Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Minna Tamper

Extraction of Entities and Concepts from Finnish Texts

Master's Thesis
Espoo, November 28, 2016

Supervisor: Professor Eero Hyvönen, Aalto University
Advisors: Eetu Mäkelä D.Sc. (Tech.)
Jouni Tuominen M.Sc.

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Minna Tamper		
Title:	Extraction of Entities and Concepts from Finnish Texts		
Date:	November 28, 2016	**Pages:**	vi+71
Major:	Software engineering	**Code:**	T-75
Supervisor:	Professor Eero Hyvönen		
Advisors:	Eetu Mäkelä D.Sc. (Tech.) Jouni Tuominen M.Sc.		

Keywords are used in many document databases to improve search. The process of assigning keywords from controlled vocabularies to a document is called subject indexing. If the controlled vocabulary used for indexing is an ontology, with semantic relations and descriptions of concepts, the process is also called semantic annotation.

In this thesis an automatic annotation tool was created to provide the documents with semantic annotations. The application links entities found from the texts to ontologies defined by the user. The application is highly configurable and can be used with different Finnish texts. The application was developed as a part of WarSampo and Semantic Finlex projects and tested using Kansa Taisteli magazine articles and consolidated legislation of Finnish legislation. The quality of the automatic annotation was evaluated by measuring precision and recall against existing manual annotations. The results showed that the quality of the input text, as well as the selection and configuration of the ontologies impacted the results.

Keywords:	Automatic annotation, Linked Open Data, named entity linking, ontologies
Language:	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Minna Tamper		
Työn nimi:	Entiteettien ja käsitteiden eristäminen suomenkielisistä teksteistä		
Päiväys:	28. marraskuuta 2016	**Sivumäärä:**	vi+71
Pääaine:	Ohjelmistotekniikka	**Koodi:**	T-75
Valvoja:	Professori Eero Hyvönen		
Ohjaajat:	Tekniikan tohtori Eetu Mäkelä Filosofian maisteri Jouni Tuominen		

Asiasanoja käytetään kuvailemaan dokumentteja ja parantamaan niiden löydettävyyttä. Asiasanoitusprosessissa asiasanat voidaan valita kontrolloidusta sanastosta. Näiden sanastojen tai ontologioiden käyttäminen mahdollistaa semanttisten kuvausten ja suhteiden hyödyntämisen. Tätä kutsutaan myös semanttiseksi annotoinniksi, ja sen avulla voidaan parantaa dokumenttien haettavuutta entisestään.

Tässä työssä kehitettiin sovellus semanttiseen annotointiin osana Sotasampo- ja Semanttinen Finlex -projekteja. Sovellus linkittää tekstistä löydettyjä tekstuaalisia entiteettejä käyttäjän valitsemiin ontologioihin. Sovellus on konfiguroitavissa erilaisten suomenkielisten tekstien asiasanoitukseen ja linkitykseen. Tässä työssä hyödynnettiin Kansa Taisteli -lehden artikkelien ja Semanttisen Finlexin ajantasaisia säädöksiä käyttötapauksina sovellukselle. Tuloksia arvioitiin vertaamalla niitä alkuperäiseen manuaaliseen annotaatioon käyttäen tarkkuus- ja saantimittauksia. Tuloksia tutkimalla havaittiin, että syötteen laatu sekä ontologioiden valinta ja konfigurointi vaikuttivat tuloksiin.

Asiasanat:	Automaattinen asiasanoitus, avoin linkitetty data, entiteettien linkitys, ontologiat
Kieli:	Englanti

Acknowledgements

I wish to thank professor Eero Hyvönen and tutors Jouni Tuominen and Eetu Mäkelä for encouraging and helpful guidance. I also wish to thank the members of the Semantic Computing Research Group (SeCo) for valuable comments. In addition, I wish to thank Matti Tolvanen from the Helsinki City Library for the informative interview.

I would also like to thank the Ministry of Education and Culture, the Association for Military History in Finland, and Bonnier Publications for providing the project with resources and for publishing the Kansa Taisteli magazine articles for public usage. In addition, I wish to thank Timo Hakala for providing the manual annotations for the Kansa Taisteli magazine articles.

Regarding the Semantic Finlex project, I would like to thank Aki Hietanen from the Ministry of Justice, Jari Linhala and Risto Talo from Edita Publishing Oy for providing the consolidated legislation and their annotations.

In addition, I wish to thank my friends, family, and relatives for supporting me through this process. In addition, thanks to my dog for being cheerful.

Thank You!

Espoo, November 28, 2016

Minna Tamper

Abbreviations and Acronyms

IE	Information Extraction
IR	Information Retrieval
KOKO	The Finnish Collaborative Holistic Ontology
LAS	Lexical Analysis Tool
LOD	Linked Open Data
NE	Named Entity
NED	Named Entity Disambiguation
NEL	Named Entity Linking
NEN	Named Entity Normalization
NER	Named Entity Recognition
NLP	Natural Language Processing
OCR	Optical Character Recognition
OWL	Web Ontology Language
POS	Part of Speech
RDF	Resource Description Framework
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language
SSHS	The Association for Military History in Finland
TF-IDF	Term Frequency - Inverse Document Frequency
Turtle	Terse RDF Triple Language
UI	User Interface
YSO	General Finnish ontology

Contents

Chapter 1

Introduction

Document databases are explored by users on a daily basis. The databases can be searched for different documents but it can be difficult to obtain satisfactory results easily. To improve the search results, search engines can utilize document metadata that contains descriptive keywords among other descriptive data about the document. [6, 19] For example, in computer systems in libraries, most books have different manually assigned natural language keywords to describe them. In addition to having information about the book (such as author, title, ISBN, language), the subject can be described using keywords. For example, a book about gardening might be described using words like *flora*, *gardening*, and *germination*. The assigning of keywords, that describe the topic or subject, and connecting these keywords with the document is called subject indexing. Ideally these keywords are picked from a controlled vocabulary, such as a thesaurus or ontology. Traditionally indexing has been performed by librarians for example while cataloging documents. The keywords can aid the user of the library computer system to find the books that he is looking for. [17, 25, 56, 93]

One way to implement subject indexing is by using Semantic Web technologies. Semantic Web[1] is a extension of the Web with a set of tools and standards. It provides a framework for sharing and reusing data across application, organization, and domain boundaries. In Semantic Web the method of publishing structured data is called Linked Data. The central idea of Linked Data is to make it possible to interlink data and make it more useful through semantic queries. [11, 12] In terms of subject indexing, it would mean that the relevant keywords would be identified from each document and linked to existing controlled vocabularies, giving the keywords semantic meanings. In the context of Semantic Web this can be also called annotating.

[1] https://www.w3.org/standards/semanticweb/

Manually annotating or subject indexing each document is, however, laborious, costly, and time consuming work. [17, 56] On the other hand, this is not a simple task for the computer either. Identification of terms from texts by extracting words can be inefficient and inaccurate. One word can mean many things and when evaluating each term in a text, it might be difficult to recognize whether the term references for example a person's name or a place. Sometimes the term may consist of multiple words; it can be difficult to identify a term if different chunks of words form a term separately and together. For example, movie titles, such as *Indiana Jones and the Last Crusade*, can be hard to identify from texts. The title can be misidentified and only chunks *Indiana Jones* and *Crusade* may be recognized instead of the full name of the movie. These tasks would require more dedicated algorithms and possibly domain specific information extraction (IE) methods to identify terms with satisfactory precision. Information extraction methods specialize in mining structured information from unstructured or semi-structured text in natural-language documents [20]. The information can be used to classify, index, or link documents to other related information. In order to get relevant information, the extraction would require a more sophisticated Natural Language Processing (NLP) approach.

In this thesis the purpose is to create a generic tool for automatic annotation as a part of WarSampo[2] and Semantic Finlex[3] projects in Semantic Computing Research Group (SeCo)[4]. The tool needs to be able to annotate Finnish documents and further more will be tested with two cases: Kansa Taisteli magazine articles and the consolidated legislation of Semantic Finlex. Kansa Taisteli magazine articles have a perspective in the WarSampo portal. The goal of the WarSampo portal is to model the Second World War in Finland as Linked Open Data (LOD). [42] Kansa Taisteli is a magazine published by Sanoma Ltd and Sotamuisto association between 1957 and 1986. [33] The magazine articles cover memoirs of WW2 from the point of view of Finnish military personnel and civilians. Semantic Finlex, on the other hand, is a service that offers Finnish legislation and case law as Linked Open Data. The results of the annotation process for both projects are published as a part of Linked Data Finland[5] service.

The goal of this thesis is to study how to produce automatically solid annotations for different Finnish texts in comparison to manually annotated texts. The goal can be divided into research questions that are enumerated below:

[2]http://seco.cs.aalto.fi/projects/sotasampo/en/
[3]http://seco.cs.aalto.fi/projects/lawlod/en/
[4]http://seco.cs.aalto.fi
[5]http://www.ldf.fi

1. How to build a generic application for automatic annotation that can be configured for different use cases?
2. How to utilize ontologies for Finnish texts to get relevant annotations?
3. How much disambiguation is needed and how to do it? What kind of problems are encountered when trying to solve ambiguity issues?
4. In regards to OCR, what is the impact of OCR'd text to the results? Is it possible to minimize the impact of errors?

The thesis is structured in the following manner. Chapter 2 describes the background and the designed model of annotation. In chapter 3, based on the model of annotation, the implementation and technology choices are discussed in more detail. Chapters 4 and 5 introduce the two use cases for the automatic annotation tool and configurations relating to the usage of the tool. The results of applying the tool to these two use cases are presented and analyzed in chapter 6. Lastly the conclusions and future work is discussed in chapter 7.

Chapter 2

Model of Annotation

Annotation or more precisely semantic annotation can be used to enrich the document metadata. There are different approaches into semantically annotating documents. The annotation process can be performed manually, semi-automatically, or fully automatically. Manual annotation is done by one or more people. Manual annotation tools allow users to add annotation to web pages or to other resources. In semi-automatic annotation users are given automatic suggestions and they can choose the fitting ones. In fully automatic annotation tools, annotations are generated by applications and users have no control over them. [78]

Traditionally manual annotation and subject indexing tasks require that a human has to read or scan the document and then produce annotations or select keywords that describe the aboutness of a text. A generic automatic tool for annotating and subject indexing could help to reduce time and money spent on the tasks. In addition, due to the increase in growth in available electronic documents the amount of unindexed texts has grown. Automatic document indexing can be helpful in digital libraries to unload the amount of unindexed texts. [17, 56, 106]

In order to build an automatic annotation tool that can process Finnish texts, different methods are required. This chapter describes semantic annotation, automatic annotation, and related topics in more detail. Afterwards, an abstract model of automatic annotation is introduced and the process and methods used in it are explored further.

2.1 Semantic Annotation

The goal of annotation is to find fitting descriptions for given resources. A resource in this context can be a text (for example article, magazine, book)

or a non-textual object such as a map or an image. The general purpose of annotating documents, in addition to improving search, is to enable new applications including highlighting and categorization, generation of more advanced metadata, and effortless traversal between unstructured text and available relevant knowledge. For example, knowledge acquisition can be performed based on extraction of more complex dependencies – analysis of relationships between entities, event and situation descriptions. [53] An example annotation would identify the term *Helsinki* in a text as a proper noun. The annotations that consists of terms can also be used to summarize, to navigate, and to visualize document contents.

Semantic annotation, on the other hand, uses controlled vocabularies, such as ontologies, and adds meaning for the identified term. To be more precise; it is about assigning to the resources links to their semantic descriptions. [53] An example semantic annotation would relate the term *Helsinki* to an ontology, identifying it as the capital city of Finland.

Semantic annotation can be divided into multiple subareas such as subject indexing or named entity recognition (NER) which are main focuses of this thesis. The goal of subject indexing [93] is to describe the given document using pregnant keywords. In automated annotation systems the keywords can be selected from the text using information extraction methodologies. Information extraction applications usually rely on two context specific resources: a dictionary of extraction patterns and a controlled vocabulary that contains semantic information and relations (semantic lexicon). The extraction patterns may be constructed manually or created automatically. Typically systems that automatically create extraction patterns, use special training resources, such as texts annotated with context specific tags or manually defined keywords, frames, or object recognizers. [85]

2.1.1 Ontologies

A central component of semantic annotation is the linking of extracted textual entities to given ontologies. In computer science, an ontology is a model of entities and their relationships in a domain of knowledge or practices. It is represented in a declarative formalism. Ontologies consist of concepts with an explicitly defined semantics and can be used as values in metadata. [31, 92]

In practice, ontologies are networks of concepts that have different kinds of relationships between them, such as meronymy, and hyponym. A meronym is a part of a whole where as a holonym is an opposite of a meronym. For example a the term page is a meronym of a book and car is a holonym for the term tire. A hyponym is a subordinate term whereas a hypernym is a superordinate term in contrast. As an example the color red is a superordi-

nate term or hypernym to the terms crimson, maroon and scarlet. Crimson, maroon and scarlet are subordinates or hyponyms of color red. [15, 46] These relationships can be used to improve search capabilities of applications. For example, the winner of the Semantic Web Challenge 2006, the E-Culture demonstrator, uses existing ontologies for annotation and search of a collection of resources. In the demonstrator, a query for *flowers* not only returns documents about flowers but also documents annotated with *roses*, since the concept *flowers* is defined as a hypernym of *roses*. [38]

In the domain of the Semantic Web, ontologies are described in RDF format. RDF (Resource Description Framework) represents information as graphs. A graph is a set of subject-predicate-object triples, where the elements are Internationalized Resource Identifiers (IRIs), blank nodes, or data typed literals. These triples are used to express descriptions of resources. [22] For example, the subject *company* can be described as a synonym of the object *firm* by using a corresponding keyword defined by a vocabulary such as RDFS, SKOS, or OWL. These vocabularies or ontology languages extend the basic RDF vocabulary. RDFS or RDF Schema provides a data-modeling vocabulary for RDF data by adding classes, instances, hierarchies, range and domain restrictions, and introducing reasoning support. [14] SKOS (Simple Knowledge Organization System) provides a model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, and folksonomies [45]. OWL (Web Ontology Language) is a computational logic-based language for the Semantic Web that extends RDF by adding more possibilities for reasoning [36].

2.1.2 Applications for Semantic Annotation

There are a number of applications that can utilize semantic annotations. On the web, the two principal use cases are searching and browsing. Linked data facilitates semantic search and semantic browsing. Searching, or moreover semantic searching, utilizes additional semantic information in order to improve search results. Browsing means that the user follows associative links to find information. In addition to search and browsing applications, annotations can be utilized, for example, in visualization and summarization. [41]

In semantic searching, the goal of utilizing semantic information is to understand the user's information query more deeply in a context to determine the relevance of search objects more accurately. The additional external information may concern the ontological properties of data such as the end-user's preferences, profile, or the spatio-temporal context where searching is

performed. [41]

The idea of semantic browsing is to support browsing documents, for example, through associative recommendation links. These links are created based on the underlying linked metadata and ontologies. Semantic recommendations are often based on semantic criteria that differ from the original search terms but at the same time are complementary to them. [41] For example, one use case could be recommending links to products in a online shop based on behavioral of the users of the same age and sex.

Faceted search is a combination of searching and browsing as the search is based on selecting facets that correspond to links of concepts. Contrary to semantic browsing, faceted search does not aim to expand the result set but rather to constrain it by filtering the results with user's selections. It has a more visualized approach to searching; faceted search is useful when the user cannot formulate his information needs in accurate terms. Faceted search allows the user to look and learn about an area of interest whereas in Google-like keyword search interface, it is usually preferred if the user knows exactly what he is looking for and is capable of expressing her information need accurately. [26, 41]

Visualization of annotations can provide useful graphical summarizations of a document or a text. By using visualizations and summarizations of the document contents the readers can learn about the document's aboutness. Otherwise, the readers may be forced to scroll through many pages to identify the information they seek. [55] Tag clouds are visual representations that summarize or describe document contents. [35, 55] The input data that is often used in tag clouds is community based tags. The tags are unstructured annotations by authors or readers that describe the document. In addition, usually used search engine query terms, word frequencies within documents, and pre-existing category labels are visualized using tag clouds. [35] Tag cloud interfaces have an advantage in presenting descriptive information and in reducing user frustration. [55]

Tag clouds can utilize semantics. In the earlier example *Helsinki* was recognized as the *capital city of Finland*. Now, if a tag cloud was created about a document that mentions Helsinki, it would be possible to enrich it semantically. Meaning that we could add terms such as *Finland* or *City* into the tag cloud.

