/CC, Chinese/JJ, Medicine/NN), named entities (Acupuncture/O, and/O, Chinese/O, Medicine/O), correct senses (Acupuncture/n-#699073, Chinese/a#3048539, Medicine/n#5964779) and have created a formula: Acupuncture/n#699073 | Chinese/a#3048539 & Medicine/-n#5964779. Thus we have created a golden standard, which enables us to evaluate our solution. This dataset contains 47.86% of 1-token labels, 33.14%, 15.64% and 2.34% of 2-, 3- and 4-token labels, respectively. Longer labels constitute the remaining 1.02%. The average label length is 1.76 tokens, with the longest label being 8 tokens long. The most frequent POS tags are singular nouns (NN, 31.03%), plural nouns (NNS, 28.20%), proper nouns (NNP, 21.17%) and adjectives (JJ, 10.08%). An important POS, the coordinating conjunctions (CC) that can introduce ambiguity in a label, which, in turn, might be carried into a formula, occupy a notable 6.58%. In total, 26 parts of speech are present, and except the ones already mentioned, other 21 parts of speech occupy the remaining 2.91%.

Table 6.1 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 [33]. One can see that we have obtained a substantial improvement of approximately 15% over the previous results.

In Figure 6.1, we report the accuracy of the translation to description

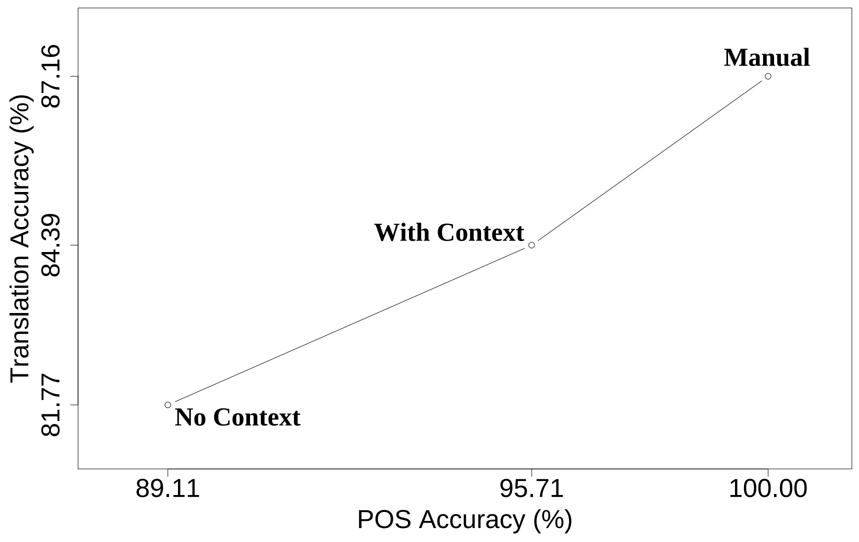


Figure 6.1: Contribution of POS Accuracy to the Translation Accuracy

logic formulas, in comparison to the POS tagger performances. We report two different POS tagging models (see Section 4.2.3) on the combined "source+target" dataset:

- **No Context** that corresponds to the best combined model, and
- **With Context** that is the best combined model trained with a context coming from the classification path of the labels.

The best combined model reached 89.11% PPL on the combined "source+-target" dataset. It compares well with the figures in the "all-except" row of Table 4.7 and shows that the model performs quite well on unseen data. We also tested the combined model trained with the context, and it reached 95.71% PPL. It compares well with the figures from the "path-cv" row of Table 4.7, also confirming that the model performs well on unseen data.

We can first observe an improvement of 6.6% in the POS tagging accuracy when using the context, which stresses the importance of such context. However, this only improves the translation accuracy by 2.62%. The improvement in POS tagging does not translate directly into a translation improvement, because of the other modules of the pipeline, such as the word sense disambiguation module, whose performance also influences the overall translation accuracy. Indeed, if we evaluate the translation with the manual POS tagging (*Manual* point in Figure 6.1), we observe that even with a "perfect" tagging, the translation accuracy does not improve much more. In comparison, a "perfect" tokenization (with a contextless POS tagging), improves the translation accuracy only by 0.02%.

To evaluate the influence of the preprocessing steps of tokenization and POS tagging of the performance of the parser, we supplied the parser with correctly tokenized labels and it reached 81.79% precision. These 0.02% can give an estimation of the tokenizer contribution. Then we supplied the parser with the correct tags and it reached 87.16% precision. These 5.37% can give an estimation of the POS tagger contribution. Out of this experiment we see that improving the POS tagger can give us a 5.35% improvement, while the remaining 18.23% should be reached by improving other translation pipeline modules.

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 the correct proper name: "Vin_Diesel". As another example consider the label "Early 20th Century", where the "previous" solution missed the concept "20th" because of too aggressive stopwords heuristics, while the proposed one recognized it. 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 because of the lack of context. Namely, the frequent misclassification which occurs 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 the examples with named entities, represented by common words are noticeable. 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.

The approach we propose here, with more accurate NLP models and the language structure analysis, achieves an accuracy of 84.39% in the translation task. This is a 17.95% improvement over the state of the art translation approach from [33] that reaches a 66.44% precision.

Chapter 7

Applications

Natural language metadata is widely used in various applications. As we already discussed, the natural language metadata is ambiguous and our proposed solution assists in tackling this problem. In other words, the best place and time to get rid of the ambiguity of a piece of a natural language metadata is when it is created and where it is created. At this moment the user who creates it is right there and, seeing this as part of the creation process, is more willing to cooperate in the task of creating a proper semantic annotation, contrary to returning later to the data and doing the annotation from scratch. This leads us to applications which might benefit if they integrate the proposed solution on the user interface level, and solve the problem in cooperation with the user, as the author of [64] concludes. This allows reaching two goals:

- it will make users more aware of semantics existence, importance and potential use and thus will create an incentive to write more accurate and descriptive email subject lines, programming interface names, titles, keywords and other natural language metadata elements.

- it will "put the user in the loop" and will help to maximally formalize the meaning that the user has in mind right at the moment of creation of an item of natural language metadata.

In Section 7.1 we provide two examples of the user-assisted scenarios of the proposed solution.

However, there are applications where user involvement is not feasible due to the volume of data or user unavailability. These applications can benefit from embedding the proposed solution and using it in the automatic processing mode. Therefore, in Section 7.2 we briefly illustrate the automated processing mode scenarios of the proposed solution.

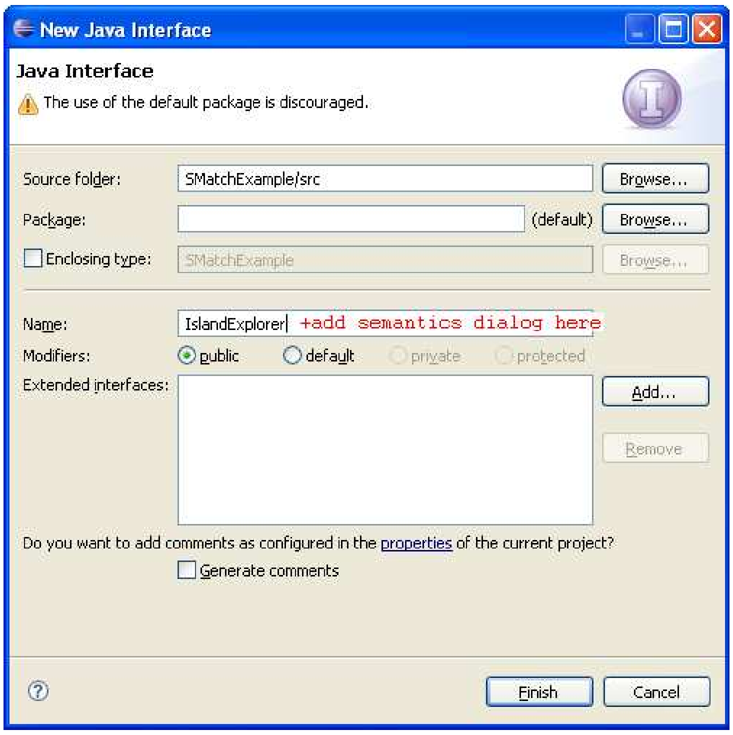
7.1 User-assisted Processing Scenarios

7.1.1 API Matching

Application program interfaces (APIs) contain function and variable names, which are a type of metadata and are written in (a subset of) natural language with the biggest difference from the normal one being the use of other means of separation between words. Tasks such as API integration, matching or achieving service interoperability [47] use semantic matching techniques, as demonstrated by the Open Knowledge[1] project's use of structure-preserving semantic matching [32]. Having precise semantics accompany the APIs description helps to do better matching and leads to successful interoperability.

A major part of modern development happens in integrated development environments (IDEs), such as Eclipse[2]. In such IDEs various program elements, such as APIs are created using dedicated user interfaces. Figure 7.1 shows the dialog for creating a Java interface. In addition, Figure 7.1 shows an insertion point for the optional semantic text enrichment dialog (such as the one in Figure 4.2), so that a developer can specify exactly the semantics of the interface being created. For modern Java-based

[1] `http://openk.org`
[2] `http://www.eclipse.org/`


New Java Interface

Java Interface

The use of the default package is discouraged.

Source folder: SMatchExample/src

Package: (default)

☐ Enclosing type: SMatchExample

Name: IslandExplorer| +add semantics dialog here

Modifiers: [✓ public] default private protected

Extended interfaces:

Do you want to add comments as configured in the properties of the current project?