2.2 Automatic Annotation

Due to the monotonous and costly nature of manual annotation, it is important to design annotation tools where the annotation process can be per-

formed as swiftly as possible. The entrance barrier for annotation can be lowered with a generic annotation tool because it would reduce development costs and preparatory work. [106]

Automatic annotation tools can be, for example, based on machine learning or statistical methods. [83, 106] These tools can have limitations regarding adaptability, reconfigurability, and reusability. It can be hard for the tools to accomplish annotation tasks outside of the typical domain of the tool. Reusability issues may arise when there is a need for manual intervention to add newly annotated documents into the iteration. [106]

One example of an automatic annotation system is the DBpedia Spotlight service[1]. DBpedia Spotlight is an open source service that recognizes DBpedia resources in natural language text. It is a solution for linking unstructured information sources to the Linked Open Data cloud. DBpedia Spotlight recognizes phrases, people's names and terms that have been mentioned (for example "Michael Jordan"), and matches them to unique identifiers (for example *http://dbpedia.org/resource/Michael_I._Jordan*, the machine learning professor or *http://dbpedia.org/resource/Michael_Jordan* the basketball player). Currently this project's focus is on English language. [23]

In Figure 2.1 depicts an example of a chunk of text annotated with DBpedia Spotlight. The demo available for public use[2]. In it a chunk of text is given to the application and by pressing the Annotate button it annotates the text. The semantic annotations are visible in the figure as links. In this case the annotations have been limited to certain types that are shown below the text field. These types have been selected with the select types button, that opens a selection of types to choose for the users.

The DBpedia Spotlight application links text only to its own DBpedia ontology. In a generic automatic annotation tool the text can ideally be linked to multiple different ontologies. In addition to linking documents, the application needs to be able to select best describing keywords for a document. This is not a simple task and needs natural language processing methods in addition to linking text correctly to ontologies.

2.3 The Process of Automatic Annotation

The automatic annotation process of documents consists of several phases. In order for it to succeed, the correct textual entities need to be extracted from the text. The task requires the usage of computational linguistic methods.

[1]https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/Introduction
[2]https://dbpedia-spotlight.github.io/demo/

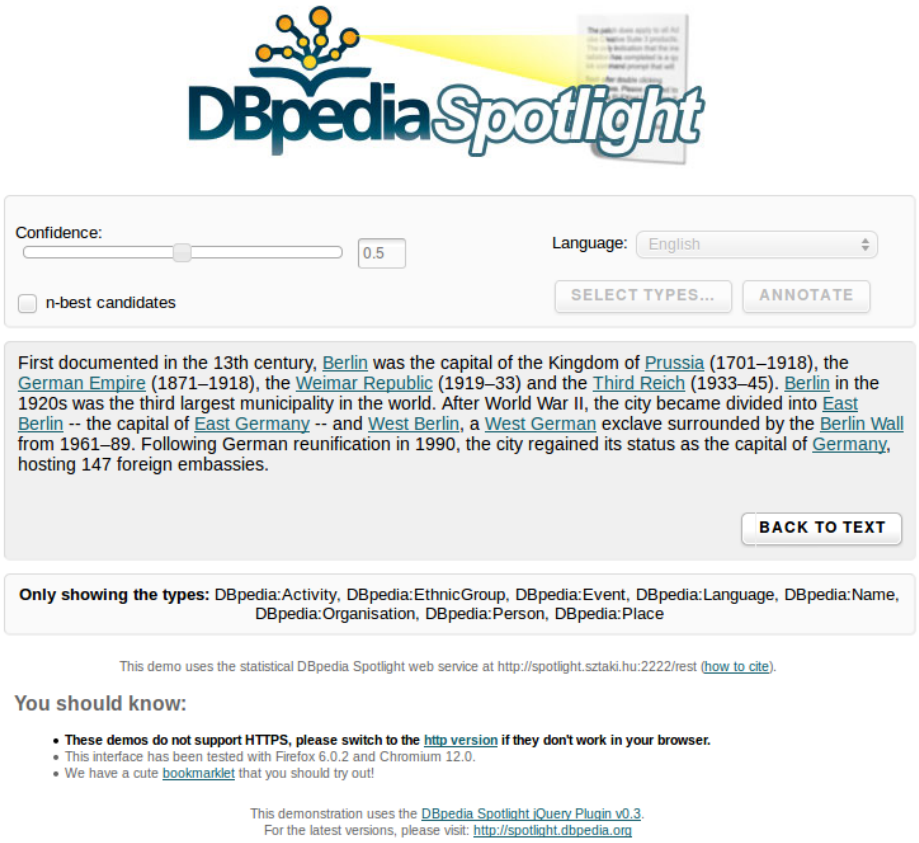

Figure 2.1: Example of DBpedia Spotlight demo.

In the case studies of this thesis, the methods are selected based on the characteristics of the Finnish language.

After the extraction of textual entities, these entities are linked to different ontologies. The semantic relations of the ontologies can be utilized to recognize relationships between textual entities during the linking phase. Once the linking is performed, the entities form a group of keyword candidates. The relevancy of each candidate is estimated and the candidates are ranked from best to worst. From this group the most relevant ones are selected as keywords. The strategy for selecting keywords is left for the user to decide.

In this thesis, model for automatic annotation is built from these components. In the following sections the topics are explored in more detail. In addition, the model of annotation is presented.

2.3.1 Computational Linguistic Methods

In the domain of Finnish documents, the language itself creates a challenge for annotation process. Finnish language is highly inflected and it expresses meaning through morphological affixation (agglutinativity). In order to find relevant connections between terms of a document and ontologies, the base form of each word in a document needs to be identified. In languages such as English plural and possessive relations, grammatical cases, and verb tenses and aspects, are expressed using syntactic structures. However, in highly inflected languages, these are characteristically represented using case endings. Another typical feature of inflected languages is the usage of compound words. [89, 93]

Stemming is a linguistic method that strives to find a word's root form. It can be described as a crude heuristic process that chops off the ends of words in hopes of achieving the root correctly most of the time, and often includes the removal of derivational affixes. [63] For these reasons modern stemmers, such as Snowball stemmer[3] [54], work poorly with inflected languages [3, 51, 89, 93]. For example, in English stemming of the word *shoes* produces the word root *shoe* which is also the word's singular form. However, in Finnish, the corresponding word root *kenk* cannot be derived from the plural form *kengät*, and stemming would result in the stem *keng* instead. The connection to the original word is lost.

An alternative for stemming is to use lemmatization. In lemmatization the goal is to find a lemma (or the basic form) from the inflected version of the word. Lemmatization can utilize vocabularies and morphological analysis of words to achieve its goal. For these reasons it can be considered to be a more sophisticated way to attain the base form for a word than stemming. [63, 64]

Morphological analysis is a part of lemmatization but can be also used as a standalone method. One of its goals is to find the base form and to identify the inflection form of a word. The analysis includes disambiguation that is a selection of the most plausible sequence of lexemes [4, 105]. A lexeme is a basic lexical unit of a language that includes a lemma and inflected forms [64]. In Finnish language, because of the complexity of the morphological structure of words, a specially designed morphological analyzer, such as FinTwol[4] [61], Omorfi[5] [81], or FDG[6] [94], is needed to identify lemmas embedded in Finnish words [60].

In order to successfully link the textual entities to ontologies, the tool

[3]http://snowball.tartarus.org/
[4]http://www2.lingsoft.fi/cgi-bin/fintwol
[5]https://github.com/flammie/omorfi
[6]http://www.connexor.com

needs lemmatization and morphological analysis methods. These methods can be used to differentiate between different textual entities. They can be also used to identify part of speech class and transform the textual entities into same forms as they are in the selected ontology. For example in KOKO (The Finnish Collaborative Holistic Ontology) ontology the concepts are nouns mostly in plural form [88]. In order to successfully link textual entities, they must be nouns and transformed into the same form as in the ontology.

2.3.2 Named Entity Linking

In natural language processing, named entity linking (NEL) is the task of determining the identity of named entities mentioned in a text. Named entity recognition (NER), named entity disambiguation (NED), and named entity normalization (NEN) [32, 47, 52] are a part of the same task but focus on different issues regarding named entity identification. NER is a sub-task of information extraction. Its purpose is to locate and classify elements in text into pre-defined categories such as the names of persons, organizations, locations, and other named entities (NEs). [10, 71, 97] NER is a central component in many natural language processing (NLP) applications, such as question answering, summarization, and machine translation. The current focus of NER is to develop language-independent systems that learn and utilize statistical models. [76]

NER alone is not enough to identify the identity of the entity extracted from a text. In addition to NER, named entity disambiguation is a crucial prerequisite for successful information extraction and information retrieval for example. [16, 21, 104] While NER identifies the occurrence or mention of a named entity in text, named entity disambiguation identifies which specific entity it is. The combination of NER and NED can also be called named entity normalization (NEN). [18] It is a task that identifies named entities from the text and normalizes them to the concepts they refer to. The normalization means that it aims to solve rising ambiguity and synonymy issues. The task includes NED to solve ambiguity issues but also handling synonyms by connecting them. [47, 52]

Named entity linking is the task of linking found named entity mentions to entries in a structured knowledge base. NEL computes direct references for example to people and to places instead of potentially ambiguous or redundant character strings. It can be used to aid search. For example if information about a certain named entity is queried, the results could contain facts about the entity in addition to pages that talk about it. [16, 32]

In this thesis, the named entity linking component needs to link the tex-

tual entities into selected ontologies. The ontologies need to be easy for the user to add, and to configure the tool to use them correctly. Regarding disambiguation, the tool needs to be able to decide the best match for the textual entity. In different ontologies the same entity may have a different meaning. Therefore the user needs to define priorities for using different ontologies.

2.3.3 Ranking of Keyword Candidates

Before assigning keywords to a document, the keywords need to be ranked and the most relevant keywords need to be selected. In this thesis it will be referred to candidate ranking but it can also be called candidate filtering [66, 102]. In order to evaluate keyword relevancy, different kinds of weighting schemes can be used to calculate a weight to rank the candidates. The ranking can be done using manually assigned heuristic rules [8, 44, 102], unsupervised statistical methods [58, 67, 102], or supervised machine learning [29, 65, 99, 102]. These term relevancy weighting schemes can estimate term relevancy based on the number of times the term occurs in a document [8, 29, 99, 102], the location of the term in the text [29, 74, 99, 102], the length of the term [8, 66, 99, 102], or based on the semantic relations and properties of the term in the thesaurus hierarchy [39, 65, 66, 102].

In order to facilitate candidate ranking, the tool needs to have a possibility to use different ranking strategies. It should be possible to add these strategies into the tool easily. And the component should produce relevancy score that can be ranked easily, creating a matrix where the terms and their scores are mapped for later usage for each document.

2.3.4 Architecture

The automatic annotation tool developed in this thesis has been designed by taking into consideration the use cases and the background of the field. In order to annotate Finnish texts, it requires specific tools designed for Finnish language. In addition to the NLP approach, it needs to identify relevant concepts and named entities and link them to controlled vocabularies with matching terms. Based on both of the requirements mentioned, a general model for annotation has been created. The model is introduced in Figure 2.2.

As shown in Figure 2.2 the automatic annotation application has a linguistic preprocessing component, a candidate extraction component, and a candidate ranking component. These components process the input in the given order and produce an output. The linguistic analysis requires input data in a text format. In Kansa Taisteli case study the input is a set of

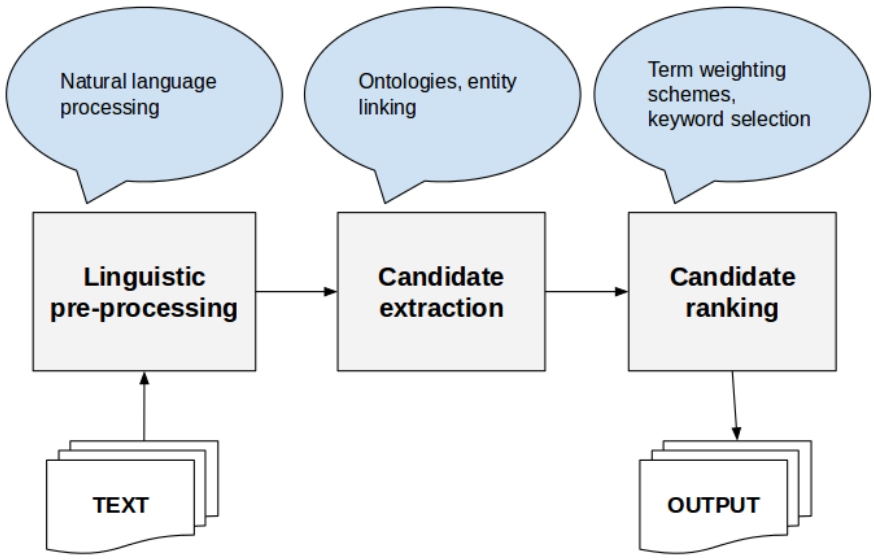

Figure 2.2: Model of annotation.

magazine articles that are in PDF format. The articles need to go through an OCR process to get the required input text for the application. In the Semantic Finlex case study, the input data is in HTML format. The application must extract text from an HTML website where each page is identified using a URL.

Once the application has the data in a text format, the next phase is to process the input using linguistic methods. In this case what would be needed is to base form the text and to exclude some parts of speech such as pronouns and conjunctions from the text to avoid clutter. For this component, the language and part of speech of the words needs to be identified, so that it is possible to extract later on keywords and named entities efficiently.

After the linguistic preprocessing components have processed the text, it is ready to be linked into internal and external vocabularies in phase two. The candidate extraction component, given external vocabularies, can link the text with other resources and retrieve data such as the term's synonyms. In addition to extraction of named entities, it is possible to identify keyword candidates from the text to create more accurate descriptions of the documents.

In the last phase the application has the linked data and extracted keyword candidates. In it the data and the text is analyzed to see which keyword candidates are useful in describing the content and to function as keywords

for the given document. For this purpose, term importance and different weighting schemes are required. Using weights, the extracted candidates can be ranked and the top candidates picked.

Finally, after the candidate selection the application needs to produce an output. The output contains the results in a required format. The application needs to be able to serialize the results into different formats for further usage. It should be possible for the user to configure the output and input formats.

Chapter 3

Implementation

In order to implement the model of automatic annotation, introduced in the previous chapter, different tools and implementation languages can be used. For this thesis, I chose Python programming language as the implementation language. Python is a general-purpose, high-level programming language, and like many other programming languages it is free, even for commercial purposes, and it is platform independent [86, 100]. Python's modularity is beneficial for this project; there are many extension modules that can be used with Python that also aid in the development of the annotator application.

In addition to selecting implementation language, the tools need to be picked to meet the requirement of having the ability to annotate multiple different kinds of document and texts. Therefore, in order to meet the diverse requirements presented in the previous chapter, configurability is introduced as a criterion for choosing tools and modules. In this chapter the implementation is discussed for each part of the model of annotation. Starting with the input and output data, followed by the linguistic preprocessing. After this, tools and technologies to implement candidate extraction are introduced and finally the candidate ranking technologies are discussed.

3.1 Input and Output Datasets

The application needs by default its input in text format. The text should be natural language text that can be processed using linguistic tools and methods. However, in order to create a general purpose tool, it could be more useful if the application could also extract natural language texts from different data formats, such as a news article from a HTML page or a webshops's product description in RDF or XML format. In addition, in some cases some documents can already contain metadata that can be used to

aid in the annotation process. For example, the ontologies could be selected based on the metadata.

For the extraction of text and document metadata, Python offers a variety of modules such as BeautifulSoup[1] and SPARQLWrapper[2]. BeautifulSoup is a Python library for extracting data from HTML and XML files. It creates a parse tree of the input and simplifies the text extraction for the user, making it possible to target any parts of the input data. [84] SPARQLWrapper is a Python module that can be used to extract text or document metadata from data in RDF format. [27]

Finally, once the application has been executed, the results must be generated in a user-specified format. For this purpose, a specialized RDF serialization module is needed for the application. One such module is RDFLib[3]. It can serialize RDF into formats, such as Turtle[4] (Terse RDF Triple Language) [9], and write the results into a file.