☐ Generate comments

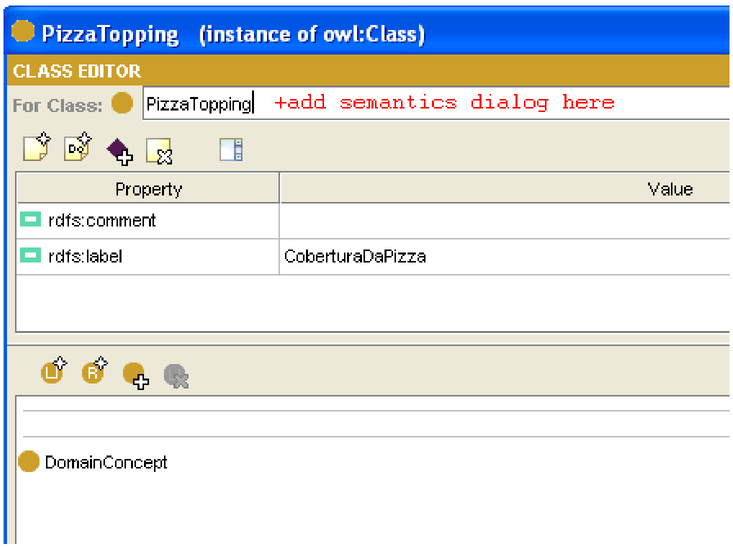
Figure 7.1: Eclipse New Java Interface Dialog.

tools there is a possibility to include semantics using the annotations mechanism, which is available since Java 5 release. Java compiler embeds the annotations in .class files and there is an API to access them at runtime. Storing semantically enriched function and variable names will improve semantic matching at runtime.

One might argue that some developers use dialogs and others prefer to type the code directly. For such cases, a mechanism similar to the code completion could be used to enable semantic annotation to happen in a text editor of the IDE.

7.1.2 Ontology Matching

Ontology matching is arguably an even more popular example and an application for semantic matching techniques. Here we find a similar situation. If ontology elements, such as class names, would have precise formal equiva-


PizzaTopping (instance of owl:Class)

CLASS EDITOR

For Class: PizzaTopping +add semantics dialog here

Property	Value
rdfs:comment	
rdfs:label	CoberturaDaPizza

DomainConcept

Figure 7.2: Protégé Class Editor Dialog.

lents associated with them, then the matching of two ontologies might get easier. Ontologies, as well as program source code, can be edited in a simple text editor, but specialized editors remain a more popular choice. Figure 7.2 shows a class editor dialog from a well-known ontology editor Protégé [2]. Similar to Figure 7.1 it shows a point for the addition of optional semantic text enrichment dialog.

It is possible to integrate the proposed solution directly into the ontology editor and its elements editing dialogs such that those ontology elements, which are written in natural language will have semantics generated and associated with them through the use of the proposed solution.

7.2 Automatic Processing Scenarios

7.2.1 Semantic Matching

As introduced in Section 1.1, one can see semantic matching as an operator that takes two tree-like structures and produces correspondences between those tree nodes that correspond semantically to each other. Some semantic matching algorithms reason over a logic formalism to deduce the correspondences. However, most classifications and schemas are created using natural language. Therefore there is a need to translate all the classification labels and schema elements into a formal language.

We take the S-Match [33] as an example of semantic matching algorithm. S-Match uses the notion of *concept of a label*, which specifies the set of documents one would classify under a label it encodes, and the notion of *concept at a node*, which specifies the set of documents one would classify under a node with a certain label and located in a specific position in a tree. S-Match splits the matching task into four steps:

1. for all labels L in the input trees, compute concepts of labels, C_L.
2. for all nodes N in the input trees, compute concepts at nodes, C_N.
3. for all pairs of labels in the input trees, compute relations among C_L.
4. for all pairs of nodes in the input trees, compute relations among C_N.

Our solution applies at the step 1, where the labels L are translated into concepts of labels C_L. Steps 1 and 2 are called *offline* processing, because they could be done anytime, even in the absence of a second tree. This particularity gives us flexibility to choose between automatic processing mode and user-assisted mode. S-Match is currently implemented as a command-line tool and therefore we have chosen automatic processing mode. We have integrated the proposed solution into an open source semantic matching framework S-Match[3] and this allows all algorithms which make part of the framework, including semantic matching [33], minimal semantic matching [29] and structure-preserving semantic matching [32], to use its services for the translation of natural language labels into L^C formulas. Specifically, we provide an implementation of the `IPreprocessor`[4]

[3]`http://semanticmatching.org`

[4]`http://semanticmatching.org/javadocs/it/unitn/disi/smatch/preprocessors/IPreprocessor.html`

interface in the class `PipelinePreprocessor`. This preprocessor uses the proposed solution to translate natural language labels into propositional description logics formulas.

7.2.2 Semantic Search

As introduced in Section 1.1, one of the proposals to improve search is to go from a syntactic search, which handles arbitrary sequences of characters and calculates string similarity, to a semantic search, which handles concepts and calculates semantic relatedness. However, most search terms and documents are in natural language.

We take *C-Search* [28] as an example of semantic search algorithm. The *C-Search*, as well as other information retrieval systems, takes a natural language query q (from a query set Q) and returns a set of documents d from a document collection D. Among other elements, the *C-Search* model uses *term* as an atomic element in document and query representations. For *C-Search*, *term* is a complex concept expressed in a propositional description logics language. However, both search terms q and documents in D are expressed in natural language.

Our solution applies at the translation of concepts expressed in natural language into *terms* expressed as propositional description logics formulas. For translating query terms q, we might apply a specialized version of a proposed solution. For translating natural language inside documents from a collection D into terms, we augmented our proposed solution and presented the robust pipeline in Section 4.3.

Chapter 8

Conclusion

Logics is heavily used for knowledge representation and information management and there are many applications which can make our life easier by managing our information and knowledge more efficiently. The problem is that natural language, and not logics, is the primary means of expressing our knowledge and information. We can enable a great many applications by providing easy and cheap way of extracting semantics of natural language and representing it in logics.

We have shown several motivating applications in Chapter 1 and have given more examples in Chapter 7. We hypothesize that these applications mostly operate on a specific subset of natural language that we call *descriptive phrases* and that descriptive phrases can be mapped onto a subset of Description Logics, namely propositional Description Logics language L^C.

We have studied several datasets that represent the kind of data our motivating applications often use and presented the results in Chapter 3. We call this kind of data a *natural language metadata*. We have found that natural language metadata almost always consists of descriptive phrases. We have found that descriptive phrases differ from the normal language as it is used in news stories and books and have shown that natural language metadata deserves to be recognized as a novel natural language domain.

We have analysed how it is different from normal language and have found out that the modern natural language processing tools such as tokenizers, part of speech taggers and named entity recognizers need an adaptation to work well with natural language metadata.

Based on our analysis made in Chapter 3, we have explored the key natural language processing problems which need to be addressed to process descriptive phrases and have presented in Chapter 4 the natural language metadata understanding architecture, complete with the models, the algorithms and the implementation. The modular structure of the proposed architecture allows easy customization for each particular need of the target application, as well as for potential changes in the input language and the output logic formalism. We have shown that descriptive phrases can be mapped into propositional Description Logics language L^C formulas.

We have evaluated four out of six modules of the proposed architecture in an isolation and have brought their performance to the state of the art levels on the novel domain of natural language metadata. In Chapter 6 we have evaluated the architecture as a whole using synthetic approach on the large manually annotated dataset widely used for semantic matching evaluation. We have shown that the proposed solution improves state of the art performance by a margin of 15% to 17% reaching translation accuracy of 84.39%.

In addition we have explored where the user can make the biggest improvement in the translation process and we have shown the way to put the user in the loop. In our architecture we have addressed the need to provide processing in different modalities: fully automated and user-assisted. We have further explored the load on the user in the word sense disambiguation task as in the most important point of processing and in Chapter 5 we have shown how to ease the cognitive load on the user by providing the word sense summary generation algorithm accompanied by the user interface. We have complemented the proposed architecture with a user interface prototype, which target applications can use to exploit the services of user-assisted language to logic translation.

We have demonstrated where and how the proposed solution can be embedded into different applications in the fully automated and in the user-assisted processing mode. We have integrated the proposed architecture into the open source S-Match framework and prepare to release the proposed architecture under an open source license.

Future work will follow different directions. First, we would like to explore deeper the impact of the proposed solution on the applications and execute an application-dependent evaluations. Second, we would like to explore a more innovative architectures, trying new decision-making processes and evaluating non-sequential decision-making architectures while trying to reuse existing modules, algorithms and implementations. Third, we would like to explore the options for augmenting the context to solve short context problem we observe in natural language metadata. We can do this, for instance, by exploiting the available hierarchy and differently treating the cases when the complete hierarchy is available for processing, contrary to the current approach of treating each label in an isolation. Fourth, we would like to integrate different proposed grammars into a unified grammar, as this would eliminate the necessity to choose which grammar out of available ones better suits particular case. Fifth, we would like to explore tighter integration between machine-learning and knowledge-based approaches, because our target applications almost always provide a knowledge base which contains at least concepts and often even more lexical information.

Bibliography

[1] SKOS Simple Knowledge Organization System. `http://www.w3.org/2004/02/skos/`.

[2] The Protégé Ontology Editor and Knowledge Acquisition System. `http://protege.stanford.edu`.

[3] *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25–30 June 2005, University of Michigan, USA*. The Association for Computational Linguistics, 2005.

[4] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2003.

[5] Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71, 1996.

[6] Raffaella Bernardi, Diego Calvanese, and Camilo Thorne. Expressing DL-Lite Ontologies with Controlled English. In Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris, editors, *Proceedings of the 20th International Workshop on Description Logic (DL 2007)*, volume 250

of *CEUR Electronic Workshop Proceedings, http://ceur-ws.org/*, pages 195–202. CEUR-WS.org, 2007.