3.2 Linguistic Preprosessing

The first phase for the extracted input text is linguistic preprosessing. In order to extract terms and do subject indexing, the text needs to be analyzed. In this phase the extracted text is analyzed using linguistic methods. For this purpose, the linguistic preprocessing could use methods such as morphological analysis and lemmatization. The words need to be in a basic form in order to be able to analyze, link to ontologies, and later on calculate weights for the terms.

One such tool is the Lexical Analysis Services (LAS). LAS uses existing linguistic tools such as Omorfi that have support Finnish and many other languages. [69] It consists of language recognition (for 95 languages)[5], lemmatization (for 20 languages)[6], morphological analysis (for 14 languages)[7], inflected form generation (for 14 languages)[8], and hyphenation (for 46 languages)[9]. All functionalities are available as web services, supporting both the HTTP and WebSocket protocols. All services are additionally CORS-enabled and return results in JSON for easy integration into HTML5 web

[1] https://www.crummy.com/software/BeautifulSoup/
[2] https://rdflib.github.io/sparqlwrapper/
[3] https://rdflib.readthedocs.io/en/stable/
[4] https://www.w3.org/TR/turtle/
[5] http://demo.seco.tkk.fi/las/#language_recognition
[6] http://demo.seco.tkk.fi/las/#lemmatization
[7] http://demo.seco.tkk.fi/las/#morphological_analysis
[8] http://demo.seco.tkk.fi/las/#form_generation
[9] http://demo.seco.tkk.fi/las/#hyphenation

applications. [69]

The diversity of functionalities makes LAS a good fit for the needs of the automatic annotation tool. The morphological analysis can produce a sound analysis that can be used by the automatic annotation tool. For example, if a Finnish language sentence such as 'Albertin koira haukkuu tuntemattomille' (Albert's dog barks at strangers, in English) is given for the tool, it produces the analysis results in JSON format for the sentence. The results contain information about each word such as the word's basic form (for example in Finnish haukkua, in English bark), original form (in Finnish haukkuu, in English barks), tense (present tense), and part of speech (verb).

3.3 Candidate Extraction

After the linguistic analysis phase, the results of the analysis can be used to extract suitable keyword candidates from the text. These candidates can be identified by using controlled vocabularies. For this purpose, a tool to do matching and entity linking is required. To do this a tool called ARPA can be used.

ARPA is a configurable automatic annotation tool that uses LAS (Lexical Analysis Services), SPARQL (SPARQL Protocol and RDF Query Language [96]), and ontologies to identify entities from a text document, and in return gives suggestions for annotating texts [69]. SPARQL is a language and a protocol to query and manipulate RDF graph content on the Web or in an RDF store [96]. When a chunk of text is sent to the ARPA service, it tries to interpret the text using LAS. This is where the ARPA configurations come into play. The configurations are applied to LAS and it is used accordingly. Eventually it produces a JSON output that has the complete results for the lexical analysis. From there ARPA takes the results and queries for matches with the given SPARQL query from a SPARQL endpoint. Finally, after querying is finished, ARPA returns the results in JSON output format. [69]

Unlike DBpedia Spotlight, ARPA is a more fitting tool for implementing automatic annotation in our case studies. DBpedia Spotlight lacks language support for Finnish and has less configuration options. ARPA is highly configurable and has support for Finnish language, as it is presented in the Figure 3.1. In the figure it is shown that the ARPA service configuration is comprised of a text field and a series of controls by which it is possible to change parameters of the lexical analysis process, as well as to specify target ontology for fetching candidate annotations. Thus, in order to use ARPA, a named configuration has to be created for each ontology. Afterwards this configuration can be used as a web service to query annotations for given

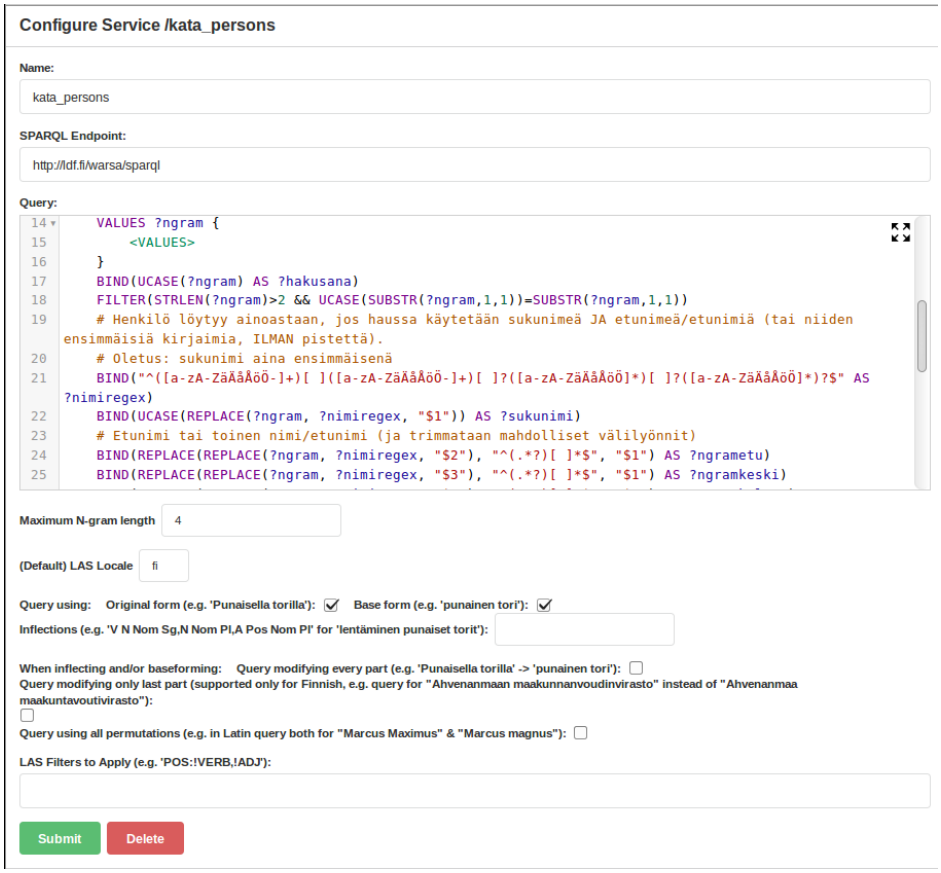

Figure 3.1: Example of an ARPA configuration.

text.

As shown in Tables 3.1, 3.2, and 3.3 ARPA configurations can be divided into three groups: LAS options, candidate filtering options, and query options. The LAS specific options, listed in Table 3.1, improve linking of entities by utilizing the linguistic tools of LAS. In order to improve the possibility of finding a matching entity for the queried text, maximum n-gram length can be specified. An n-gram is a sequence of n words: a 2-gram (bigram) is a two-word sequence of words like “European Union” and a 3-gram (trigram) is a three-word sequence of words. [48] In addition to n-gram length, the user can define the analysis depth from levels 0 to 2. The analysis depth feature can be used to define the precision used in the disambiguation of words. However, depending on the level of disambiguation, the processing can be laborious for the tool and slow it down on higher levels. In addition to these configurations, the user can define the language of the input text. For this purpose the user needs to define the language code (for example *fi* or *en*) for

LAS options
Maximum N-gram length
Analysis depth
LAS Locale

Table 3.1: LAS options in ARPA.

Candidate filtering options
Required LAS tags
Disallowed LAS tags
Strongly disallowed LAS tags

Table 3.2: Candidate filtering options in ARPA.

Query options
SPARQL endpoint
SPARQL query
Query using original form
Query using base form
Query using inflections
Guess baseforms for unknown words
Query modifying every part
Query modifying only last part
Query using all permutations

Table 3.3: Query options in ARPA.

the LAS Locale option.

Once LAS has analyzed the input data and extracted candidates for entity linking, the next step enables the filtering of these candidates. The candidate filtering options, listed in Table 3.2, can be used to filter the textual entities or words used in linking. The user can define different LAS tags corresponding to parts of speech classes to include or exclude only specific classes. It can be used to disallow for example pronouns and conjunctions or allow only for example nouns. The disallowing of LAS tags has two levels: disallowing and strongly disallowing. The difference between the two is that the strongly disallowing of tags filters out words if any of their analyses contains the defined tags whereas the disallowing filters out the words that are interpreted as the user specified tags by LAS.

After possible candidate filtering the tool can perform entity linking. For entity linking a set of query options, listed in Table 3.3, are required. The linking is done by matching extracted candidates into the ontologies. ARPA requires the use of ontologies that are in RDF format and can be queried using their SPARQL endpoints. Therefore, it is required that the user defines a targeted SPARQL endpoint and a SPARQL query. The SPARQL query needs to be configured to be able to find matching concepts for the given candidates. In addition to the query and endpoint, the candidates can be transformed into their base form (basic form) or inflected form to improve

results. The possible options in ARPA for modifying the candidates are query using the original form, base form, or inflected form. The inflections can be defined for the tool to transform the textual entity into the user specified form. In addition to modifying the word form, the tool can be configured to modify open compound words (such as *post office*, *ice cream*, and *real estate*) and all their parts or only the last part. For example the inflected form of a Finnish place *Viipurin torilla* (at Vyborg's square, in English) transforms into its base form *Viipurin tori* (Vyborg's square, in English) when modifying only the last part of the name. The tool also enables the user to use all possible permutations that the tool can generate for a textual entity. For example, if some parts of the given string need to be transformed into their base forms but not the ending, the application can try to query using different versions of the input string. In case the tool does not succeed in base forming of words there is a possibility to guess the base form. For this purpose there is the guess base forms for unknown words setting that can be used when the input text contains words that do not exist in tools vocabulary.

After processing text, the tool produces results in JSON format where the found matches are in the original format and in specified query format. In addition, the URI, the original form and label of a concept in the target ontology are given. The unidentified textual entities from the texts can be recognized by other ARPA configuration or processed later. These terms can also serve as keywords and therefore should not be ignored by the application. Therefore the automatic application tool needs to have an option for the user to specify whether these terms should be ignored or not.

The candidates identified by ARPA are collected by the automatic annotation application from the JSON response. The candidates can be matched back to the original string input. In case of finding two overlapping matches for a string in the original text (such as *Indiana Jones and the Last Crusade* and *Crusade*), the longer string is selected for further processing. In addition, ARPA tool can be configured to use hierarchical structures and semantic relations to do enriching of terms. Some ontologies and thesauri contain information regarding relationships between terms such as synonymy, hypernymy, or meronymy. The information can be used to find more fitting keywords or to get a more accurate measure of a keyword's relevancy. For example, the terms *friend*, *buddy*, and *pal* can be grouped together the ontology contains these terms and has identified them as synonyms.

3.4 Candidate Ranking

Finally, the candidate ranking component uses the results of the linguistic preprocessing and candidate extraction components to identify the relevant candidates. The application evaluates each candidate based on information gathered in the candidate extraction phase to evaluate the relevancy for the candidate. The evaluation can use different weighting schemes to calculate a score for each candidate, and rank them from best to worst. Because there are many ways to evaluate and estimate relevancy for the candidates, it is possible to develop and add new weighting schemes into the application.

The ranking itself is optional and there are many possible heuristics for the selection of keywords, such as defining a fixed amount of keywords, selecting a cutoff value for keywords weights as a filter, or selecting a range for keywords based on document length. Due to these reasons the ranking needs to be a highly configurable feature. Based on the configuration, all or some keywords are picked for the document. The selection of keyword candidates is the end result of the candidate ranking component.

Chapter 4

Case Study: Kansa Taisteli Magazine

The first case for automatic annotation is the Kansa Taisteli magazine articles. Currently the magazine articles are publicly available in PDF format via a website of The Association for Military History in Finland[1] in collaboration with Bonnier Publishing. The magazine articles are accompanied with a PDF file containing metadata for 3,385 articles. The metadata has been collected manually by Timo Hakala. [33] The metadata contains information regarding the article (author, title, issue, volume, and pages) in addition to annotations describing the content (war, arms of service, military unit, place, and comments). The metadata has been converted into an RDF format by Kasper Apajalahti and it can be used to browse the magazine articles via a faceted search demo application[2]. The data is available as linked open data service [3] and there is the semantic WarSampo portal [4] that uses the data. The semantic portal user interface contains several different perspectives to war history. The data covers the Winter War (1939–1940), the Continuation War (1941–1944), and the Lapland War (1944–1945). [42]

The purpose of automatic annotation is to enrich the existing Kansa Taisteli magazine article metadata by identifying named entities (such as people, places, and military units) from the text and to provide links to related materials. In addition, the metadata is used to improve the faceted search application and to make it easier to search and browse articles.

The annotation process is illustrated in Figure 4.1. For the Kansa Taisteli magazine articles, this process is as follows: the input is a collection of PDF

[1] http://kansataisteli.sshs.fi
[2] http://www.ldf.fi/dataset/kata/faceted-search/
[3] http://www.ldf.fi
[4] http://www.sotasampo.fi

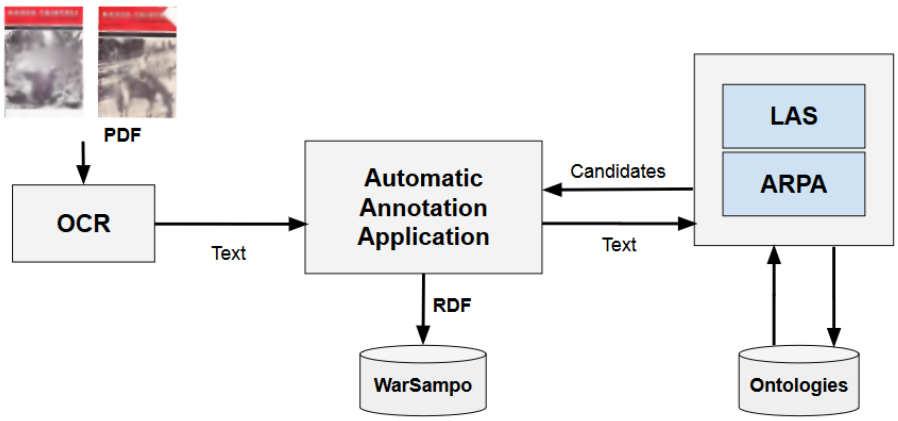

Figure 4.1: The annotation process for Kansa Taisteli magazine articles.

files and the text for annotation is extracted using OCR (Optical Character Recognition) tools. Afterwards the original metadata is used to identify the articles based on page numbers. After identification of articles, one article at a time, the text is used to query for matching concepts from selected ontologies. Once the annotations have been created, the results are written in RDF format and the WarSampo dataset is updated. The OCR (optical character recognition) process, selection of ontologies, data linking, and finally the usage of the enriched data is described in the following sections of this chapter in more detail.

4.1 Optical Character Recognition

In automatic annotation it is important to be able to annotate and process different inputs. Some automatic annotation systems require annotated PDF files or images that may contain text. For this purpose, an automatic annotation tool needs to be able to provide annotation and process text within these file formats. Optical Character recognition (OCR) tools can be used to extract texts from these file formats.

OCR is a complex technology that converts PDFs or images (typically scanned books, screenshots, and photos with text) with text into editable formats such as TXT, DOC, or PDF (with an added text layer) files. [75] Today, there are different types of OCR software available, such as Desktop OCR, Server OCR, and Web OCR. The accuracy rate of OCR tools varies from 71 to 98 percent. [37, 80]

OCR can recognize both handwritten and printed text. However, the

performance of OCR is directly dependent on the quality of the input documents. OCR is designed to process images that consist almost entirely of text, with very little non-text clutter obtained from pictures. [80]

The OCR process consists of multiple steps. Every step is a set of related algorithms that do a piece of the OCR processing. Each step is important for the success; the whole OCR process will fail if only one of its step cannot handle the input. Every algorithm is critical and they are required to work correctly on the range of images. The OCR process yields better quality results if some features of the given images are known making the task easier. Usually this becomes possible if only one kind of images are processed. A good OCR system must have the ability to adjust the most important parameters of every algorithm; sometimes this is the only way to improve recognition quality. [75]

During execution of an OCR system, a few problems can occur. One common issue for the system is that it can confuse letters, symbols and digits while trying to interpret the given input. Some them, such as the digit "1" and letter "l", are very similar to one another and it can be hard to differentiate and recognize them correctly from the input. In addition, text on a very dark background or printed on graphics can be difficult to extract. [80]

OCR Process

In order to implement the automatic annotation for the Kansa Taisteli magazine articles, the application requires the articles in text format. For extracting texts from the PDF files, there exists several OCR tools and from them two were selected: ABBYY FineReader and Tesseract.

Tesseract is a free open source optical character recognition tool. It was originally developed at HP in 2005 and was released as open source. Since 2006 it has been developed by Google and is available at GitHub[5]. [68] It has developed greatly over the years and fares well already in comparison to the commercial OCR systems although its architecture has stayed mainly unchanged over the years. [13] Its OCR engine differs from the other tools and even today some of the phases of its OCR process can be considered unique. [90, 91] The system supports 108 different languages, including Finnish, and can be trained to recognize other languages. Tesseract supports various output formats such as plain-text, HTML, and PDF. Regarding input formats, Tesseract supports image formats such as TIFF or PNG. [107]

[5]https://github.com/tesseract-ocr

ABBYY FineReader[6] is a commercial OCR system. In this project FineReader 11.0 was used as it was the latest version at the time available. It supports a wide variety of languages, a total of 189 languages (including Finnish), and can be considered one of the most popular OCR systems on the market. [1, 101, 103] The OCR system supports numerous input and output formats. The supported input formats can be images (such as PNG, JPEG or TIFF) or PDF files and most common images are supported as black and white or as color pictures. It has a rich selection of supported input and output formats such as PDF, CSV and TXT. [1]

During evaluation of the OCR tools, it was noted that Tesseract consistently produces solid results that contained few errors. ABBYY FineReader, on the other hand, seemed to fare better with the Finnish texts as the error rates were much lower than with Tesseract. However, during testing it was noted that, unlike with Tesseract, ABBYY seems to mix up paragraphs for unidentified reasons. Therefore, it was decided that both tools needed to be used to get the best results from the OCR process. Both tools would be used to extract text and the results would be combined.

The combining was be done by fixing Tesseract's results using ABBYY's results with the exception of avoiding parts where the order of paragraphs was broken. The process of combining results was semiautomatic; using word processor tools and comparing the results of the two and merging them into one result in the end. In addition, occasionally some errors (such as problems with paragraphs) needed to be fixed manually.

In addition to the issues created by the differences of the tools, there were also problems with interpretation of the text itself regardless of the tool. For example, the names of military units have abbreviations that are commonly used in the text. The abbreviations are inflected in Finnish by appending a colon and the inflection ending into the end of a word's basic form. However, in OCR, the colon was often read as i or z. Most of these errors could be corrected using regular expressions on the results. [42] For this purpose a list of regular expression rules was created and it can be found in Appendix A. The list of rules contains 164 rules in total and it was created while fixing the results of the OCR tools. The end result of this phase was the fixed texts that could be used later on in the annotation process in addition to the regular expression list for fixing OCR results.

[6] http://www.abbyy.com

4.2 Ontologies

In order to annotate the articles, the automatic annotator application requires for the process a set of ontologies and their ARPA configurations. The chosen ontologies here were from the WarSampo project: people, military units, Karelian places, and municipalities. External ontologies, such as KOKO ontology and DBpedia, were also used to enrich the annotations with more general terms. The order of ontologies impacts the annotations; the first ontologies match the most terms from their vocabularies and this can impact the ability to match terms into other ontologies. For example a text can contain the open compound word *ice cream truck* that can be found from ontology A whereas the term **ice** can be found from ontology B. Therefore it is good practice to arrange the ontologies into the configurations in the order of execution, starting from the most context specific and lastly the more general ontologies. In this case the order of ontologies is the following: people, military units, Karelian places and municipalities, DBpedia, and lastly the KOKO ontology. The general configurations for all ontologies include the filtering of forenames and surnames (except for person and place ontologies), extracting only terms that have been POS tagged as nouns or proper nouns, base forming of terms and setting the default language to Finnish. Nouns and proper nouns are selected as keyword candidates because nouns are preferred parts of speech for terms [5]. In addition, selected ontologies mainly have the terms in form of nouns (for example KOKO ontology) [88] and proper nouns (for example ontologies of the WarSampo project). Each ontology and their ARPA configurations are described in the following sections. The SPARQL queries are represented for each ontology in the Appendix B.

Person Ontology

The person ontology contains information about military personnel, including their ranks and details about casualties. Currently the dataset consists of 96,000 person instances of which approximately 95,000 instances extracted from National Archives Service's war casualty records from 1939 to 1945. In each instance record there are the basic properties such as the person's surname, forenames, a possible description, and a link to the information source. Most of the information is modeled as events like person's birth, death, promotion, or joining a military unit. [42] The person ontology is used to identify mentions of people from the Kansa Taisteli articles.

In Kansa Taisteli articles the mentions of people are in multiple different forms: first forename and surname, full name (all forenames and surname),

initials of the forenames and surname. In addition, the name is at times preceded by a rank or title. Sometimes there is only the rank and the surname mentioned (for example marsalkka Mannerheim, Marshal Mannerheim in English). Therefore, the use of the ontology had to be configured to accommodate all these alternatives. Also the SPARQL queries for ARPA had to be built keeping these variations in mind. [42]

In regards to ARPA configurations, due to the long names and titles the n-gram length is 5. The SPARQL query is configured to only match strings that are longer than 3 characters and contain substrings starting with a capital letter indicating that they contain a name. In addition, the query is set to use base form by transforming every part of the string into a basic form. Regarding unknown words, the application is set to guess their basic form in order to find more matches. Due to performance reasons the analysis depth is defined as 0.

Military Unit Ontology

The military unit ontology consists of over 3,000 Finnish Army units including Land Forces, Air Forces, Navy and its vessels, Medical Corps, stations of Anti-Aircraft Warfare and Skywatch, Finnish White Guard, and Swedish Volunteer Corps. The data model of a military unit has many similarities with the person ontology. [42] The military unit ontology is used to identify mentions of military units in the Kansa Taisteli articles.

In Kansa Taisteli context there are military units of different wars referenced in the articles. The official military unit names are usually relatively easy to recognize from the text. Occasionally, some of the units have ambiguous special names which makes the recognition difficult. One example of such a unit is the military unit called *Puolukka* (lingonberry in English). These special cases get mixed up with ordinary terminology outside the war context. They require specialized handling and careful interpretation of the context and language. [42] Therefore the military units ontology is used before the more general ontologies. The purpose of this order is to ensure that the alternative names for military units are identified from the text before they are linked to other ontologies. In addition, the SPARQL query constructed for this ARPA configuration is configured to aid in the detection of these proper nouns. Typically unit names such as "Puolukka" are usually written with the capital first letter and the SPARQL query can use this rule to identify the unit names from other words unless they are starting a sentence.

Another challenge that arose during the case was the definition of a military unit. Sometimes the names of the units have changed significantly

between and during the wars. The identification of military units poses a problem for linking because the names change and it is difficult recognize military units correctly as the same name can be used by other units during the wars. Currently the ontology covers units of the Winter War and only Air Force and Navy units of the Continuation War. Arms of services are same for all the wars during the WW2 in Finland. [42] The automatic annotator was required to take these features of this ontology into consideration. [42] In Kansa Taisteli metadata, each article contains a property that indicates the war during which the events of the article took place. The information was used to annotate only the articles that concern the events of Winter War using the military unit ontology to minimize misinterpretations.

Regarding the ARPA configurations, due to the long names and titles the n-gram length is 3. The SPARQL query is set to match strings longer than two characters and strings that start with a capital letter. The special characters are filtered out in order to improve the performance of the query. Also the query is set to modify every part of open compound words, to modify only the last part, and to use base form. Unlike in other ARPA configurations, the analysis depth is set to 2 in order to provide more in-depth analysis of the input so it can be linked with better accuracy. The level of disambiguation is high to decrease the possibility of misidentifying a name of a military unit such as "Puolukka".

Place Ontologies

Places have been modeled with a simple schema used in the Ontology Service of Historical Places (Hipla) [40], which contains properties for the name of the place, type of the place, coordinates, polygon, and part-of relationships of the place. [42] The place ontologies are used to identify Finnish and Karelian places in Russia from the articles.

In the case of places there were two ontologies available for identifying places from text: Karelian places and Finnish municipalities. Karelian places and municipalities contain towns, cities, municipalities, villages, water formations, buildings, and place targets in the terrain (such as a bunker). In terms of configurations, the ontology is restricted to fetch only entities that match to cities, villages, and towns. In most cases most of the smaller places are never mentioned in the Kansa Taisteli articles. Often times there may be a village carrying the same name as a building or a lake. Therefore, it was seen as useful to rule out all but municipalities, towns, and villages to minimize confusion. Other configurations include n-gram length of 2 and analysis depth being 1. The SPARQL query is set to match strings longer than two characters and strings that start with a capital letter. The special

characters are filtered out in order to improve the performance of the query. In this configuration filtering of forenames and surnames cannot be used because Finnish names for places and villages are similar to surnames [82], such as "Kestilä" which can be a name of a place or to a person's surname. The ARPA configuration is also set to query modifying only the last part of open compound words and to use their base forms.

In addition to these configurations, the application was configured to solve ambiguity issues. The configurations are based on the assumption that in most cases some types of places are usually better matches than others. For example, it was assumed that municipalities are better matches than towns or villages. So in case the application needed to choose which concept suits the best for a particular text, it always chose a municipality if possible and afterwards a town or a village, if a there was no matching municipality.

DBpedia

DBpedia[7] is a multilingual knowledge base that has been built by extracting structured information from Wikipedia. It has been created as a part of the DBpedia project and its structure is maintained by the DBpedia user community. The DBpedia project has published releases of all DBpedia knowledge bases for download and provides SPARQL query access to 19 language editions, excluding Finnish. [24, 57] The SPARQL endpoint for Finnish knowledge base has been created by the Semantic Computing Research Group based on the data obtained from the DBpedia project. [87]

In Kansa Taisteli case the DBpedia has been used to extract more general war terminology from the documents. In order to achieve this goal, DBpedia has been restricted to allow ARPA to match only to war related concepts. In the SPARQL query this was achieved by restricting the query to only match to concepts defined in war related categories. The SPARQL query is presented in Listing 4.1. In the SPARQL query, the first letter is capitalized and a language code is added to the query strings. Afterwards the modified query strings are used to match related terms from DBpedia's war related categories, ignoring matches that refer to categories or properties.

Listing 4.1: DBpedia SPARQL query.

```
PREFIX text: <http://jena.apache.org/text#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

[7] http://wiki.dbpedia.org/

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dbpfi: <http://fi.dbpedia.org/resource/>

SELECT ?id ?label ?ngram ?source {

 # VALUES contains n-gram query strings.
 VALUES ?ngram {
  <VALUES>
 }

 # Transforming the first letters into the capitalized form
 # and adding a language code for the query string.
 BIND(STRLANG(CONCAT(UCASE(SUBSTR(?ngram,1,1)),
      LCASE(SUBSTR(?ngram,2))),"fi") AS ?label)

 # query for labels using the original query string.
 ?id rdfs:label ?label .

 # filtering targeted categories of DBpedia ontology
 # to include war related categories such as WW2,
 # warfare, and war history categories.
 {
  ?id dct:subject/skos:broader*
      dbpfi:Luokka:Toinen_maailmansota .
 } UNION {
  ?id dct:subject/skos:broader*
      dbpfi:Luokka:Sodankäynti .
 } UNION {
  ?id dct:subject/skos:broader*
      dbpfi:Luokka:Sotahistorian_teemasivun_luokat .
 }

 # removing matches that are categories or properties.
 FILTER(!STRSTARTS(STR(?id),
        "http://fi.dbpedia.org/resource/Luokka:"))
 FILTER(!STRSTARTS(STR(?id),
        "http://fi.dbpedia.org/property/"))
}
```

In addition to the SPARQL configuration, the other configurations used were the n-gram length set to 3, analysis depth set to 1 to do more careful disambiguation of concepts, and queries done modifying every part of open compound words and only the last part to get a large variety of combinations matched.

KOKO Ontology

The KOKO ontology[8] is a collection of Finnish core ontologies. It consists of General Finnish ontology (YSO) and a group of expanding ontologies. [43] The KOKO ontology is a part of the Finto service. Finto offers a platform for publishing and using mainly Finnish thesauri, ontologies, and classifications. [49, 50] The KOKO ontology is used to annotate Kansa Taisteli articles with general terminology.

The KOKO ontology was configured to query using base form and original form with n-gram length 3 to target as many words and open compound words as possible. The analysis depth is defined as 0 due to performance reasons.

4.3 Data Linking

After configuring the ontologies and executing the application, it has produced a dataset that has been enriched with new annotations. For each article the application has added all or some of the following properties *:mentionsPerson*, *:mentionsPlace*, *:mentionsGeneral*, *:mentionsWar*, and *:mentionsUnit*, indicating military people, places, general terms, general war themed terms, and military units found from the articles, respectively. These properties are defined in the configuration along with the output format and target file. In the case Kansa Taisteli magazine articles, the annotations are not ranked based on relevancy. All found and linked annotations have been added into the dataset without candidate filtering based on term relevancy. The unidentified textual entities are filtered out respectively.

There were several difficulties that were faced during the linking and annotating of the Kansa Taisteli articles. One of them was that the texts contained a lot of clutter for example from the images. The OCR tools could not ignore images that contained text or interpret correctly PDF magazine issues that were in a bad condition. These issues made some magazine articles difficult to process and resulted in clutter such as strings containing erroneous letters, special characters, and numbers. The clutter was difficult to process in LAS and ARPA as it slowed down both tools and needed to be removed. Because of these reasons clutter removal was added into the application and sometimes also into the SPARQL queries. In addition, the application was limited to query only a maximum of 3,000 characters long strings at the time. Before querying the input text was sliced to chunks of text, avoiding splitting

[8] http://seco.cs.aalto.fi/ontologies/koko/

of words primarily. Also preferably, the application tries to avoid splitting of sentences but this cannot always be guaranteed.

In addition, the quality of the original metadata caused issues during the annotation process. For example, because the magazine articles were manually scanned in a laborious process, full-page advertisements were sometimes not included. However, when locating the articles based on the metadata, this threw off the application sometimes even by several pages. Resulting in annotating partially the wrong articles and producing erroneous annotations. In addition, the start and end of an article could not always be identified efficiently and some erroneous annotations were included from the preceding and following articles occasionally. In order to improve the results, the magazine metadata was fixed manually.

Another difficulty that arose from the annotation process was the ambiguity issue with places and people. Often, the application found multiple hits for different names, especially when matching initials. The person ontology also contained a vast amount of deceased people. Originally, the person ontology contained only high ranking military personnel but it also has been linked to casualties data. Via the person ontology the application matches the text to casualties data. Sometimes this results in several possible matches and that is hard to minimize because it would require more sophisticated identification of other references.

4.4 Application: Faceted Search

The purpose of the faceted search application is to help a user to find Kansa Taisteli articles and to provide context to the found articles by extracting links to related WarSampo data from the texts. After annotating documents the new metadata was added to the WarSampo portal. In addition to adding new metadata, the Contextual Reader (CORE) [70] was integrated into it to improve the user experience and to utilize the new data by adding into the application a facet that contains all the mentions that were found during the annotation process.

The updated Kansa Taisteli magazine article perspective is shown in Figure 4.2. In the perspective the user can find articles by using the author, magazine, related place, army units, or mentioned terms facets. The facets are on the left side of the perspective and the article details on the right. All but the army units (arms of service and/or unit) facet are by default closed and can be opened by clicking at the title of the facet. The facet will show a list of mentioned terms and names that can be used to filter the article list.

The magazine and army unit facets contain data in hierarchies. The

magazine facet represents initially magazine issues by year and by selecting a year, it shows magazine issues for that year. The user can pick a one issue and its articles are shown on the article listing. Similarly the army unit facet shows the arms of services and under each of them the corresponding military units from which the user can pick one and update the article listing. The hierarchies can be also used for query expansion: by selecting an upper category in the facet hierarchy one can perform a search using all subcategories.