[7] Raffaella Bernardi, Diego Calvanese, and Camilo Thorne. Lite Natural Language. In *Proceedings of IWCS-7*, 2007.

[8] Abraham Bernstein and Esther Kaufmann. GINO – A Guided Input Natural Language Ontology Editor. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 2006.

[9] Abraham Bernstein, Esther Kaufmann, Norbert E. Fuchs, and June von Bonin. Talking to the Semantic Web — A Controlled English Query Interface for Ontologies. In *14th Workshop on Information Technology and Systems*, pages 212–217, December 2004.

[10] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July 2006. Association for Computational Linguistics.

[11] 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.

[12] Michael Collins. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics*, 29(4):589–637, 2003.

[13] Anne Cregan, Rolf Schwitter, and Thomas Meyer. Sydney OWL Syntax – towards a Controlled Natural Language Syntax for OWL 1.1. In Golbreich et al. [36].

[14] Michael Crystal, Alex Baron, Katherine Godfrey, Linnea Micciulla, Yvette J. Tenney, and Ralph M. Weischedel. A Methodology for Extrinsically Evaluating Information Extraction Performance. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada.* The Association for Computational Linguistics, 2005.

[15] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA.* The Association for Computational Linguistics, 2002.

[16] Ronald Denaux, Vania Dimitrova, Anthony G. Cohn, Catherine Dolbear, and Glen Hart. ROO: Involving Domain Experts in Authoring OWL Ontologies. In Christian Bizer and Anupam Joshi, editors, *International Semantic Web Conference (Posters & Demos)*, volume 401 of *CEUR Workshop Proceedings.* CEUR-WS.org, 2008.

[17] 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*, 2009.

[18] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* Language, Speech, and Communication. The MIT Press, Cambridge, MA, May 1998.

[19] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25–30 June 2005, University of Michigan, USA* [3].

[20] Enrico Franconi, Michael Kifer, and Wolfgang May, editors. *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3–7, 2007, Proceedings*, volume 4519 of *Lecture Notes in Computer Science*. Springer, 2007.

[21] Norbert E. Fuchs, Stefan Höfler, Kaarel Kaljurand, Fabio Rinaldi, and Gerold Schneider. Attempto Controlled English: A Knowledge Representation Language Readable by Humans and Machines. In Norbert Eisinger and Jan Maluszynski, editors, *Reasoning Web*, volume 3564 of *Lecture Notes in Computer Science*, pages 213–250. Springer, 2005.

[22] Norbert E. Fuchs, Kaarel Kaljurand, and Gerold Schneider. Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces. In Geoff Sutcliffe and Randy Goebel, editors, *FLAIRS Conference*, pages 664–669. AAAI Press, 2006.

[23] Norbert E. Fuchs, Uta Schwertel, and Rolf Schwitter. Attempto Controlled English — Not Just Another Logic Specification Language. In Pierre Flener, editor, *Logic Programming Synthesis and Transformation, 8th International Workshop, Manchester, UK, June 15–19, 1998, Proceedings*, volume 1559 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 1998.

[24] Norbert E. Fuchs, Uta Schwertel, and Sunna Torge. Controlled Natural Language Can Replace First-Order Logic. In *The 14th IEEE International Conference on Automated Software Engineering, 12–15 October 1999, Cocoa Beach, Florida, USA, Proceedings*, pages 295–298. IEEE Computer Society, 1999.

[25] Norbert E. Fuchs, Uta Schwertel, and Sunna Torge. A Natural Language Front-End to Model Generation. *Journal of Language and Computation*, First(2):199–214, December 2000.

[26] Norbert E. Fuchs and Rolf Schwitter. Web-Annotations for Humans and Machines. In Franconi et al. [20], pages 458–472.

[27] Michael R. Genesereth. Knowledge Interchange Format. In James F. Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning. Cambridge, MA, USA, April 22–25, 1991*, pages 599–600. Morgan Kaufmann Publishers, 1991.

[28] Fausto Giunchiglia, Uladzimir Kharkevich, and Ilya Zaihrayeu. Concept Search. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 429–444. Springer, 2009.

[29] Fausto Giunchiglia, Vincenzo Maltese, and Aliaksandr Autayeu. Computing Minimal Mappings. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natalya Fridman Noy, and Arnon Rosenthal, editors, *OM*, volume 551 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[30] Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu. Encoding Classifications into Lightweight Ontologies. *Journal on Data Semantics*, 8:57–81, 2007.

[31] Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko. A Large Dataset for the Evaluation of Ontology Matching. *The Knowledge Engineering Review Journal*, 24(2):137–157, 2009.

[32] Fausto Giunchiglia, Mikalai Yatskevich, and Fiona McNeill. Structure Preserving Semantic Matching. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, and Bin He, editors, *OM*, volume 304 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[33] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic Matching: Algorithms and Implementation. *Journal on Data Semantics*, 9:1–38, 2007.

[34] Fausto Giunchiglia and Ilya Zaihrayeu. Lightweight Ontologies. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 1613–1619. Springer US, 2009.

[35] Fausto Giunchiglia, Ilya Zaihrayeu, and Uladzimir Kharkevich. Formalizing the Get-Specific Document Classification Algorithm. In László Kovács, Norbert Fuhr, and Carlo Meghini, editors, *Research and Advanced Technology for Digital Libraries, 11th European Conference, ECDL 2007, Budapest, Hungary, September 16-21, 2007, Proceedings*, volume 4675 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2007.

[36] Christine Golbreich, Aditya Kalyanpur, and Bijan Parsia, editors. *Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, Innsbruck, Austria, June 6-7, 2007*, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[37] Glen Hart, Martina Johnson, and Catherine Dolbear. Rabbit: Developing a Control Natural Language for Authoring Ontologies. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, volume 5021 of *Lecture Notes in Computer Science*, pages 348–360. Springer, 2008.

[38] James A. Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems*, 16(2):30–37, 2001.

[39] Martin Hepp. Representing the Hierarchy of Industrial Taxonomies in OWL: The gen/tax Approach. In *Proceedings of the ISWC Workshop Semantic Web Case Studies and Best Practices for eBusiness (SWCASE05), Galway, Irland*, 2005.

[40] 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 Franconi et al. [20], pages 129–144.

[41] Stefan Hoefler. The Syntax of Attempto Controlled English: An Abstract Grammar for ACE 4.0. Technical Report ifi-2004.03, Department of Informatics, University of Zurich, Zurich, Switzerland, 2004.

[42] Kaarel Kaljurand and Norbert E. Fuchs. Bidirectional Mapping Between OWL DL and Attempto Controlled English. In José Júlio Alferes, James Bailey, Wolfgang May, and Uta Schwertel, editors, *Principles and Practice of Semantic Web Reasoning, 4th International Workshop, PPSWR 2006, Budva, Montenegro, June 10–11, 2006, Revised Selected Papers*, volume 4187 of *Lecture Notes in Computer Science*, pages 179–189. Springer, 2006.

[43] Kaarel Kaljurand and Norbert E. Fuchs. Mapping Attempto Controlled English to OWL DL. In Sure and Domingue [70].

[44] Kaarel Kaljurand and Norbert E. Fuchs. Verbalizing OWL in Attempto Controlled English. In Golbreich et al. [36].

[45] Tobias Kuhn, Loic Royer, Norbert E. Fuchs, and Michael Schroeder. Improving Text Mining with Controlled Natural Language: A Case Study for Protein Interations. In Ulf Leser, Barbara Eckman, and Felix Naumann, editors, *Proceedings of the 3rd International Workshop on Data Integration in the Life Sciences 2006 (DILS'06)*, number 4075 in Lecture Notes in Bioinformatics. Springer, 2006.

[46] Massimo Marchiori. Towards a People's Web: Metalog. In *2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, volume 0, pages 320–326, Beijing, China, September 20–24 2004. IEEE Computer Society.

[47] Fiona McNeill, Fausto Giunchiglia, Paolo Besana, and Juan Pane. *Cases on Semantic Interoperability for Information Systems Integration: Practices and Applications*, chapter Service Integration through Structure-Preserving Semantic Matching. IGI Global, 2009.

[48] Rada Mihalcea and Andras Csomai. SenseLearner: Word Sense Disambiguation for All Words in Unrestricted Text. In *43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25–30 June 2005, University of Michigan, USA* [3].

[49] Dan I. Moldovan and Vasile Rus. Logic Form Transformation of WordNet and its Applicability to Question Answering. In *ACL*, pages 394–401, 2001.

[50] Dan I. Moldovan and Vasile Rus. Transformation of WordNet Glosses into Logic Forms. In Ingrid Russell and John F. Kolen, editors, *FLAIRS Conference*, pages 459–463. AAAI Press, 2001.

[51] Ian Niles and Adam Pease. Towards a Standard Upper Ontology. In Chris Welty and eds Barry Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Ogunquit, Maine, October 17-19 2001.

[52] Adam Pease and Christiane Fellbaum. Language to Logic Translation with PhraseBank. In *Proceedings of the 2nd International WordNet Conference*, pages 187–192, 2004.

[53] Adam Pease and William Murray. An English to Logic Translator for Ontology-based Knowledge Representation Languages. In *Proceedings of the 2003 IEEE International Conference on Natural Language Processing and Knowledge Engineering*, pages 777–783, Beijing, China, 2003. Prentice Hall.

[54] Jonathan Pool. Can Controlled Languages Scale to the Web? In *5th International Workshop on Controlled Language Applications (CLAW2006 at AMTA 2006)*, 2006.

[55] 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.