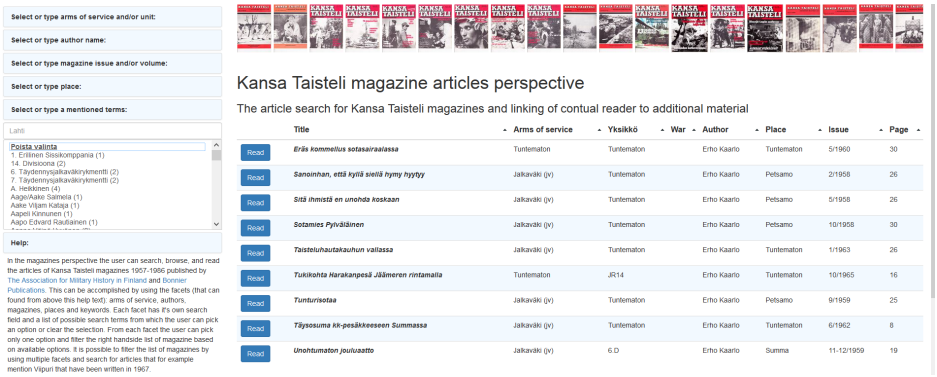

Figure 4.2: The faceted search browser targeting the Kansa Taisteli magazine articles.

The mentioned terms facet adds diversity into the article search. Originally there were no general terms and names of people in the manually created CSV metadata file. By adding mentions of terms and names as a facet into the web application, the user can find articles that contain certain terms, military units, people, or places. For example, a user can search for articles that mentions a person or the term *lice*. The original facets (the author, place, magazine, or the military unit facets) do not contain mentions of people or general terms. In addition, the facet contains all mentioned places and military units; unlike in the original which contains only one place or military unit/arms of service per article. The mentioned terms and names facet improves the findability of the articles.

Chapter 5

Case Study: Semantic Finlex

Semantic Finlex service offers Finnish legislation and case law as linked open data. Originally Finnish legislation and case law was published as human readable documents in the Finlex service[1] that has been open for public. The data, however, was never published in a machine readable format via open interfaces and it was not linked to any external data sources. In order for third parties such as developers, different web services, and computer applications to use the data in their own applications, it was required that the data is transformed into a machine readable format and published as linked data. [30] Eventually, all this was wrapped up into an open service[2] as a part of the Semantic Finlex project for users and developers.

The purpose of automatic annotation in the Semantic Finlex project was to make it easier to read, find, and browse statutes and case laws. To achieve this the metadata had to be enriched by linking it to internal and external ontologies. [30] The goal of automatic annotation is to describe the contents of each document accurately and plentifully using keywords. In addition to the enriching of the metadata, the task was to provide visualization of the documents in a form of a tag cloud.

The annotation process is depicted in Figure 5.1. For Semantic Finlex this process goes as follows: the input is a collection of HTML documents and from them the titles and the content is extracted for annotation using existing Python libraries. Afterwards the text is used to query for matching concepts from selected ontologies. Once the annotations have been created the results are written in RDF format and the Semantic Finlex dataset is updated. The text extraction, selection of ontologies, data linking, and finally the usage of the enriched data is described in this chapter in more detail.

[1] http://www.finlex.fi/

[2] http://data.finlex.fi

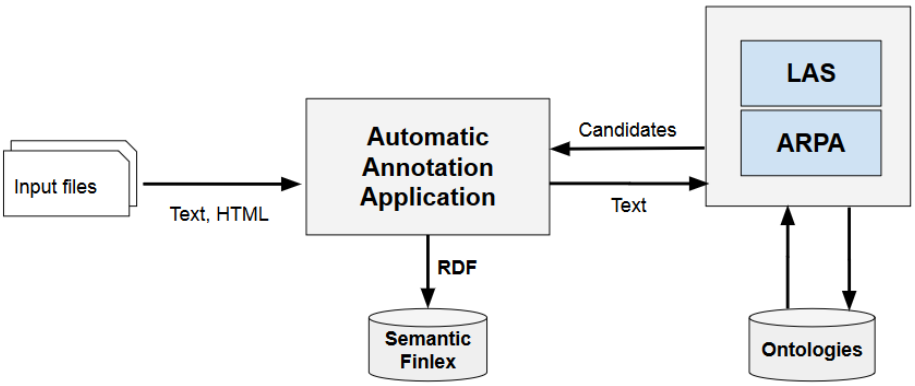

Figure 5.1: The annotation process for Semantic Finlex.

5.1 Ontologies

While annotating the law documents, the automatic annotator used several vocabularies similarly to the Kansa Taisteli case. These vocabularies or ontologies required a set of configurations to get the optimal result of the annotation process. The chosen ontologies here were: Combined Legal Concept Ontology, Original Finlex Vocabulary (FinlexVoc), EuroVoc ontology, KOKO ontology, and DBpedia. The general ARPA configurations for all cases include the filtering out all but nouns and proper nouns, base forming of terms, query modifying every part of a term, analysis depth set to 0 for performance reasons, and the default language is set to Finnish. The reason to use only nouns and proper nouns the same as in the Kansa Taisteli case: most of the terms in ontologies are proper nouns or nouns. Each ontology and their ARPA configurations are described in the following sections. The SPARQL queries are presented for each ontology in the Appendix B.

Combined Legal Concept Ontology

Combined Legal Concept Ontology is an ontology of legal concepts. It has been created by combining three different thesauri (Asseri, Suomen Laki, Edilex) in the field of the Finnish legislation. It contains ca. 9,000 different terms, all in Finnish. [30] The ontology is used to retrieve relevant legal terminology.

Regarding the ARPA configurations, the n-gram is set to 3 to pick up longer terms. The SPARQL query is set to exclude numbers and the length of the terms is calculated to enable selecting of the longest match for the terms. Also the query uses the original form to increase the amount of

matching terms and guessing of the basic forms for unknown words to increase performance. In addition, the nouns of the query strings are inflected into their plural form in order to target also terms that are in plural form in the ontology.

EuroVoc

EuroVoc[3] is a diverse multilingual thesaurus that covers the activities of the EU. It contains terms in 26 languages. The users of EuroVoc includes for example the European Union institutions, national and regional parliaments in Europe, national governments, and private users around the world. [95]

The EuroVoc thesaurus is used to query general purpose terminology. In regards to the ARPA configurations, the query is not confined to any specific part of the thesaurus and the n-gram is set to 3 to pick up longer terms. The SPARQL query is set to exclude numbers, match only to Finnish terms, and to also try to match strings into synonyms. The length of the terms is also calculated to enable selecting of the longest match for the terms. Also the query is set to use the original form to increase the amount of matched terms. The inflections are set in order to target inflected forms of terms as well.

Finlex Vocabulary

The Finlex Vocabulary (FinlexVoc) contains legislation specific terminology in addition to more general terminology. It has been used to annotate existing consolidated legislation and used in browsing of the documents in the Finlex service. It is based on the vocabulary of the Statutes of Finland and is maintained by Edita Publishing Oy. [77]

The Finlex vocabulary is used by the automatic annotation tool to match general and legislation specific terms. The SPARQL query is set to match only to Finnish terms. In terms of ARPA configurations, the n-gram is set to 3 to pick up longer terms. In addition, ARPA is set to query using the original form to increase the amount of matching terms from KOKO ontology. Also, in order to match longer terms with better accuracy, the query is set to modify every part and the last part of a query string. The inflections are set in order to target possible inflected forms of terms.

[3]http://eurovoc.europa.eu

KOKO Ontology

KOKO ontology is used to recognize law terminology and to find more general terms. In the project's configurations for KOKO ontology the queries are not confined to any specific terminology. The purpose is to find more general terminology to complement the end result. The SPARQL query is set to exclude numbers and to match only to Finnish terms. In terms of ARPA configurations, the n-gram is set to 1 to pick up single words. In addition, ARPA is set to query using the original form to increase the amount of matching terms from KOKO ontology.

DBpedia

DBpedia is used to extract law terminology from the documents just like from the previous two ontologies. DBpedia is also restricted to law terminology in SPARQL and the matches to category names or properties are ignored. The SPARQL query is set to exclude numbers. Other ARPA configurations include the n-gram being set to 3 to pick up longer terms. Also the configurations include a setting for guessing base forms for unknown words to get more diverse end results from DBpedia.

5.2 Data Linking

After creating a set of ARPA configurations for the given ontologies, they were used in the following order: Combined Legal Concept Ontologies, EuroVoc, FinlexVoc, DBpedia, and KOKO ontology. Just like with Kansa Taisteli magazine articles, the order of the ontologies can be used to minimize ambiguity in the results by linking the text to more domain specific ontologies first and later to more general purpose ontologies. The results of all ontologies are collected and filtered based on relevancy of the concept. These concepts were added into the default dataset and they have their own properties that links the concept into the corresponding document. These properties are *:mentionsLaw*, *:mentionsGeneral*, *:mentionsEuroVoc*, *:mentionsFinlexVoc*, and *:mentionsDBpedia*. The unidentified textual entities are filtered out respectively.

The initial results, however, were not satisfactory as there were problems with word recognition and ambiguity. The initial results contained keywords such as *artikla* (article, in English), *Suomi* (Finland, in English), and *laki* (law, in English). A stopword list was required to filter out most common terms such as *article*, *Finland* or *law*. A stopword list is a list frequent words

in a document collection that have little value in describing the aboutness of the document. [59]. The need to add and term relevancy analysis or weighing schemes arose with Semantic Finlex as the purpose of the task is to identify the relevant concepts and not all named entities like in the Kansa Taisteli case. The used vocabularies were more generic, and just fetching the all matching terms was not enough. That would have returned results that included terms that were not relevant in describing the content in a useful way. In addition to evaluating the term relevancy, the amount of terms needs to be limited in order to select only the most relevant keywords for each document. The need for a parameter that defines the limit became evident it needed to be evaluated for this document collection separately.

5.2.1 Weighting Schemes

One of the most central themes of information retrieval (IR) systems are the term weighing schemes that largely define the effectiveness of search. Most retrieval models have three central variables to determine the relevancy of a word for a document. Term frequency within a document, document length, and the specificity of the term in the document collection. Term frequency and document length are used in combination to derive how crucial the word is in a document. Term specificity is used to reward the documents that contain words rare in the collection. [79]

Based on term weight estimation, IR models can be divided into two groups: probabilistic models and vector space models. Probabilistic models focus on evaluating the probabilities of the words in the documents. The vector space model on the other hand queries documents as finite dimensional vectors, where the weight of an individual component can be calculated using variations of TF-IDF (term frequency and inverse document frequency) scores. [79] In the annotation process for Semantic Finlex, a simple TF-IDF was chosen to score each term found in the text. TF-IDF is one of the most commonly used term weighting schemes. The TF-IDF weight has two components, TF and IDF. TF, described in equation (5.1), computes the normalized term frequency f(t) by counting how many times a term appears in a document (d) and dividing that by the total number of words (w) in that document. The second term IDF, described in equation (5.2), the inverse document frequency is computed as the logarithm of the number of the documents in the collection of documents (D) divided by the number of documents where the specific terms (t) appears. [63] TF-IDF, described in equation (5.3), is the product of the two components.

$$TF(t,d) = \frac{f(t)}{|w \in d|} \tag{5.1}$$

$$IDF(t,D) = ln(\frac{|D|}{|d \in D : t \in d|}) \tag{5.2}$$

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D) \tag{5.3}$$

5.2.2 Keyword Density

After the application ranks the keywords with the help of a weighting scheme, it selects the set of keywords. Selecting the most relevant keywords is not an easy task for an application nor for a human. Semantic annotation is a tedious and difficult task; human annotator does not always produce optimal results and ends up taking multiple shortcuts. [28, 34] This results in fewer and less accurate annotations. [28, 34, 62, 73, 89] On the other hand, an automatic annotation tool can produce more than enough keywords but the quality of annotation does not compare with the quality of the human annotators. [62]

The Semantic Finlex dataset contained already manual annotations. The manual annotations were, however, scarce. The documents contained from 1 to 14 keywords. When analyzed comparing document lengths to the amount of keywords, the two did not seem to correlate. The most common amount of keywords per document was 1. On contrast, an automatic annotation tool can produce as many annotations as there are unique words and compound words in the document. The amount of keywords also known as keyword density needs to be defined for the tool.

It is hard to define an ideal amount of keywords for each document. There is no general gold standard for this task. The amount of keywords depends on the source material and the usage of the keywords. In document databases there are usually guidelines for annotations. For example, AustLit[4], an authoritative database about Australian literature and storytelling, uses ERIC (Educational Research Information Center) Process Manual [72] that has guidelines for selection of keyword amounts for different types of materials. [7] Based on an interview of a service manager of the Helsinki City Library, the libraries of Helsinki use from 3 to 30 keywords per document. The amount depends on the context of the document. [98]

An automatic annotation tool needs to provide the user with multiple configurations in order to limit the amount of keywords correctly for different

[4]http://www.austlit.edu.au/

types of texts. In order to find a good configuration for the Semantic Finlex material, different methods needed to be tested. Currently the automatic annotation tool can be configured to get a specific number of keywords (for example top 10 keywords), or to select keywords that have a weight above the average keyword weight. In addition, it is possible to limit the number of keywords into a user specified range (for example from 5 to 30 keywords) based on the document length. The configurational possibilities make it possible for the user to select the best method based on the used source materials and use cases.

5.3 Application: Tag Clouds

In addition to annotating the law documents, a tag cloud was generated. The main goal of generating tag clouds was to make it easier for the readers to grasp the central theme of a document. The tag cloud emphasizes the most relevant terms in the document by utilizing the font weight. In order to achieve this goal a fitting tool had to be made. After analyzing and testing a handful of good tag cloud tools PyTagCloud was chosen.

PyTagCloud is a python module that generates tag clouds from the given text in PNG and HTML formats. The module served as a suitable starting point for creating tag clouds. Initially it was created for Python 2 and it had only language support for German, French, Italian, English, and Spanish. [2] In addition there was no base forming nor other linguistic tools for the languages. In order to create tag clouds for the law documents in Finnish, the tool would need to add the language support for Finnish and the module would have to be updated to Python 3 so that is compatible with the automatic annotation tool.

The application was modified based on the shortcomings to produce sound tag clouds. However, this was not possible by only adding the language support and by making it Python 3 compatible. More modifications had to be made. Initially, PyTagCloud was added as a module to the automatic annotator that can be executed if configured. Many of the components of the annotator tool (such as linguistic preprosessing) were added into the tag cloud generator module. The tag cloud generator uses the same input as the automatic annotation tool to produce tag clouds and therefore there are several similar steps that need to be taken in order to produce sound tag clouds for Finnish texts.

Just like with the automatic annotator, the stopword lists are utilized to filter the results. Digits were also filtered out from the input data because they are not informative without context. In addition to the stopword

lists and digit removal, base forming and morphological analysis was added. The linguistic preprosessing module is used to apply the base forming and morphological analysis of LAS. These linguistic tools are needed to get more accurate results for the tag cloud. Morphological analysis is also used to only extract nouns and proper nouns from the text.

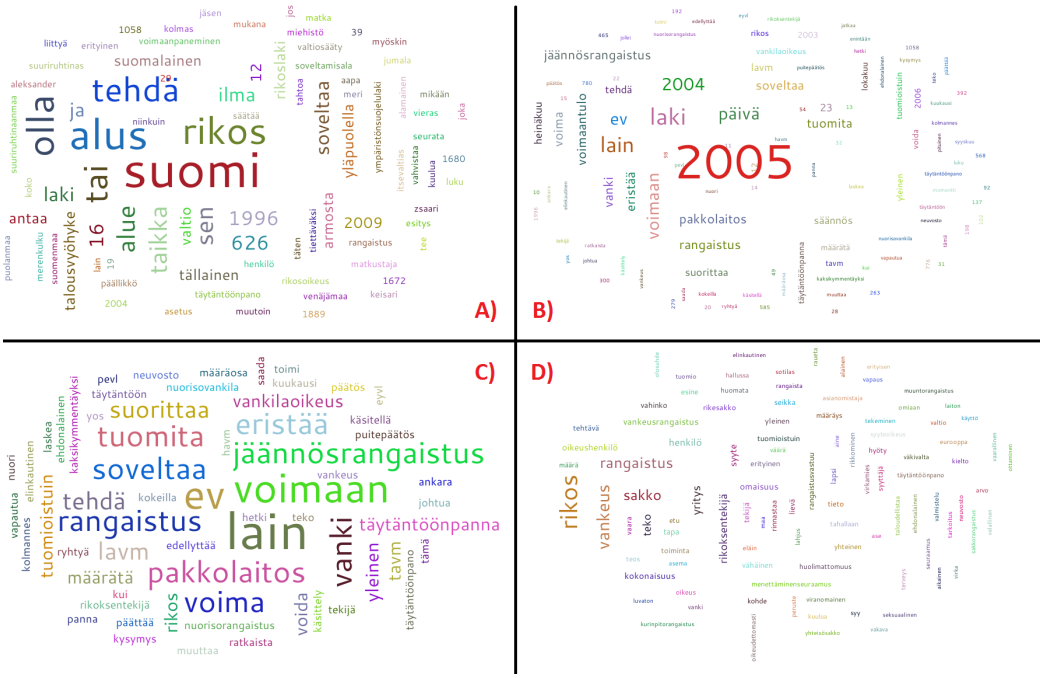

Figure 5.2: A Semantic Finlex tag cloud development.

After the linguistic preprocessing, the terms from the texts were extracted and linked using mainly the same ontologies as with the annotation process. The weights were assigned to the linked entities and other terms after the candidate extraction phase. The PyTagCloud tool had initially a weighting scheme. The default weighting scheme was overdriven with the same module that was applied with the automatic annotation tool to improve the emphasizing of the most relevant terms of the document. Originally the PyTagCloud module used term frequency and emphasized the most frequent terms regardless of their specificity to the document.

Because of the purpose of visually summarizing the documents, the amount of keywords required for the tag clouds was also much higher than the keywords for each document. The right amount of keywords was selected by testing with different configurations for the documents, eventually settling with the maximum of 50 keywords per tag cloud. In few tag clouds the amount of keywords was scarce due to lack of words in the source document.

These documents have tag clouds with a low amount of keywords. Because of these reasons the unidentified and unlinked terms were not filtered out from the results and the amount of terms increased in tag clouds but did not solve the issue completely. Little can be done to increase the number of terms in the document. One idea for further development might be to add hypernyms for the keywords. However, the initial results, without hypernyms and using the described setup, can be seen in the Figure 5.2.

Figure 5.2 shows the evolution of the tag clouds generated from the The Criminal Code of Finland after different modifications. In the cloud A) there is the initial untouched version of a tag cloud created by the fresh installation of the PyTagCloud. In the cloud B) a method of using stop word lists has been applied to the result, whereas cloud C) has removed digits, and finally cloud D) is a result of fine-tuning of the application by using linguistic methods and weighting schemes. The result of the fine-tuning in generally produced sound tag clouds that correctly emphasize the terms that are more relevant to the context of each document.

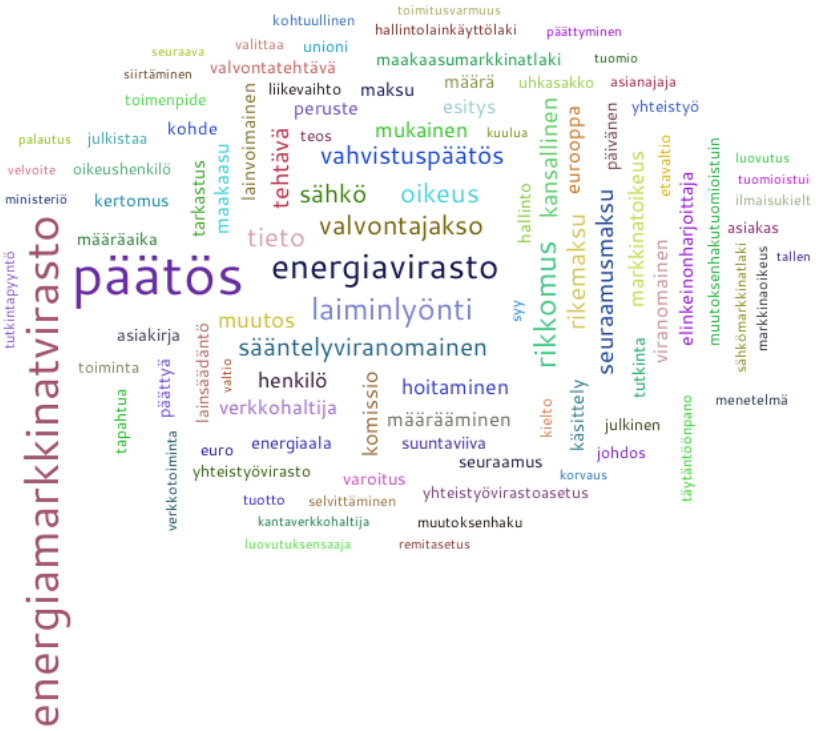

Figure 5.3: The Semantic Finlex tag cloud for the law on monitoring the electricity and natural gas markets (2013/590).

In addition to the tag clouds shown in Figure 5.2, another example of a tag cloud is shown in Figure 5.3 which is the Law on Monitoring the Electricity and Natural Gas Markets. The difference between the tag cloud in Figure 5.3 and the tag cloud D) in Figure 5.2 is how they magnify terms differently. Cloud D) in Figure 5.2 contains less terms than its predecessor tag cloud C). Also, the application occasionally produces tag clouds that magnifies insufficiently the most relevant terms; all terms are presented using too small font like in tag cloud D). The small weights impact the cloud and there is less variance between the font weight used in the cloud. The tag cloud configurations must be studied more to produce more solid quality clouds by the tool. The tag cloud in Figure 5.3 presents the terms more satisfiably than in tag cloud D) but not flawlessly. The word *energiamarkkinavirasto* (Energy Authority, in English) is problematic because the length of the term makes it difficult to place the term every time into an optimal spot. The issue exists also in some other tag clouds that contain longer words. However, the results are promising and with small adjustments it might be possible to improve the tag clouds even more.

Chapter 6

Evaluation

After executing the automatic annotation tool for both cases Kansa Taisteli magazine and Semantic Finlex, the results have been evaluated. For this three common information retrieval measures were picked: precision, recall, F-measure, and R-precision. Precision calculates the systems ability to present only relevant annotations whereas recall calculates the systems ability to present all relevant annotations. R-precision, on the other hand, expresses the precision for the top n keywords where n is the number of keywords in the original annotations. [63]

In this chapter the results are presented and evaluated for both cases. Firstly Kansa Taisteli magazine articles are viewed from a statistical point of view, followed by a more in-depth evaluation by calculating the precision, recall, and their weighted harmonic mean F measure to see how well the tool has performed in the context and by comparing three different inputs: untouched OCR output text from Tesseract OCR tool, automatically fixed text using regular expression patterns, and semi-automatically fixed text. After this the Semantic Finlex results are represented and evaluated by calculating the R-precision for the results.

6.1 Kansa Taisteli Magazine

In case Kansa Taisteli magazine, a small sample of 433 articles were annotated for evaluation because of the laborious nature of OCR processing and post-processing. From each decade a year was selected randomly and all magazine issues of that year were selected for processing. The statistical breakdown of the automatic annotation results is shown in Table 6.1. In the table there is the comparison of three different executions of the application for 3 different sets of inputs that have been produced by the OCR process:

#	Statistics	Tesseract output	Automatic	Semi-automatic
1	Total of documents	433	433	433
2	Total of linked entities	21 002	21 001	21 405
3	Total of unique linked entities	5550	5529	5705
4	Mode	35	34,35	35,37,38
5	Minimum amount of linked entities	1	1	1
6	Maximum amount of linked entities	216	216	271

Table 6.1: Statistical analysis of result dataset for Kansa Taisteli magazine articles.

untouched OCR output text from Tesseract OCR tool, automatically fixed text using regular expression patterns, and semi-automatically fixed text. The semi-automatically fixed text is the end result of using two OCR tools and combining their results using text processing tools. Some of the results were combined manually due to difficulties in combining the results of the two OCR tools. The automatically fixed text using regular expression patterns utilizes the regular expression list in appendix A that was created as a part of the process of semi-automatic fixing process. The regular expressions were applied to the unfixed OCR output in order to compare the results. Lastly, the unfixed OCR output was used to compare its results to the other two result sets.

The Table 6.1 shows that there are differences between result sets of Tesseract output (OCR), automatically fixed input (REGEX), and semi-automatically fixed input (HYBRID). The semi-automatically fixed texts produced the most promising results as expected while the differences between the automatically fixed and the unfixed pure OCR output texts remain small. The mode (the most typical number of linked entities for each article) is higher (between 35 and 38) for the semi-automatically fixed text. It produced the most linked entities in total and per article. In addition, the result set indicates that the maximum amount of linked entities per article is higher than with two other results sets. The result set of the automatically fixed texts indicates that it found less linked entities than the original unfixed text. The assumption here is that, although the texts fixed with regular expressions produced less linked entities, they produced more accurate results than the unfixed texts because some of the fixes clarify and fix OCR errors that made it hard to identify terms correctly. For example, an article that

talks about military unit *II/JR 37* contains erroneous references to it such as *11/JR37* and *1I/JR37*. These references are interpreted as *JR 37* and *I/JR 37* that refer to military units that are not mentioned in the text but are related to the military unit *II/JR37*. The regular expression rules can fix the errors and transform erroneous units into the form *II/JR37* from which the tool can link it to the corresponding resource.

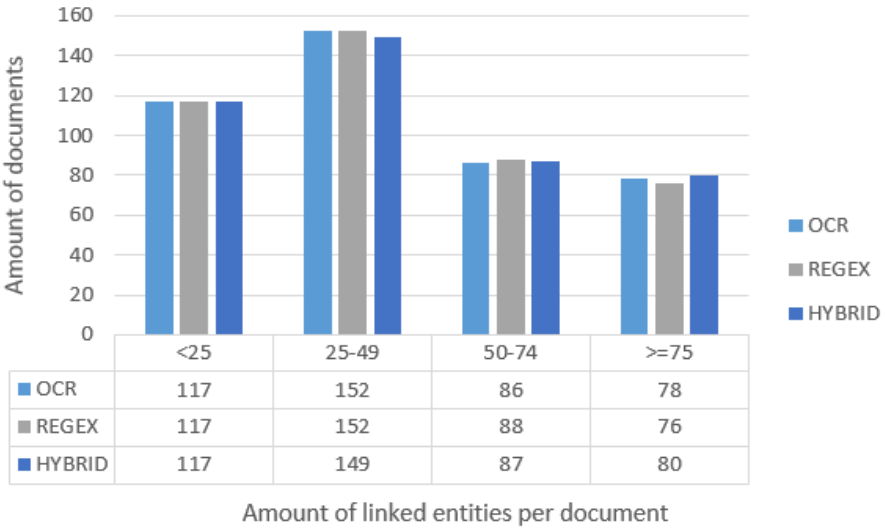

Figure 6.1: The distribution of linked entities amounts for magazine articles.

In Figure 6.1 the distribution of found linked entities is shown for all versions of the input data and compared to each other. From the distribution it can be noted that roughly a third of articles have more than 50 linked entities. The majority of the articles, however, have more linked entities. The distribution for each version of annotations varies little. The semi-automatically fixed input text results differ from the other two result sets by having slightly more documents with more than 75 linked entities. Respectively, the number of documents that have less linked entities has decreased.

A distribution of found linked entities based on six ontologies is shown in Figure 6.2. The most linked entities are found from the KOKO ontology followed by the WarSampo military person ontology and DBpedia. The place and the military unit ontologies contain the least amount of linked entities. As can be seen from the figure, there are also some differences between the different versions of input texts. The most notable differences come in the usage of the WarSampo persons and military units ontology. From the semi-automatically and automatically fixed texts, the same amount of military units have been found. Linked entities in the person ontology are

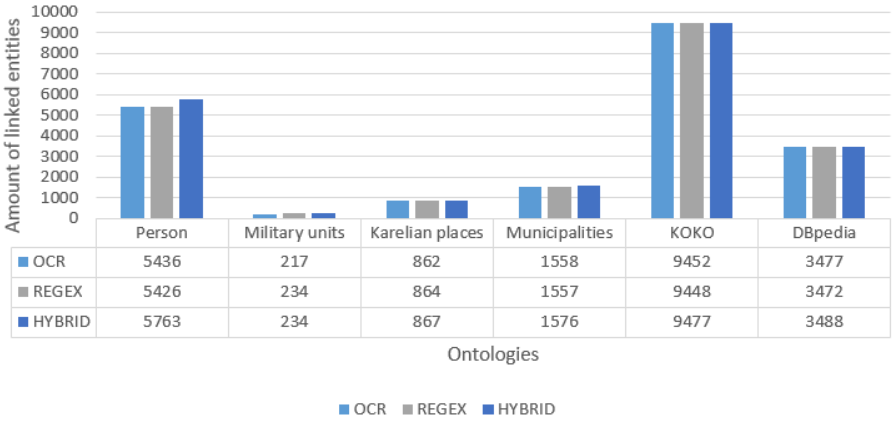

Figure 6.2: The distribution of found linked entities for each ontology used.

found in higher numbers from the semi-automatically fixed texts whereas the automatically fixed text has the least amount of linked entities. By taking a closer look at the result data, the OCR input text has OCR errors that impact the entity linking. The OCR errors reduce the amount of produced entity links. Regarding the person ontology, the OCR input text has more linked entities than the automatically fixed dataset. On closer inspection, the OCR result set contains erroneous linked entities. Some of these entities have been linked to wrong people due to OCR errors. In other ontologies there are some minimal differences between the inputs.

6.1.1 Precision and Recall

In contrast to Timo Hakala's original manual annotations, the results were richer. This also became visible when calculating and inspecting the precision and recall results. In order to calculate precision and recall, 50 articles were selected randomly from the pool of 433 articles. The evaluation of the annotation is laborious and therefore not all articles of the 433 could be selected for the analysis.

The calculations of precision and recall were done by looking for all the found matches in contrast to the manual annotations of Timo Hakala. From the original annotations the military units and places can be compared with the results of automatic annotations. The war, arms of service and comment annotations cannot be compared with the results of the automatic annotation tool because they are too general (war, arms of service) or the annotation field is used too irregularly (comments). For example, the arms of service field contains a name of the arms of service, such as the Air Force, which can

be found seldomly from the text. The identification would require careful inspection of the mentioned military units which are more frequently found from the text.

When comparing the results to the manual annotations, in most cases the exact match was found but also many related matches were found. The results were calculated using two methods: exact matches (method 1) and accepting also direct meronyms (method 2). Method 1 accepts only exact matches where the automatically found term must match the manually annotated term. Method 2 counts also other values as positive if the Timo Hakala's annotation served as an umbrella term for them. For example, Timo Hakala may have annotated the text with a place that is a municipality. The text itself may mention for example villages of that municipality and they were counted as positive matches for the municipality in method 2. In method 1 the villages are negative matches and only the municipality is a positive match. Based on these two methods the precision, recall, and F-measure were calculated and are presented for military units (denoted M. units) and places in Table 6.2. In this table the precision (denoted P) and recall (denoted R) values are represented with F-measure (denoted F) for the Tesseract output texts. In addition to methods 1 and 2, a third method is introduced. Method 3 calculates precision based on a manual evaluation of the annotation results. It interprets all matches evaluated as relevant extracted from the text as positive ones. In the following Tables 6.3 and 6.4 the same measures are shown for the annotations from the automatically and semi-automatically fixed texts.

Depending on the method the results vary. The precision is poor for all but method 3 as the annotation tool finds a great amount of military units and places from the texts. In method 3 this has been taken into consideration and improves the results significantly. The results contained also holonyms, meronyms, and occasionally relevant terms that are ignored in method 1. In method 2 the meronyms are also counted as positive matches. All of these are included in the method 3 and it explains the great improvements in the precision.

In addition to the differences between the methods, the difference between precision and recall for places and military units is notable. The precision is lower for the places mainly because of the regular expression fixes concentrating on military units. The military unit results are weighted down by a few remaining OCR errors whereas the issue with places was the configuration of ARPA queries in combination with the fact that some of the places could not be found using only the domain specific ontologies.

The difference between results of unfixed, automatically fixed, and semi-automatically fixed results, shown in Tables 6.2, 6.3, and 6.4, are notable

	METHOD 1		METHOD 2		METHOD 3	
	M. Units	**P**	**M. Units**	**Places**	**M. Units**	**Places**
P	26.14 %	6.78 %	30.26 %	10.47 %	80.95 %	62.87 %
R	69.70 %	38.46 %	67.65 %	51.92 %		
F	38.02 %	11.53 %	41.82 %	17.42 %		

Table 6.2: Evaluation of the annotations produced from the unfixed Tesseract output of Kansa Taisteli magazine.