[56] Vasile Rus. High Precision Logic Form Transformation. In *ICTAI*, pages 288–, 2001.

[57] Vasile Rus, Dan I. Moldovan, and Orest Bolohan. Bracketing Compound Nouns for Logic Form Derivation. In Susan M. Haller and Gene

Simmons, editors, *FLAIRS Conference*, pages 198–202. AAAI Press, 2002.

[58] Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. Multiword Expressions: A Pain in the Neck for NLP. In Alexander F. Gelbukh, editor, *CICLing*, volume 2276 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002.

[59] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, volume 4, pages 142–147, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[60] Beatrice Santorini. Part-of-Speech Tagging Guidelines for the Penn Treebank Project. Technical report, University of Pennsylvania, 1990. (3rd revision, 2nd printing).

[61] Uta Schwertel. Controlling Plural Ambiguities in Attempto Controlled English. In *Proceedings of the 3rd Internaltional Workshop on Controlled Language Applications*, Seattle, Washington, 2000.

[62] Rolf Schwitter. English as a Formal Specification Language. In *DEXA Workshops*, pages 228–232. IEEE Computer Society, 2002.

[63] Rolf Schwitter. Creating and Querying Linguistically Motivated Ontologies. In *Proceedings of the Knowledge Representation Ontology Workshop (KROW 2008), Conference in Research and Practice in Information Technology*, volume 90, pages 71–80, 2008.

[64] Rolf Schwitter. Reconstructing Hard Problems in a Human-Readable and Machine-Processable Way. In Tu Bao Ho and Zhi-Hua Zhou,

editors, *PRICAI*, volume 5351 of *Lecture Notes in Computer Science*, pages 1046–1052. Springer, 2008.

[65] Rolf Schwitter. Working for Two: A Bidirectional Grammar for a Controlled Natural Language. In Wayne Wobcke and Mengjie Zhang, editors, *Australasian Conference on Artificial Intelligence*, volume 5360 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2008.

[66] Rolf Schwitter, Kaarel Kaljurand, Anne Cregan, Catherine Dolbear, and Glen Hart. A Comparison of three Controlled Natural Languages for OWL 1.1. In *4th OWL Experiences and Directions Workshop (OWLED 2008 DC)*, Washington, USA, 1–2 April 2008.

[67] 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.

[68] Rolf Schwitter and Marc Tilbrook. Controlled Natural Language meets the Semantic Web, 2004.

[69] Rolf Schwitter and Marc Tilbrook. Let's Talk in Description Logic via Controlled Natural Language. In *Logic and Engineering of Natural Language Semantics (LENLS2006)*, 2006.

[70] York Sure and John Domingue, editors. *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11–14, 2006, Proceedings*, volume 4011 of *Lecture Notes in Computer Science*. Springer, 2006.

[71] John Uebersax. Raw Agreement Indices. `http://www.john-uebersax.com/stat/raw.htm`.

[72] Mark van Assem, Véronique Malaisé, Alistair Miles, and Guus Schreiber. A Method to Convert Thesauri to SKOS. In Sure and Domingue [70], pages 95–109.

[73] Mark van Assem, Maarten R. Menken, Guus Schreiber, Jan Wielemaker, and Bob J. Wielinga. A Method for Converting Thesauri to RDF/OWL. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *The Semantic Web, 3rd International Semantic Web Conference, ISWC 2004, Hiroshima, Japan, November 7–11, 2004, Proceedings*, volume 3298 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2004.

[74] Chong Wang, Miao Xiong, Qi Zhou, and Yong Yu. PANTO: A Portable Natural Language Interface to Ontologies. In Franconi et al. [20], pages 473–487.

[75] Colin White and Rolf Schwitter. An Update on PENG Light. In L. Pizzato and R. Schwitter (eds.), editors, *Proceedings of ALTA 2009*, pages 80–88, Sydney, Australia, 2009.

[76] Bob J. Wielinga, A. Th. Schreiber, Jan Wielemaker, and Jacobijn Sandberg. From thesaurus to ontology. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001), October 21-23, 2001, Victoria, BC, Canada*, pages 194–201. ACM, 2001.

[77] Bob J. Wielinga, Jan Wielemaker, Guus Schreiber, and Mark van Assem. Methods for Porting Resources to the Semantic Web. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings*, volume 3053 of *Lecture Notes in Computer Science*, pages 299–311. Springer, 2004.

[78] A. Wyner, K. Angelov, G. Barzdins, D. Damljanovic, N. Fuchs, S. Hoefler, K. Jones, K. Kaljurand, T. Kuhn, M. Luts, J. Pool, M. Rosner, R. Schwitter, and J. Sowa. On Controlled Natural Languages: Properties and Prospects. In N.E. Fuchs, editor, *N.E. Fuchs (ed.), Workshop on Controlled Natural Languages*, volume 5972 of *LNCS/LNAI*. Springer, 2010.

[79] 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 Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 623–636. Springer, 2007.

Appendix A

DMoz Statistics

Table A.1: DMoz Labels Lengths Distribution.

Token count	Label count	Share of labels (%)
1	251 140	50.8336
2	86 454	17.4993
3	136 425	27.6140
4	12 232	2.4759
5	3 762	0.7615
6	2 641	0.5346
7	810	0.1640
8	319	0.0646
9	150	0.0304
10	62	0.0125
11	19	0.0038
12	13	0.0026
13	11	0.0022
14	1	0.0002
15	1	0.0002
16	1	0.0002
17	1	0.0002
22	1	0.0002

Table A.2: DMoz Common Labels Lengths Distribution.

Token count	Label count	Share of labels (%)
1	160 880	49.5968
2	49 035	15.1167
3	108 838	33.5530
4	3 821	01.1780
5	607	00.1871
6	1 080	00.3329
7	72	00.0222
8	22	00.0068
9	20	00.0062
14	1	00.0003

Table A.3: DMoz Common Labels POS Tag Distribution.

POS Tag	Token count	Share of tokens (%)
NN	298 342	51.59
NNS	132 797	22.96
CC	107 006	18.50
JJ	36 888	6.38
,	2 173	0.38
IN	486	0.08
CD	361	0.06
POS	97	0.02
NNPS	76	0.01
NNP	73	0.01
JJR	9	0.00
TO	3	0.00

Table A.4: Top 20 DMoz POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
104 695	33.70	NN	Compensation
62 625	20.16	NN CC NN	Pregnancy and Birth
44 847	14.43	NNS	Sidecars
21 854	7.03	NNS CC NNS	Invitations and Announcements
13 047	4.20	NN NNS	Restaurant Chains
12 281	3.95	JJ NN	Global Software
9 773	3.15	JJ	Veterinary
9 252	2.98	JJ NNS	Used Vehicles
8 879	2.86	NN CC NNS	Polytechnics and Institutes
8 686	2.80	NNS CC NN	Products and Equipment
6 654	2.14	NN NN	Defense Litigation
815	0.26	NN NN CC NN	Vehicle Repair and Maintenance
545	0.18	NN NN NN	Supply Chain Management
515	0.17	NN CC JJ NNS	Forensics and Anti-Forensic Degaussers
482	0.16	JJ NN CC NNS	Military Equipment and Parts
399	0.13	JJ , JJ , CC JJ	Biological, Chemical, and Radiological
352	0.11	JJ NN NN	Medical Call Scheduling
341	0.11	CD	2
324	0.10	NN NN NNS	Volunteer Focus Groups
319	0.10	JJ JJ	Classical Indian

Table A.5: Unambiguous DMoz POS Tag Patterns.

POS Tag Pattern	Label Count	Share (%)	Disambiguation
NNS CC NN NNS	97	10.13	NNS | (NN & NNS)
NNS CC NN NN	21	2.19	NNS | (NN & NN)
NNS CC JJ NNS	15	1.57	NNS | (JJ & NNS)
JJ CC NN NNS	6	0.63	(JJ | NN) & NNS
NNS CC JJ NN	6	0.63	NNS | (JJ & NN)
NN NN CC JJ NN	4	0.42	(NN & NN) | (JJ & NN)
NN NNS CC NN NN	4	0.42	(NN & NNS) | (NN & NN)
JJ CC NN NN	3	0.31	(JJ | NN) & NN
NN CC NN NN NNS	3	0.31	(NN | (NN & NN)) & NNS
NN IN JJ CC JJ NN	3	0.31	NN & (JJ | JJ) & NN
NN NNS , NNS CC NNS	3	0.31	NN & (NNS | NNS | NNS)
JJ NN CC JJ NN	3	0.31	(JJ & NN) | (JJ & NN)
NNS , NNS CC NN	2	0.21	NNS | NNS | NN
NN CC NNS NNS	2	0.21	(NN | NNS) & NNS
JJ , NN CC NN NN	2	0.21	(JJ | NN | NN) & NN
NN NN NN CC NN	2	0.21	(NN & NN) & (NN | NN)
JJ NNS CC JJ NNS	2	0.21	(JJ & NNS) | (JJ & NNS)
NN CC JJ NN NNS	2	0.21	(NN | JJ) & NN & NNS
NNS CC NNS NN	2	0.21	(NNS | NNS) & NN
JJ , JJ , CC JJ NNS	2	0.21	(JJ | JJ | JJ) & NNS

Table A.6: Some Ambiguous DMoz POS Tag Patterns.