	METHOD 1		METHOD 2		METHOD 3	
	M. units	**Places**	**M. units**	**Places**	**M. units**	**Places**
P	25.26 %	6.78 %	30.38 %	10.47 %	82.81 %	62.87 %
R	72.73 %	38.46 %	72.73 %	51.92 %		
F	37.50 %	11.53 %	42.86 %	17.42 %		

Table 6.3: Evaluation of the annotations produced from the automatically fixed Kansa Taisteli magazine articles.

	METHOD 1		METHOD 2		METHOD 3	
	M. units	**Places**	**M. units**	**Places**	**M. units**	**Places**
P	25.77 %	6.80 %	30.77 %	10.55 %	84.62 %	62.87 %
R	75.76 %	38.46 %	75.00 %	51.92 %		
F	38.46 %	11.56 %	43.64 %	17.53 %		

Table 6.4: Evaluation of the annotations produced from the semi-automatically fixed Kansa Taisteli magazine articles.

Name	Irrelevant annotations	Erroneous annotations	Total of false positives	Percentage
Method 1	151	88	239	63.18 %
Method 2	111	88	199	55.77 %

Table 6.5: Amount of incorrect place annotations based on the used gold standard.

but not significant. For method 1, the unfixed seems to be producing better results than the automatically fixed whereas the semi-automatically fixed produces the best results when comparing F measure scores for military units and places. The assumption that the unfixed OCR output text would produce less accurate (lower precision) than the automatically fixed text seems to be false for method 1. For the methods 2 and 3 the assumption, however, is true, and the precision is lower for the unfixed text in comparison to the results produced using the other two inputs. The results of the method 1 are impacted by the fact that the original annotations contain at most one military unit. However, automatically fixed input produces more mentions of military units than the unfixed input because many OCR errors have been fixed, resulting in decrease of precision.

The semi-automatically corrected output produced the best results in the annotation process. In Table 6.5 the amount of false positive matches of places are shown. The sum of false positive matches is formed from two components: the irrelevant annotations and the erroneous annotations. The number of irrelevant annotations are shown in Table 6.5 along with the errors that are presented later in Table 6.6 with more details. The irrelevant annotations are terms that have been identified from the text but are not included in the original manual annotations that function as a gold standard and therefore cannot be counted as positive matches. For both methods the high number of irrelevant annotations is the main reason why the precision is relatively low. The percentage column calculates the amount of these false positives relative to the total of all false positive annotations. In comparison to a study by Kettunen et al. [51] the automatic annotation tool produces similar results. It performed somewhat better in finding correct matches. The OCR post-processing had a positive impact on the results and it is visible that the recall was impacted by the amount of OCR post-processing, especially in the case of military units.

When inspecting the annotations in more detail they are mainly correct with some exceptions. The identification of military units suffered only from 5 to 6 errors depending on the method that originated from the OCR pro-

	Error type	Amount	Percentage
1	Wrong place	32	12.12 %
2	Ambiguous	14	5.30 %
3	Confusion between places and people's names	16	6.06 %
4	Noise from other articles	9	3.41 %
5	Clutter (for example advertisements)	7	2.65 %
6	ARPA / LAS error	1	0.38 %
7	Misidentified POS	9	3.41 %
#	TOTAL	88	36.07 %

Table 6.6: The breakdown of the error types found when the place annotations for semi-automatically texts were analyzed.

cess. However, the identification of places suffered from variety of errors. In Table 6.6 is the breakdown of the error types and they can be divided into three different groups based on the nature of the error. Firstly, there are errors related to the used ontologies (rows 1 - 3), secondly errors that come from clutter or noise (rows 4 - 5), and thirdly errors that result from the tools indicating issues in configurations (rows 6 - 7). The last row contains the total of occurred errors from the entire result set. The total is calculated from all the mismatched annotations that are included in the results set.

The first group of errors relating to ontologies contains errors relating to identifying wrong places and ambiguity issues. In terms of identifying wrong places, the text contained many references of Karelia. As it happens, the ontology for the municipalities does not contain the Karelia region but a small municipality in western Finland carrying the same name. Every time Karelia region is mentioned in the text, the application has annotated it using the village in the western Finland. Regarding ambiguity issues there were two kinds of issues. There was confusion between place names and sometimes the place names and people's names were confused. This is partially a configuration issue but also partially an issue with the ontology because it can contain errors. For example, one error existing in the Karelian places and municipalities is that there are two entities for Kemi that seem to be one and the same place based on the coordinates.

The second group constitutes of clutter and noise picked up from advertisements, maps, and previous or following articles. The input magazine issues contained many advertisements embedded into the start, end, and next to the actual articles. It was difficult to avoid clutter from these. The identification of images could also help to reduce clutter and noise from the maps, images, and some advertisements. Also the identification of the start

#	Name of the linked entity	Translation	Frequency	TF-IDF
1	aarteet	treasures	2	0.59
2	nainen, naiset	a woman, women	1	0.39
3	sanat	words	2	0.30
4	koti, kodit	a home, homes	1	0.20
5	mies, miehet	a man, men	1	0.20
6	metsä, metsät	a forest, forests	1	0.20
7	pellot	fields	1	0.20
8	lotat	members of Lotta Svärd	1	0.20
9	Eero Johannes Eräsaari	a person	2	0.20
10	ruoste	rust	1	0.10

Table 6.7: Annotation results for article: Sana rintamanaisille by Pentti Pyy.

and end of an article could possibly help to minimize the clutter. The start and the end of articles was, however, hard to identify. The RDF data for the articles only contained the starting page of each magazine article. Before annotating Kansa Taisteli magazine articles, there were issues with the data. It contained erroneous page numbers that had to be fixed. There still remain some faulty page numbers in the data, causing some errors with the annotations.

The third group consists of issues relating to configuration. As mentioned earlier, some ambiguity issues could be solved by changing the configurations. The forenames and surnames were not filtered out because some of the villages may carry the same names as persons. Especially Finnish last names originate from Finnish places sometimes [82].

6.1.2 The Results of the Annotation Process

The results of two randomly selected articles are presented in Table 6.7 and Table 6.8. In each table there are the top 10 terms that are ranked based on their calculated TF-IDF value. The term frequency is also shown in the last column. Although the TF-IDF value was not used to rank or to pick the most relevant annotations, it was used here in both tables to show a sample of the results of the annotation process for both articles.

The article presented in Table 6.7 has acquired some relevant and irrelevant linked entities, the most relevant being in the top of the list. The last two linked entities come from advertisements and a magazine information column located at each side of the article. It is difficult to ignore them as

#	Name of the linked entity	Translation	Frequency	TF-IDF
1	hevonen	a horse	4	0.76
2	sota	war	3	0.22
3	matkat	travels	2	0.19
4	teltat	tents	2	0.19
5	mies, miehet	a man, men	3	0.16
6	tallit	stables	1	0.11
7	esikunta	military staff	2	0.11
8	korsut	dugouts	1	0.08
9	tykit	cannons	3	0.08
10	jonot	queues	1	0.08

Table 6.8: Annotation results for article: Minä ja Mustini Talvisodassa by Kirkola Nestori.

they both have understandable text that mixes with the article text. The other linked entities from 1 to 8 are relevant to the text. After a careful inspection of the article text it seems to be mostly valid information. The same goes for the linked entities of Table 6.8. The table, however, does not contain noise from the advertisements as the pages don't contain any. The advertisement issue concerns mainly the first and the last articles of the magazine but occasionally also the other articles.

All in all the outcome demonstrate the quality of the results. In general the annotation tool produced promising results but there are still a few issues that need more work. More advanced methods may be required to filter out the clutter that has been generated from the advertisements. Also the ability to recognize and ignore images in the OCR process might help to reduce clutter. These, however, are minor tweaks that can be done to improve the otherwise sound results of the annotation tool.

6.2 Semantic Finlex

The annotation process for Semantic Finlex was executed for 2,803 documents using the configurations and ontologies presented in Chapter 5. The execution produced a dataset that in total contains 51,789 annotations and 5,402 unique keywords. In addition the results of the tool needed to be evaluated independently using the R-Precision measure. For this purpose the annotation process was performed again by using only the FinlexVoc ontology and configured to produce comparable results to the original key-

words. The results of both executions of the application are presented in the following sections and starting from the evaluation of the result.

6.2.1 R-Precision

In order to measure the R-precision of the annotation for the law documents, the annotator was configured to use the same controlled vocabulary that was used manually in the original material. After the execution of the annotator, 30 documents were selected randomly and their keywords compared to the original keywords. The original annotations contained only from 1 to 14 keywords for each document. The majority of documents contained less than 6 keywords per document. The most common amount of keywords was 1, regardless of the length of the document.

The calculations for R-precision were done by selecting the same amount of keywords from the the automatically produced keywords as in the original keywords and comparing them. The keywords from the annotation tool result set were selected by picking the keywords that were evaluated by the weighting scheme as the most relevant to the document. The R-precision result is equal to the precision and recall measures when the amount of keywords for both sets used in the calculations is the same. In addition the precision and recall were not suitable measures for this case because the amount of keywords was fixed to the amount of keywords in the original annotations; this use case is more suitable for the R-precision measure. [63] The result of the R-precision calculation is 45.45 % for this result set.

The low amount of keywords in the original annotations has impacted the result of the R-precision calculations. For example, sometimes a keyword was found by the annotation tool but it was evaluated not relevant enough for the document. If the amount of keywords for a document would have been 5 instead of 1 in the original annotations, the keyword would have been included in the list of generated keywords. The results are, however, similar but not fully comparable to the results of Sinkkilä et al. [89] for different Finnish texts. The automatic annotation tool performed well in contrast to the tools and strategies used in the study. The precision and recall are slightly higher than what was produced using TF-IDF, FDG and other tools by the earlier study by Sinkkilä et al.

In addition, to analyze the precision and recall for the keywords of law documents, also the errors were analyzed and broken down into Table 6.9. The first two error groups (rows 1 and 2) consist of errors relating to the keyword relevancy that has been estimated using a weighting scheme by the application and identification of keywords from the text. These errors weight down the results the most. One explanation for these is that the original

#	Error type	Amount	Percentage
1	Keyword found but evaluated not relevant enough	20	29.85 %
2	Keywords not found in the document	14	20.90 %
3	Configuration error (language detection)	1	1.49 %
4	Source material error	1	1.49 %
5	Tool error	1	1.49 %
	TOTAL	37	55.22 %

Table 6.9: Error types found from the result set of the Semantic Finlex.

manual annotation was scarce and done without taking into consideration the length of each document. In terms of not finding keywords, the problem is that the keyword is not mentioned in the document.

The second group of errors (rows 3 - 5) consists mainly of configuration errors, tool errors, or errors related to the source materials. The errors relating to the configuration happen due to the tools inability to identify foreign language names. Solving the issue would require testing with different sets of configurations. Regarding the tool errors, the application in most cases can compare found matches for a piece of text and select the most suitable match from the pool of results for each ARPA query. This means that the application picks the longest text match it has found for the given text. For example, for a title *Indiana Jones and the Last Crusade* it does not match strings *Indiana Jones* and *Crusade* but selects the whole title of the movie, if available. Currently there are still some marginal cases where this is not done and that is what causes this error. Lastly there is one error caused by the source material. The error was caused by the letter S with a háček (š) in a name of a country. In the source material the háček was used in the law documents but not in the original annotations. Due to this small difference in spelling, one keyword was not found.

6.2.2 The Results of Subject Indexing

In addition to measuring the accuracy for the annotations, the tool produced a set of keywords using the given configuration described in chapter 5. The keywords were limited based on document length to have a maximum of 5 to 15 keywords depending on the length of the document. The range was selected based on the analysis of the material and manual annotations. I estimated that a little higher minimum would be better than the minimum of 1 in the original manual annotations and therefore selected 5 as the minimum. The maximum was selected based on the original annotations' maximum amount (14) but was raised to 15 to have a slightly bigger range for keyword

#	Keyword	Translation	Frequency	TF-IDF
1	neuvoston asetus	a council regulation	1	1.01
2	kumoaminen	repeal	1	0.68
3	Ulkoasiainministeriö	Ministry of Foreign Affairs in Finland	3	0.34
4	Iran	Iran	2	0.34
5	rikoslaki	a criminal code	3	0.34
6	Syyria	Syria	2	0.34
7	yhteisö	a community	1	0.34
8	Afganistan	Afghanistan	2	0.34
9	yritys, yritykset	a firm, firms	2	0.34
10	Eurooppa	Europe	1	0.34
11	henkilö, henkilöt	a person, people	1	0.34

Table 6.10: Keywords for document 2012/162: Ministry of Foreign Affairs in Finland notifies of the changes in punishment rules.

amounts. With this configuration the documents got from top 5 to top 15 keywords based on the length of the document. The longer the document, the more keywords were selected for the document.

In Tables 6.10 and 6.11 there are two randomly selected legal documents and their keywords. The first document is a notification of the Ministry of Foreign Affairs in Finland regarding changes in punishment rules. The second is a act about the handling of the welfare documents.

When annotating the documents, the amount of keywords selected was based on document length. In this case both documents have 11 unique keywords after the annotation process. In both tables the original names of the keywords are shown with their meaning translated next to the name. The term frequency column is next to the translation and followed by the TF-IDF column. The keywords are sorted in the table based on their estimated relevancy.

On closer inspection Table 6.10 contains 2 irrelevant keywords as the first two. They are more general legal terminology that are not necessarily relevant in describing the aboutness of the document content. In comparison to the original annotations, the document had 3 keywords: *Iran*, *Syyria* (Syria, in English) and *Rangaistukset* (Punishments, in English). Two of three are found from this result set and the third is filtered out due to too low TF-IDF score.

In Table 6.11 the document is too recent to have original annotations. The new annotations are listed in the array and initially seem satisfactory. However, on closer look they do not seem to fully describe the entire docu-

#	Keyword	Translation	Frequency	TF-IDF
1	asiakkaat	customers	18	0.74
2	sosiaalihuolto	welfare	21	0.65
3	palvelu, palvelut	a service, services	8	0.38
4	asiakirja	a document	16	0.32
5	päätös, päätökset	decision, decisions	6	0.18
6	käsittely	handling	5	0.15
7	sosiaalipalvelut	welfare services	12	0.15
8	kuolema	death	3	0.15
9	henkilö, henkilöt	a person, people	5	0.14
10	lapsi	a child	4	0.11
11	viranomainen	an authority	4	0.11

Table 6.11: Keywords for document 2015/254: Law of welfare documents.

ment. By looking at the application's execution logs for all the annotations found, it identifies them correctly but some of the keyword candidates have not been identified from the text as often as they should have. One such term is *asiakastiedot* (customer information, in English). The linguistics component LAS is unable to identify the term's base form and therefore its term frequency and TF-IDF score are calculated too low. The low scores caused the term to be filtered out of the top 11 result set.

Both documents have mainly good keywords but there is also room for improvement. It would be interesting to try with a bigger range like with the Helsinki City Library, from 3 to 30 keywords. In addition, a few new terms should be added to the stopword list to see how it would impact the results. All this is fine-tuning of the application configurations. In general, the application manages to produce satisfactory results.

Chapter 7

Conclusions

This thesis presents a new highly configurable and generic tool for annotating and subject indexing documents. It can be configured in multiple ways to produce semantic annotations for different Finnish texts. It links textual entities to matching concepts in controlled vocabularies of the user's choice and produces output in RDF format. For subject indexing, the application supports adding different evaluation methods and it supports multiple ways to define keyword density. The application also has a module that generates tag cloud visualizations of the keywords.

In this thesis there were two use cases for the application: Kansa Taisteli magazine and Semantic Finlex. For the Kansa Taisteli magazine the application produced semantic annotations that were linked to other ontologies of the WarSampo project. The data was used in the WarSampo portal in a separate Kansa Taisteli magazine perspective to search for documents based on a keyword. In the Semantic Finlex use case the application was used to provide the consolidated legislation with subject indexes. In addition, tag clouds were generated to provide the user a visual summarization of what the documents are about.

In the beginning of this thesis, a set of research questions were defined. In the following, questions are answers:

1. How to build a generic application for automatic annotation that can be configured for different use cases?

   The automatic annotation tool has been built and designed to be highly configurable. The tools and modules used in entity liking and linguistic processing offered many configuration opportunities. In addition, the application has more general configurations, such as RDF serialization setup, input, and output formats. The different weighting schemes can be added for the tool and the keyword density can be configured by

the user based on the source materials. These diverse configurations make it possible for the user to add their own ontologies and configure them freely. The user can also create rules to solve ambiguity and use different methods for candidate filtering, such as apply stopword lists.