POS Tag Pattern	Label Count	Share (%)	Disambiguation	Share (%)
			Disambiguation	Share (%)
NN CC NN NN	165	17.22	NN | (NN & NN)	41.82
			(NN | NN) & NN	58.18
NN CC NN NNS	139	14.51	NN | (NN & NNS)	15.11
			(NN | NN) & NNS	84.89
NN NNS CC NNS	86	8.98	(NN & NNS) | NNS	13.95
			NN & (NNS | NNS)	86.05
NN NN CC NN	61	6.37	NN & (NN | NN)	73.77
			(NN & NN) | NN	26.23
JJ NNS CC NNS	51	5.32	JJ & (NNS | NNS)	88.24
			(JJ & NNS) | NNS	11.76
NN CC JJ NN	29	3.03	(NN | JJ) & NN	37.93
			NN | (JJ & NN)	62.07
NN CC JJ NNS	26	2.71	NN | (JJ & NNS)	38.46
			(NN | JJ) & NNS	61.54
NN NN CC NNS	22	2.30	(NN & NN) | NNS	18.18
			NN & (NN | NNS)	81.82
JJ NN CC NN	19	1.98	(JJ & NN) | NN	15.79
			JJ & (NN | NN)	84.21
NN NNS CC NN	18	1.88	(NN & NNS) | NN	11.11
			NN & (NNS | NN)	88.89
NN NN CC NN NN	16	1.67	(NN & NN) | (NN & NN)	81.25
			NN & (NN | NN) & NN	18.75
JJ NNS CC NN	12	1.25	(JJ & NNS) | NN	50.00
			JJ & (NNS | NN)	50.00
JJ NN CC NNS	11	1.15	(JJ & NN) | NNS	27.27
			JJ & (NN | NNS)	72.73
JJ NN CC NN NN	10	1.04	(JJ & NN) | (NN & NN)	40.00
			JJ & (NN | NN) & NN	60.00
NN CC NN NN NN	8	0.84	(NN | NN) & NN & NN	62.50
			(NN | (NN & NN)) & NN	37.50
NN NN CC NN NNS	5	0.52	(NN & NN) | (NN NNS)	80.00
			((NN & NN) | NN) & NNS	20.00

Appendix B

eCl@ss Statistics

Table B.1: eCl@ss Labels Lengths Distribution.

Token count	Label count	Share of labels (%)
1	2 512	17.4529
2	3 661	25.4360
3	1 236	08.5875
4	1 260	08.7543
5	1 958	13.6038
6	1 338	09.2962
7	1 035	07.1910
8	671	04.6620
9	335	02.3275
10	153	01.0630
11	89	00.6184
12	56	00.3891
13	32	00.2223
14	23	00.1598
15	5	00.0347
16	13	00.0903
17	3	00.0208
18	3	00.0208
19	3	00.0208
20	1	00.0069
21	1	00.0069
22	1	00.0069
23	3	00.0208
26	1	00.0069

Table B.2: eCl@ss Labels POS Tag Distribution.

POS Tag	Token count	Share of tokens (%)
NN	31 888	56.3890
)	6 204	10.9708
(	5 906	10.4439
JJ	5 421	9.5862
,	3 324	5.8780
NNS	2 078	3.6746
IN	977	1.7277
CC	340	0.6012
CD	117	0.2069
TO	64	0.1132
:	45	0.0796
VBN	39	0.0690
JJR	29	0.0513
VB	29	0.0513
DT	22	0.0389
RB	18	0.0318
PP$	17	0.0301
NNP	17	0.0301
VBG	9	0.0159
FW	4	0.0071
.	2	0.0035

Table B.3: Top 20 eCl@ss POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
2 853	19.82	NN NN	Methyl benzoylformate
2 457	17.07	NN	Acylase
583	4.05	NN NN (NN)	Laboratory app. (repair)
567	3.94	NN NN NN	Block heat exchanger
566	3.93	JJ NN	Exterior radiator
361	2.51	NN (NN)	Cooling (lab.)
228	1.58	NN NN (JJ)	Packing plant (compl.)
210	1.46	JJ NN NN	1,8-Naphthalic acid anhydride
205	1.42	NN (JJ)	Scraper (other)
157	1.09	NN (NN NN)	Bag (packing material)
144	1.00	NN (JJ NN)	Recorder (special design)
120	0.83	NN NN (NNS)	load-break switch (parts)
119	0.83	NN NN (NN NN)	door opener (bell system)
117	0.81	JJ NN (NN)	capillary pipette (laboratory)
112	0.78	NN NN (JJ NN)	power supply (decentralized system)
111	0.77	NN NN NN (NN)	Pipeline form piece (glass)
106	0.74	NN NN (NN , NN)	Cutting mach. (maint., serv.)
106	0.74	NN (NN , NN)	Cap (shaking, lab)
95	0.66	NN (NNS)	terminal (accessories)
95	0.66	NN , NN	Nonprint, Multimedia

Appendix C

LCSH Statistics

Table C.1: LCSH Labels Lengths Distribution.

Token count	Label count	Share of labels (%)
1	28 110	08.3727
2	67 678	20.1583
3	48 138	14.3382
4	45 279	13.4866
5	46 800	13.9396
6	34 846	10.3791
7	26 757	07.9697
8	15 948	04.7502
9	8 495	02.5303
10	5 685	01.6933
11	3 061	00.9117
12	2 115	00.6300
13	1 297	00.3863
14	683	00.2034
15	356	00.1060
16	164	00.0488
17	95	00.0283
18	57	00.0170
19	38	00.0113
20	28	00.0083
21	29	00.0086
22	20	00.0060
23	8	00.0024
24	9	00.0027
25	2	00.0006
26	1	00.0003

Table C.2: LCSH Labels POS Tag Distribution.

POS Tag	Token count	Share of tokens (%)
NNP	386 302	26.1166
NN	331 775	22.4302
,	210 808	14.2520
NNS	164 186	11.1001
JJ	129 578	8.7603
(	87 533	5.9178
)	87 525	5.9173
CC	26 089	1.7638
IN	24 557	1.6602
CD	17 314	1.1705
FW	5 051	0.3415
POS	2 485	0.1680
DT	2 059	0.1392
TO	1 832	0.1239
:	1 625	0.1099
RB	212	0.0143
JJR	171	0.0116
NNPS	42	0.0028
VBG	1	0.0001
VB	1	0.0001

Table C.3: Top 20 LCSH POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
22 192	6.61	NNP NN	Teach family
14 444	4.30	NNP NNP (NNP)	Coconucos Range (Colombia)
13 474	4.01	NNP	Myzocallis
11 211	3.34	JJ NN	Negative staining
8 771	2.61	NN NNS	Museum docents
8 030	2.39	NNP NNP NNP (NNP)	White River Valley (Wash.)
7 856	2.34	NN	Amortization
7 714	2.30	NN NN	Lipoprotein lipase
7 379	2.20	JJ NNS	Photoelectric measurements
7 291	2.17	NNP NNP	Spotted cutworm
6 757	2.01	NNS	Quarks
6 104	1.82	NNS , NNP	Canyons, Alabama
4 437	1.32	NNS , JJ	Proverbs, Ladino
3 738	1.11	NNP (JJ NN)	Maat (Egyptian deity)
3 648	1.09	NNP NNP (NNP , NNP)	Whitemarsh Hall (Philadelphia, Pa.)
3 596	1.07	NN , NN	Memory, Fiction
3 493	1.04	NNS , NN	Electrons, Compton
3 381	1.01	NN , JJ	Pottery, Akan
3 015	0.90	NNP NNP NNP (JJ NN)	Rumplemayer, Fenton (Fictitious character)
2 749	0.82	NN IN NN	Nude in art

Table C.4: Some LCSH Chunk Types with Examples.

Chunk-Pattern	POS Tag pattern	Example
event, geo, time	NNP NNP NNP NNP, NNP, NNP, CD	Clydeside Apprentices' Strike, Glasgow, Scotland, 1937
event, time	NNP NNP, CD	Turco-Montenegrin Wars, 1711-1714
event, time, geo	NNP NNP, CD, NNP	World War, 1939-1945, Poland
event, time, NP	NNP NNP, CD, NNS	Sino-Japanese War, 1894-1895, Causes
event, time, NP, geo	NNP NNP, CD, NNS, NNP	Crimean War, 1853-1856, Campaigns, Romania
event, time, RNP	NNP NNP, CD, JJ NNS, JJ	Yugoslav War, 1991-1995, Personal narratives, Croatian
geo	NNP, NNP	Mexico, Southeast
geo (geo-dis)	NNP NNP (NNP)	Coconucos Range (Colombia)
geo (geo-dis), event, time	NNP (NNP), NN, NN, CD	Magdeburg (Germany), History, Bombardment, 1945
geo (geo-dis), NP	NNP (NNP), NN	Tokyo (Japan), History
geo, domain	NNP, IN NN	Chile, In art
geo, NP	NNP, JJ NNS	Paraguay, Economic conditions
geo, NP, event, time	NNP, NN, JJ NN, CD	France, History, Papal Interdict, 1199-1200
geo, NP, event, time, NP	NNP, NN, NN, CD, NNS	Hungary, History, Revolution, 1956, Sources
geo, NP, geo	NNP, NNS, NNP	Greece, Colonies, Asia
geo, NP, name, time	NNP NNP, NN, NNP NNP, CD	Great Britain, History, Henry VII, 1485-1509
geo, NP, NP	NNP, JJ NN, JJ NN	England, Intellectual life, 20th century
geo, NP, time	NNP, NN CC NN, CD	Russia, Politics and government, 1894-1917

Appendix D

NALT Statistics

Table D.1: NALT Labels Lengths Distribution.

Token count	Label count	Share of labels (%)
1	18 246	42.3951
2	20 533	47.7090
3	2 018	04.6889
4	1 776	04.1266
5	271	00.6297
6	163	00.3787
7	20	00.0465
8	9	00.0209
9	1	00.0023
10	1	00.0023

Table D.2: NALT Labels POS Tag Distribution.

POS Tag	Token count	Share of tokens (%)
NNP	49 181	65.5450
NN	15 470	20.6173
NNS	5 732	7.6392
JJ	3 520	4.6912
)	423	0.5637
(	418	0.5571
CC	170	0.2266
IN	68	0.0906
,	39	0.0520
TO	5	0.0067
VB	2	0.0027
FW	2	0.0027
NNPS	1	0.0013
RB	1	0.0013
VBD	1	0.0013
JJR	1	0.0013

Table D.3: Top 20 NALT POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
13 356	31.03	NNP NNP	Rhode Island
12 325	28.64	NNP	Diachros
3 858	8.96	NN	thyroglobulin
2 651	6.16	NN NN	milk allergy
2 063	4.79	NNS	defoliants
1 605	3.73	NN NNS	livestock exhibitions
1 385	3.22	JJ NN	antigenic variation
1 252	2.91	NNP NNP NNP NNP	Potato black ringspot virus
1 230	2.86	JJ NNS	inactivated vaccines
875	2.03	NNP NNP NNP	Helena National Forest
236	0.55	NN NN NN	quantity food preparation
210	0.49	NNP NN	Gallionella group
177	0.41	JJ NN NN	amnesic shellfish poisoning
164	0.38	NN NN NNS	body temperature changes
157	0.36	JJ NN NNS	cultural soil types
124	0.29	NNP NNP NNP NNP NNP	Paramecium bursaria Chlorella virus AL1A
118	0.27	NNP NNP NNP NNP NNP NNP	South Georgia and South Sandwich Islands
91	0.21	NNP (NNP)	Tilapia (Cichlidae)
70	0.16	NN (NN)	anemia (disease)
62	0.14	NN NNP	fumonisin B2

Appendix E

UNSPSC Statistics

Table E.1: UNSPSC Labels Lengths Distribution.

Token count	Label count	Share of labels (%)
1	1 932	09.7679
2	6 328	31.9935
3	4 018	20.3145
4	3 191	16.1333
5	1 965	09.9348
6	1 124	05.6828
7	603	03.0487
8	306	01.5471
9	142	00.7179
10	86	00.4348
11	37	00.1871
12	22	00.1112
13	13	00.0657
14	8	00.0404
15	1	00.0051
16	2	00.0101
19	1	00.0051

Table E.2: UNSPSC Labels POS Tag Distribution.

POS Tag	Token count	Share of tokens (%)
NN	31 569	47.9772
NNS	18 659	28.3571
JJ	9 350	14.2097
CC	5 172	7.8602
IN	655	0.9954
DT	116	0.1763
RB	101	0.1535
PP$	58	0.0881
VBG	50	0.0760
NNP	23	0.0350
CD	14	0.0213
WRB	9	0.0137
TO	8	0.0122
VB	7	0.0106
JJR	4	0.0061
,	2	0.0030
PRP	2	0.0030
RP	1	0.0015

Table E.3: Top 20 UNSPSC POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
3 347	16.92	NN NNS	Sheet lifters
1 662	8.40	NN NN NNS	Slickline paraffin scrappers
1 511	7.64	NN NN	Play sand
1 046	5.29	NNS	Levels
1 009	5.10	JJ NNS	Brominated retardants
931	4.71	JJ NN NNS	Rotary position sensors
880	4.45	NN	Gelatin
514	2.60	NN NN NN NNS	Credit card service providers
465	2.35	NN NN NN	Filter cartridge adapter
395	2.00	NN NNS CC NNS	Manufacturing Components and Supplies
353	1.78	JJ NN	Chorionic gonadotropin
350	1.77	JJ NN NN NNS	Wireless network interface cards
214	1.08	NN JJ NN NNS	Zinc closed die forgings
213	1.08	NN CC NN NNS	Hose or pipe clamps
210	1.06	JJ NN NN	Intermodal cargo transport
168	0.85	JJ JJ NNS	Seismic magnetic systems
158	0.80	NN NN NNS CC NNS	Computer support parts or accessories
153	0.77	NN JJ NNS	Photo sensitive transistors
132	0.67	NN NN JJ NNS	Feng shui instructional materials
129	0.65	JJ NNS CC NNS	Dental tables or accessories

Appendix F

Yahoo! Directory Statistics

Table F.1: Yahoo Labels Lengths Distribution.

Token count	Label count	Share of labels (%)
1	432 092	52.1170
2	141 905	17.1159
3	206 726	24.9344
4	25 050	03.0214
5	5 722	00.6902
6	11 290	01.3617
7	3 405	00.4107
8	1 118	00.1348
9	743	00.0896
10	332	00.0400
11	517	00.0624
12	84	00.0101
13	45	00.0054
14	17	00.0021
15	17	00.0021
16	7	00.0008
17	6	00.0007
18	1	00.0001
19	1	00.0001
20	1	00.0001
21	1	00.0001
22	1	00.0001

Table F.2: Yahoo Labels POS Tag Distribution.

POS Tag	Token count	Share of tokens (%)
NN	610 235	38.5370
NNS	338 313	21.3648
NNP	270 046	17.0537
CC	188 653	11.9136
JJ	113 685	7.1793
(	12 899	0.8146
)	12 889	0.8140
,	11 009	0.6952
CD	8 696	0.5492
TO	8 667	0.5473
IN	4 870	0.3075
POS	1 740	0.1099
:	799	0.0505
NNPS	447	0.0282
DT	228	0.0144
JJR	185	0.0117
VB	77	0.0049
FW	21	0.0013
VBG	18	0.0011
JJS	7	0.0004
PRP	6	0.0004
PP$	5	0.0003
RB	4	0.0003
VBD	3	0.0002
RP	1	0.0001

Table F.3: Top 20 Yahoo POS Tag Patterns with Examples.

Label count	Share (%)	Pattern	Example
211 753	25.54	NN	Slowpitch
136 156	16.42	NNS	Sidecars
84 762	10.22	NN CC NN	Support and Assistance
52 316	6.31	NNP	Hitwise
38 395	4.63	JJ NN	High Jump
33 855	4.08	NNS CC NNS	Programs and Services
32 290	3.89	NN CC NNS	Love and Relationships
31 004	3.74	JJ	Vegetarian
29 835	3.60	NN NNS	Lesson Plans
24 848	3.00	NNP NNP	Rhode Island
24 208	2.92	JJ NNS	Used Vehicles
23 310	2.81	NNP NNP NNP	Neuwirth, Bebe
19 413	2.34	NN NN	Pliocene Epoch
9 645	1.16	NNS CC NN	Guides and Advice
8 459	1.02	NN TO NN	Seeking to Adopt
5 252	0.63	NNP NNP NNP NNP	VisiCom Laboratories, Inc.
4 365	0.53	NN CC NN NN	Air and Water Craft
3 773	0.46	NNP NNP NNP (CD)	Artaud, Antonin (1896-1948)
2 903	0.35	NN NN NNS	Con-Way Transportation Services
2 523	0.30	NNS , NNS , CC NNS	Crystals, Clocks, and Oscillators

Appendix G

Tokenizer Incremental Training

We report here the graphs of the incremental training of the OpenNLP tokenizer on our natural language metadata datasets. We performed the incremental training with the default settings of the tool, incrementing the size of the dataset by a thousand tokens a time and obtaining performance measure by applying 10-fold cross-validation procedure.

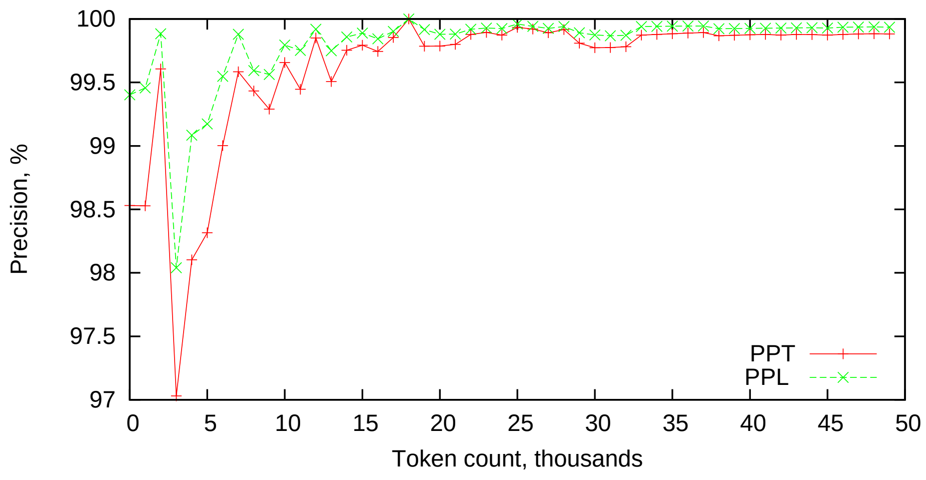


Figure G.1: Tokenizer Incremental Training on DMoz.

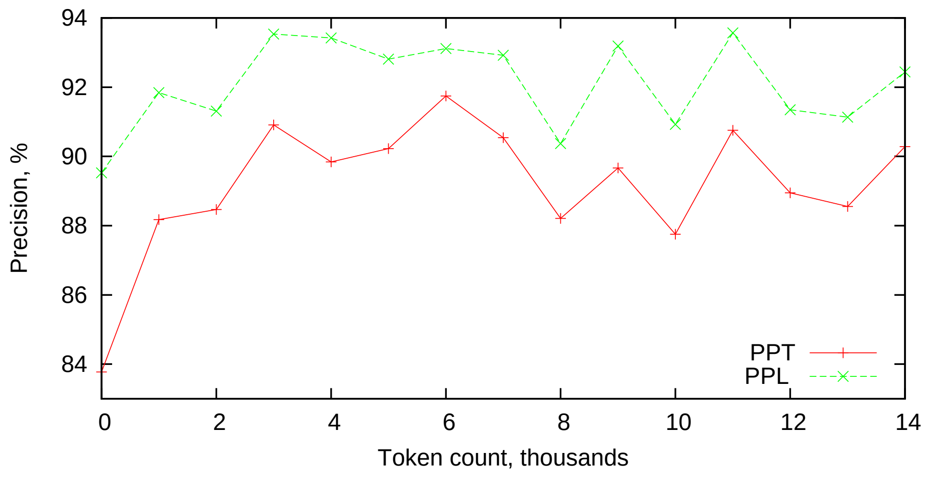


Figure G.2: Tokenizer Incremental Training on eCl@ss.

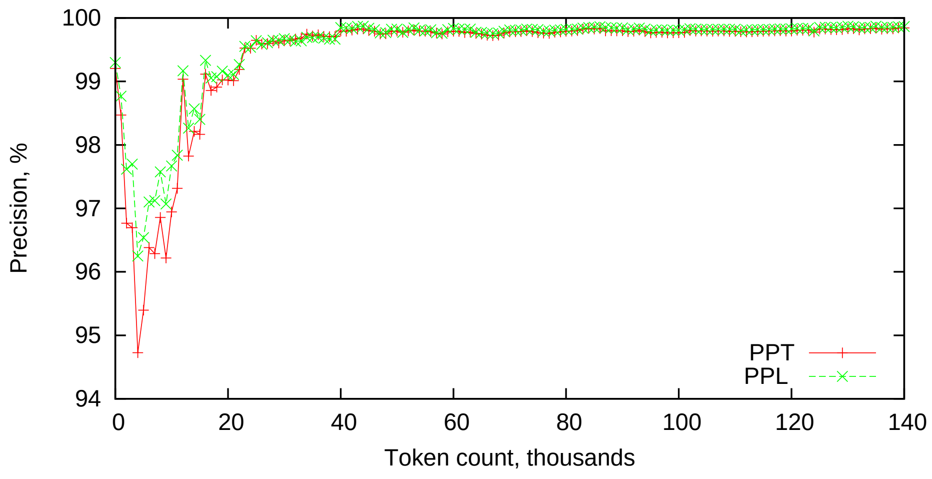


Figure G.3: Tokenizer Incremental Training on LCSH.

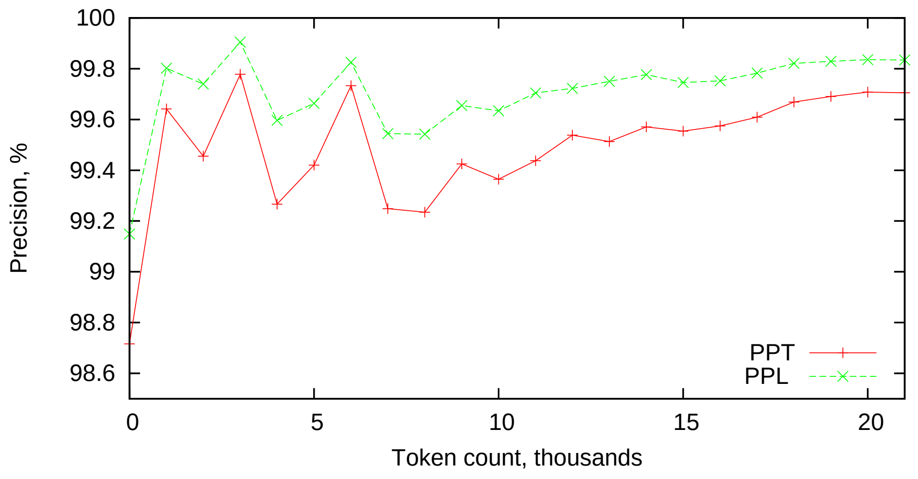


Figure G.4: Tokenizer Incremental Training on NALT.

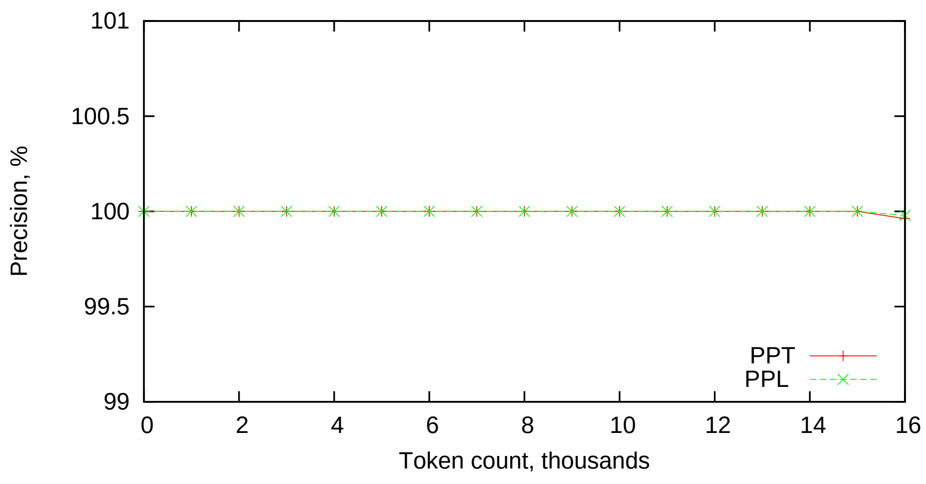


Figure G.5: Tokenizer Incremental Training on UNSPSC.

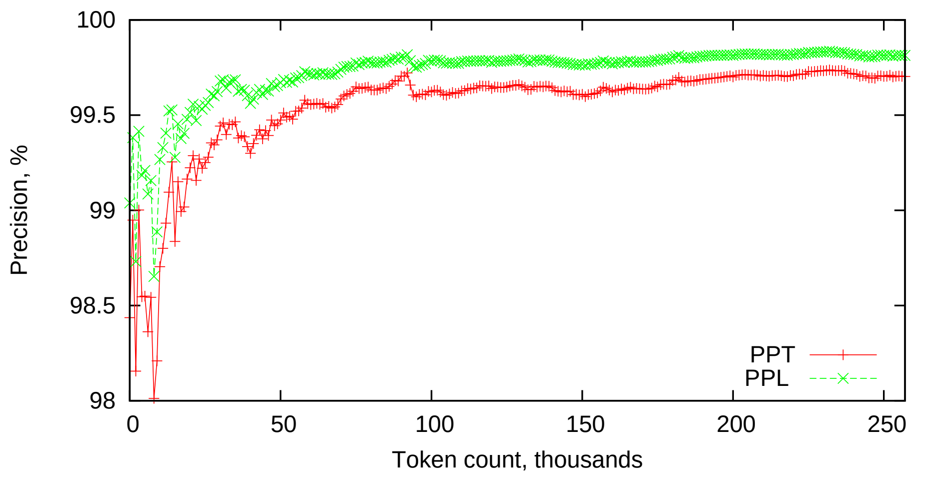


Figure G.6: Tokenizer Incremental Training on Yahoo.

Appendix H

POS Tagger Incremental Training

We report here the graphs of the incremental training of the OpenNLP POS tagger on our natural language metadata datasets. We performed the incremental training with the default settings of the tool, incrementing the size of the dataset by a thousand tokens a time and obtaining performance measure by applying 10-fold cross-validation procedure.

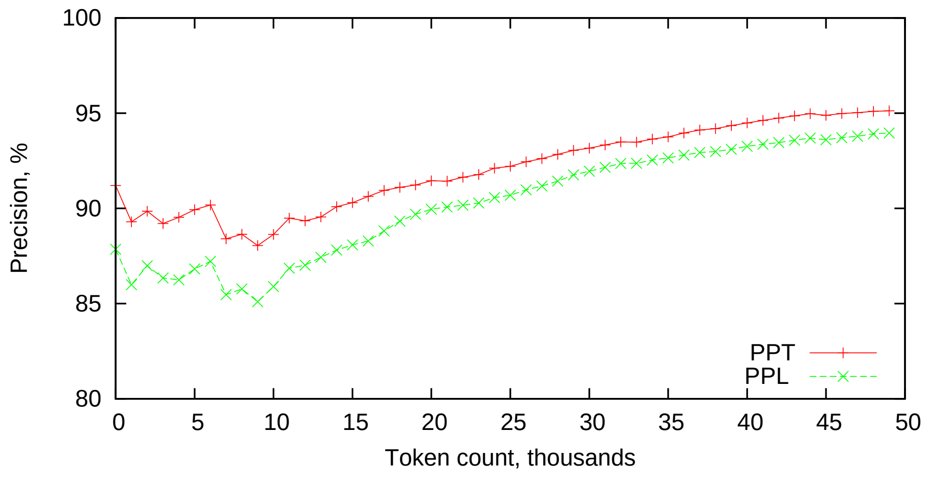


Figure H.1: POS Tagger Incremental Training on DMoz.

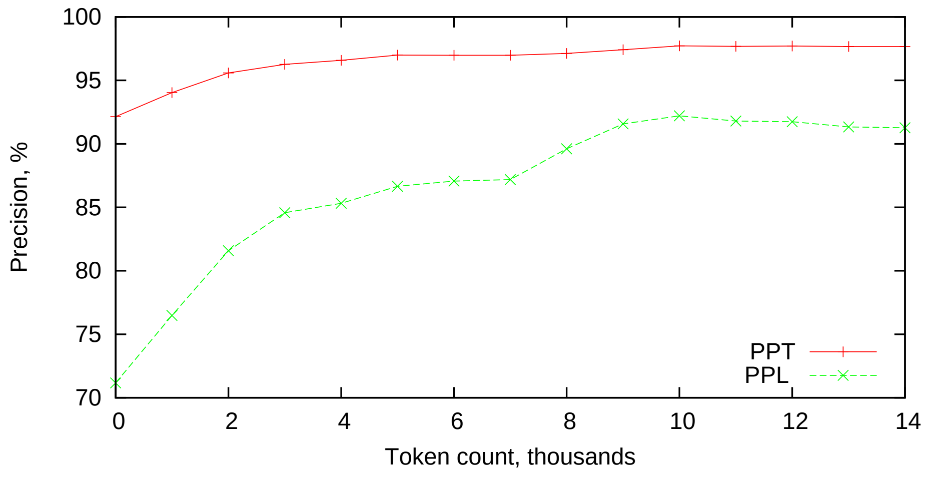


Figure H.2: POS Tagger Incremental Training on eCl@ss.

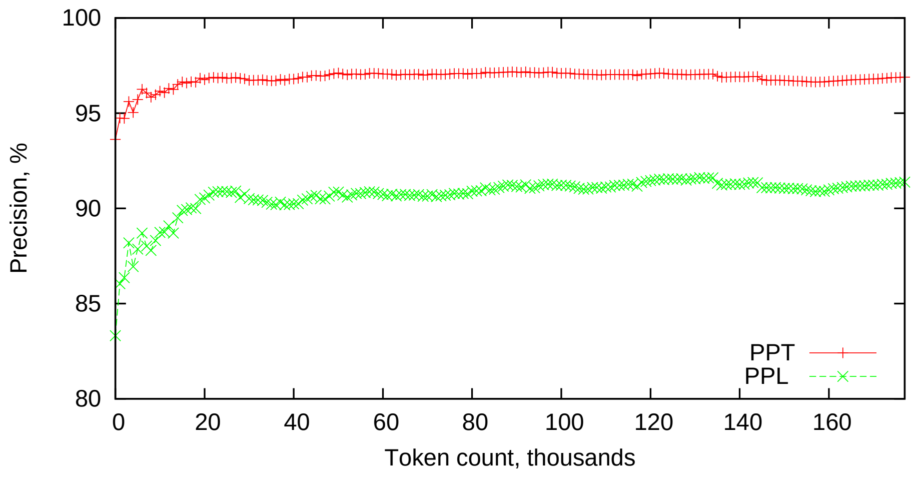


Figure H.3: POS Tagger Incremental Training on LCSH.

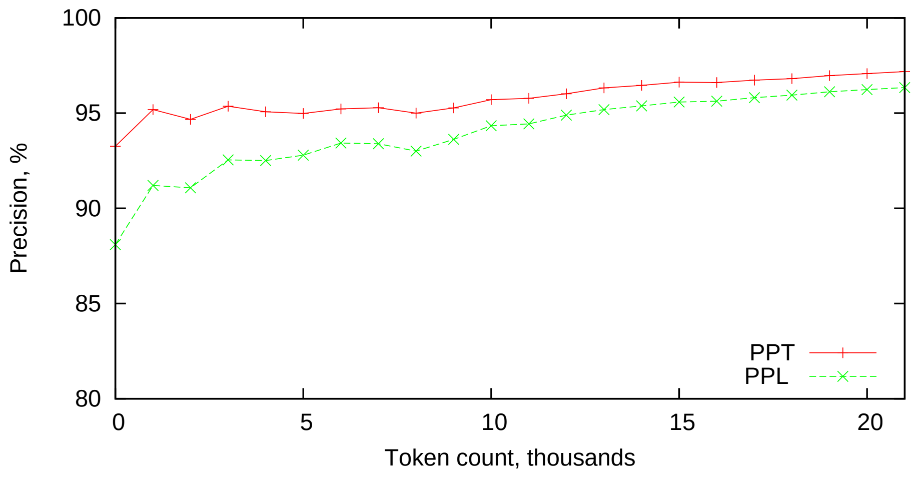


Figure H.4: POS Tagger Incremental Training on NALT.

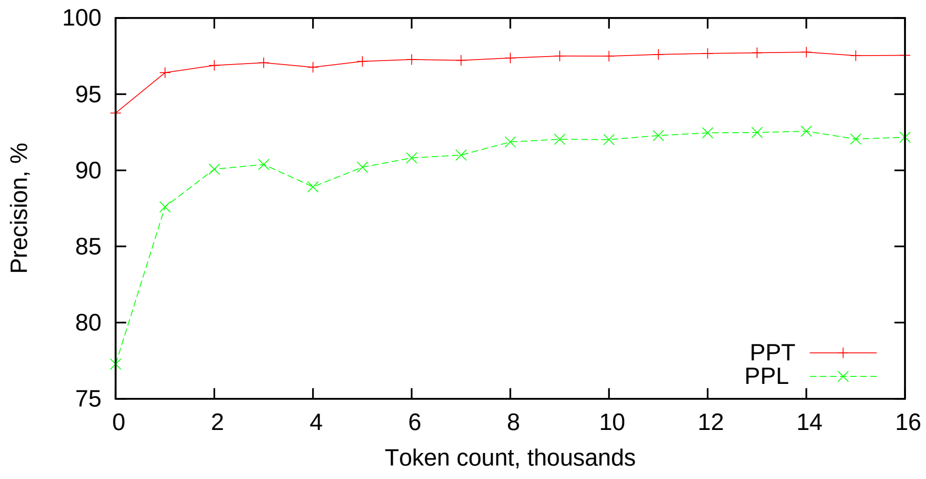


Figure H.5: POS Tagger Incremental Training on UNSPSC.

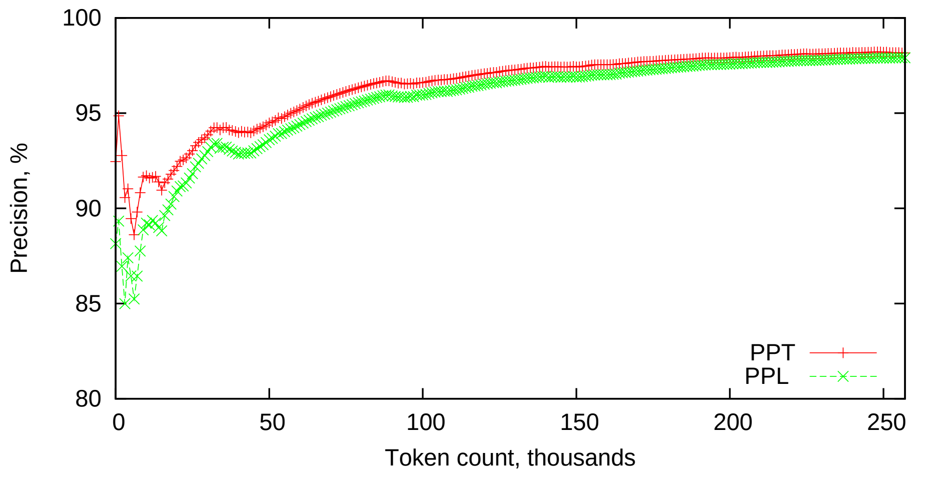


Figure H.6: POS Tagger Incremental Training on Yahoo.

Appendix I

Sense Summarization Evaluation

In the tables below in columns we present heuristics, in rows we present percent of answers falling into specific answer group. Percent is calculated as amount of answers falling into that specific group divided by the total amount of answers. The most important of all groups is the group of correct answers as it determines the order of the heuristics. A column full of zeros means the heuristic does not apply to this part of speech. We keep such columns to make all tables uniform and ease the comparison.

I.1 Associative Power

Table I.1: Noun Heuristics by Associative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
unknown	17	0	0	11	0	9
none	24	0	0	10	0	6
semicorrect	2	0	0	4	0	4
incorrect	22	0	0	14	0	17
> 1 selected sense	6	0	0	12	0	13
correct	32	0	0	53	0	56

Table I.2: Adjective Heuristics by Associative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
unknown	0	0	0	0	4	12
none	0	0	0	0	4	5
semicorrect	0	0	0	0	2	7
incorrect	0	0	0	0	22	19
> 1 selected sense	0	0	0	0	7	10
correct	0	0	0	0	63	55

Table I.3: Verb Heuristics by Associative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
unknown	17	0	6	12	0	7
none	21	0	8	15	0	2
semicorrect	4	0	5	5	0	10
incorrect	18	0	13	21	0	14
> 1 selected sense	10	0	10	11	0	22
correct	34	0	62	41	0	55

Table I.4: Adverb Heuristics by Associative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
unknown	0	4	0	0	0	11
none	0	7	0	0	0	3
semicorrect	0	12	0	0	0	16
incorrect	0	27	0	0	0	18
> 1 selected sense	0	20	0	0	0	21
correct	0	43	0	0	0	47

I.2 Discriminative Power

Table I.5: Noun Heuristics by Discriminative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
incorrect	28	0	0	21	0	52
unknown	10	0	0	11	0	7
correct	62	0	0	68	0	41

Table I.6: Adjective Heuristics by Discriminative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
incorrect	0	0	0	0	57	58
unknown	0	0	0	0	4	11
correct	0	0	0	0	39	32

Table I.7: Verb Heuristics by Discriminative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
incorrect	33	0	30	34	0	49
unknown	7	0	10	8	0	8
correct	60	0	60	58	0	43

Table I.8: Adverb Heuristics by Discriminative Power (%)

Answer Type	hyponym	derived	gloss	hypernym	similar_to	synset
incorrect	0	43	0	0	0	57
unknown	0	29	0	0	0	14
correct	0	29	0	0	0	29