2. How to utilize ontologies for natural language Finnish texts to get relevant annotations?

   The application processes natural language texts with the help of linguistic tools, and extracts textual entities from the text. These textual entities are identified and linked to ontologies by utilizing their semantic relations and information. Finally, the application estimates the relevancy in the given context. Keywords are picked by ranking the candidates from best to worst and based on the configurations a set of keywords is selected. In both use cases the success of the tool depended on the interpretation of the results. Compared to a human annotator the tool provides a richer amount of annotations.

3. How much disambiguation is needed and how to do it? What kind of problems are encountered when trying to solve ambiguity issues?

   Disambiguation of the keywords is a challenging task. The application uses configurations to do disambiguation. The selection and the order of ontologies can be used to remove ambiguity. For example, in Kansa Taisteli magazine articles the issue was approached by prioritizing the context specific ontologies. First the application searched for references to people and military units. Afterwards it focused on places and more general terminology. In addition, there are ontology specific configurations for determining if some concepts are better than others and need to be prioritized. These actions helped to minimize the amount of issues regarding ambiguity of terms. In case Kansa Taisteli, there remain two challenges. First, how to differentiate between last names and place names. Second, how to determine the places with more accuracy and use more efficiently the place information in the ontologies.

4. In regards to OCR, what is the impact of OCR'd text to the results? Is it possible to minimize the impact of errors?

   The OCR quality impacted the results for Kansa Taisteli magazine articles. A Semi-automatic handling of the results was required and as a byproduct a list of regular expressions was constructed to aid in

the correction of the errors. During the evaluation it was noticed that the post-processing of the OCR output improved the annotations and prevented erroneous annotations. However, there is still need for an improvement and developing an automatic set of rules could speed up the process of post-processing of OCR output.

In addition to improvements mentioned above, the application can benefit from future development. It requires more fine tuning and optimization. In order to utilize the application more efficiently it needs to be possible to run as a compact command line tool. Also a graphical user interface can be useful for the users and for testing purposes. In addition to these improvements, large scale testing is needed.

The application source code will be available for the public in the near future.

Bibliography

[1] ABBYY. *ABBYY FineReader Version 11 User's Guide*. `http://www.abbyy.com/finereader/11/user-guide/` (in English). Accessed 16 Apr 2016.

[2] Aebersold, R. PyTagCloud, 2013. `https://github.com/atizo/PyTagCloud`. Accessed 13 Oct 2016.

[3] Alkula, R. From plain character strings to meaningful words: Producing better full text databases for inflectional and compounding languages with morphological analysis software. *Information Retrieval 4*, 3-4 (2001), 195–208.

[4] Allen, J. *Natural Language Understanding*, 2nd ed. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1995.

[5] Anderson, J. D., et al. *Guidelines for indexes and related information retrieval devices*. NISO Press Bethesda, MD, 1997.

[6] Antoniou, G., and van Harmelen, F. *A Semantic Web Primer*, 3rd ed. MIT Press, Cambridge, MA, USA, 2004.

[7] Ayres, M.-L., Kilner, K., Fitch, K., and Scarvell, A. Report on the successful AustLit: Australian literature gateway implementation of the FRBR and INDECS event models, and implications for other FRBR implementations. Tech. rep., ERIC, 2002.

[8] Barker, K., and Cornacchia, N. Using noun phrase heads to extract document keyphrases. In *Conference of the Canadian Society for Computational Studies of Intelligence* (2000), Springer, pp. 40–52.

[9] Beckett, D. and Berners-Lee, T. and Prud'hommeaux, E. and Carothers, G. RDF 1.1 Turtle. Terse RDF Triple Language. W3C Recommendation 25 February 2014., 2014. `https://www.w3.org/TR/turtle/`. Accessed 05 Jun 2016.

[10] Bird, S., Klein, E., and Loper, E. *Natural Language Processing with Python*, 1st ed. O'Reilly Media, Inc., 2009.

[11] Bizer, C., Heath, T., and Berners-Lee, T. Linked data – the story so far. *International Journal on Semantic Web and Information Systems 5(3)* (2009), 1–22.

[12] Bojrs, U., Breslin, J. G., Finn, A., and Decker, S. Using the Semantic Web for linking and reusing data across web 2.0 communities. *Web Semant. 6*, 1 (Feb. 2008), 21–28.

[13] Boschetti, F., Romanello, M., Babeu, A., Bamman, D., and Crane, G. Improving OCR accuracy for classical critical editions. In *Research and Advanced Technology for Digital Libraries*. Springer, 2009, pp. 156–167.

[14] Brickley, D., and Guha, R. RDF Schema 1.1. W3C Recommendation 25 February 2014, 2014. `https://www.w3.org/TR/rdf-schema/`. Accessed 01 Jun 2016.

[15] Brinton, L. J. *The structure of modern English: A linguistic introduction*. John Benjamins Publishing, 2000.

[16] Bunescu, R. C., and Pasca, M. Using encyclopedic knowledge for named entity disambiguation. In *EACL* (2006), vol. 6, pp. 9–16.

[17] Chung, Y.-M., Pottenger, W. M., and Schatz, B. R. Automatic subject indexing using an associative neural network. In *Proceedings of the third ACM conference on Digital libraries* (1998), ACM, pp. 59–68.

[18] Cohen, A. M. Unsupervised gene/protein named entity normalization using automatically extracted dictionaries. In *Proceedings of the acl-ismb workshop on linking biological literature, ontologies and databases: Mining biological semantics* (2005), Association for Computational Linguistics, pp. 17–24.

[19] Committee on Cataloging. Task force on metadata. final report. Tech. rep., June 2000. `http://libraries.psu.edu/tas/jca/ccda/tf-meta6.html`. Accessed 30 May 2016.

[20] Cowie, J., and Lehnert, W. Information extraction. *Commun. ACM 39*, 1 (Jan. 1996), 80–91.

[21] Cucerzan, S. Large-scale named entity disambiguation based on Wikipedia data. In *EMNLP-CoNLL* (2007), vol. 7, pp. 708–716.

[22] Cyganiak, R., Wood, D., and Lanthaler, M. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 25 February 2014, 2014. `https://www.w3.org/TR/rdf11-concepts/`. Accessed 30 May 2016.

[23] Daiber, J., Jakob, M., Hokamp, C., and Mendes, P. N. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)* (2013).

[24] DBpedia. Language chapters, 2016. `http://wiki.dbpedia.org/about/language-chapters`. Accessed 02 Nov 2016.

[25] Dewey, M. *A Classification and subject index for cataloguing and arranging the books and pamphlets of a library.* Amherst, Massachusetts, 1876.

[26] English, J., Hearst, M., Sinha, R., Swearingen, K., and Lee, K. Flexible search and navigation using faceted metadata. Tech. rep., Technical report, University of Berkeley, School of Information Management and Systems., 2002.

[27] Fernández, S., Tejo, C., Herman, I., and Zakhlestin, A. SPARQL endpoint interface to python (1.7.6), 2016. WWW page of the SPARQLWrapper: `http://rdflib.github.io/sparqlwrapper/`. Accessed 09 Oct 2016.

[28] Fort, K., Nazarenko, A., and Rosset, S. Modeling the complexity of manual annotation tasks: a grid of analysis. In *International Conference on Computational Linguistics* (2012), pp. 895–910.

[29] Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., and Nevill-Manning, C. G. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (San Francisco, CA, USA, 1999), IJCAI '99, Morgan Kaufmann Publishers Inc., pp. 668–673.

[30] Frosterus, M., Tuominen, J., and Hyvönen, E. Facilitating re-use of legal data in applications – Finnish law as a linked open data service. In *Proceedings of the 27th International Conference on Legal Knowledge and Information Systems (JURIX 2014)* (December 2014), IOS Press, pp. 115–124.

[31] Gruber, T. R. A translation approach to portable ontology specifications. *Knowledge acquisition 5*, 2 (June 1993), 199–220.

[32] Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. R. Evaluating entity linking with Wikipedia. *Artificial Intelligence 194* (Jan. 2013), 130–150.

[33] Hallitus SSHS. Kansa Taisteli lehdet 1957–1986, 2014. WWW page of the Association for Military History in Finland: `http://www.sshs.fi/sitenews/view/-/nid/92/ngid/1`. Accessed 26 Nov 2015.

[34] Hawking, D., and Zobel, J. Does topic metadata help with web search? *Journal of the American Society for Information Science and Technology 58*, 5 (2007), 613–628.

[35] Hearst, M. A., and Rosner, D. Tag clouds: Data analysis tool or social signaller? In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual* (2008), IEEE, pp. 160–160.

[36] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation 11 December 2012, 2012. `https://www.w3.org/TR/owl2-primer/`. Accessed 01 Jun 2016.

[37] Holley, R. How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs. *D-Lib Magazine 15*, 3/4 (March/April 2009).

[38] Hollink, L., Schreiber, G., and Wielinga, B. Patterns of semantic relations to improve image content search. *Web Semantics: Science, Services and Agents on the World Wide Web 5*, 3 (2007), 195–203.

[39] Hulth, A., Karlgren, J., Jonsson, A., Boström, H., and Asker, L. Automatic keyword extraction using domain knowledge. In *International Conference on Intelligent Text Processing and Computational Linguistics* (2001), Springer, pp. 472–482.

[40] Hyvönen, E., Tuominen, J., Ikkala, E., and Mäkelä, E. Ontology services based on crowdsourcing: Case national gazetteer of historical places. In *Proceedings of 14th International Semantic Web Conference (ISWC 2015), Posters and Demos* (2015).

[41] Hyvönen, E. *Publishing and Using Cultural Heritage Linked Data on the Semantic Web*. Morgan & Claypool, Palo Alto, CA, USA, October 2012.

[42] Hyvönen, E., Heino, E., Leskinen, P., Ikkala, E., Koho, M., Tamper, M., Tuominen, J., and Mäkelä, E. Warsampo data service and semantic portal for publishing linked open data about the second world war history. In *The Semantic Web – Latest Advances and New Domains (ESWC 2016)* (May 2016), Springer-Verlag.

[43] Hyvönen, E., Mäkelä, E., Kauppinen, T., Alm, O., Kurki, J., Ruotsalo, T., Seppälä, K., Takala, J., Puputti, K., Heini Kuittinen, K. V., Tuominen, J., Tuomas Palonen, M. F., Sinkkilä, R., Paakkarinen, P., Laitio, J., and Nyberg, K. CultureSampo – a national publication system of cultural heritage on the semantic web 2.0. In *Proceedings of the 6th European Semantic Web Conference (ESWC2009), Heraklion, Greece* (May 31 – June 4 2009). Springer-Verlag.

[44] Imran, A. S., Rahadianti, L., Cheikh, F. A., and Yayilgan, S. Y. Semantic keyword selection for automatic video annotation. In *Signal-Image Technology & Internet-Based Systems (SITIS), 2013 International Conference on* (2013), IEEE, pp. 241–246.

[45] Isaac, A., and Summers, E. SKOS Simple Knowledge Organization System Primer. W3C Working Group Note 18 August 2009, 2009. `https://www.w3.org/TR/skos-primer/`. Accessed 01 Jun 2016.

[46] Ježek, E. *The Lexicon: An Introduction*. Oxford University Press, 2016.

[47] Jijkoun, V., Khalid, M. A., Marx, M., and De Rijke, M. Named entity normalization in user generated content. In *Proceedings of the second workshop on Analytics for noisy unstructured text data* (2008), ACM, pp. 23–30.

[48] Jurafsky, D., and Martin, J. H. *Speech and Language Processing.*, 2nd ed. 2009.

[49] Kansalliskirjasto. Finto, 2016. `https://finto.fi/fi/`. Accessed 02 Nov 2016.

[50] Kansalliskirjasto. Finto - suomalainen sanasto- ja ontologiapalvelu, 2016. `https://www.kiwi.fi/display/Finto/`. Accessed 02 Nov 2016.

[51] Kettunen, K., Kunttu, T., and Järvelin, K. To stem or lemmatize a highly inflectional language in a probabilistic IR environment? *Journal of Documentation 61*, 4 (2005), 476–496.

[52] Khalid, M. A., Jijkoun, V., and De Rijke, M. The impact of named entity normalization on information retrieval for question answering. In *European Conference on Information Retrieval* (2008), Springer, pp. 705–710.

[53] Kiryakov, A., Popov, B., Terziev, I., Manov, D., and Ognyanoff, D. Semantic annotation, indexing, and retrieval. *Web Semant. 2*, 1 (Dec. 2004), 49–79.

[54] Korenius, T., Laurikkala, J., Järvelin, K., and Juhola, M. Stemming and lemmatization in the clustering of Finnish text documents. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management* (2004), ACM, pp. 625–633.

[55] Kuo, B. Y., Hentrich, T., Good, B. M., and Wilkinson, M. D. Tag clouds for summarizing web search results. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 1203–1204.

[56] Lauser, B., and Hotho, A. Automatic multi-label subject indexing in a multilingual environment. In *Research and Advanced Technology for Digital Libraries*. Springer, 2003, pp. 140–151.

[57] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web 6*, 2 (2015), 167–195.

[58] Liu, Z., Li, P., Zheng, Y., and Sun, M. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1* (2009), Association for Computational Linguistics, pp. 257–266.

[59] Lo, R. T.-W., He, B., and Ounis, I. Automatically building a stopword list for an information retrieval system. In *Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR)* (2005), vol. 5, pp. 17–24.

[60] Löfberg, L., Archer, D., Piao, S., Rayson, P., McEnery, T., Varantola, K., and Juntunen, J.-P. Porting an English semantic tagger to the Finnish language. In *Proceedings of the Corpus Linguistics 2003 conference* (2003), pp. 457–464.

[61] Lounela, M. Exploring morphologically analysed text material. In *Inquiries into Words, Constraints and Contexts*. 2005, pp. 259–267.

[62] Malaisé, V., Hollink, L., Gazendam, L., et al. The interaction between automatic annotation and query expansion: a retrieval experiment on a large cultural heritage archive. *SemSearch 334* (2008), 44–58.

[63] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to information retrieval*, vol. 1st ed. Cambridge University Press, New York, NY, USA, 2008.

[64] Manning, C. D., and Schütze, H. *Foundations of statistical natural language processing*, 1st ed. MIT Press, 1999.

[65] Medelyan, O. *Human-competitive automatic topic indexing*. PhD thesis, The University of Waikato, 2009.

[66] Medelyan, O., and Witten, I. H. Thesaurus based automatic keyphrase indexing. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries* (2006), ACM, pp. 296–297.

[67] Mihalcea, R., and Tarau, P. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04 and the 2004 Conference on Empirical Methods in Natural Language Processing* (Barcelona, Spain, July 2004), Association for Computational Linguistics, pp. 404–411.

[68] Mithe, R., Indalkar, S., and Divekar, N. Optical character recognition. *International Journal of Recent Technology and Engineering (IJRTE) 2*, 1 (March 2013).

[69] Mäkelä, E. Combining a REST lexical analysis web service with SPARQL for mashup semantic annotation from text. In *The Semantic Web: ESWC 2014 Satellite Events, Revised Selected Papers* (May 2014), pp. 424–428.

[70] Mäkelä, E., Lindquist, T., and Hyvönen, E. CORE - a contextual reader based on linked data. In *Proceedings of Digital Humanities 2016, long papers* (July 2016), pp. 267–269.

[71] Nadeau, D., and Sekine, S. A survey of named entity recognition and classification. *Lingvisticae Investigationes 30*, 1 (2007), 3–26.

[72] National Institute of Education. *ERIC PROCESSING MANUAL. Rules and Guidelines for the Acquisition, Selection, and Technical Processing of Documents and Journal Articles by the Various Components of the ERIC Network*. U.S. Department of Education, 1992.

[73] Navarro, B., Izquierdo, R., and Saiz-Noeda, M. Exploiting semantic information for manual anaphoric annotation in cast3lb corpus. In *Proceedings of the 2004 ACL Workshop on Discourse Annotation* (2004), Association for Computational Linguistics, pp. 65–71.

[74] Nguyen, T. D., and Kan, M.-Y. Keyphrase extraction in scientific publications. In *Proceedings of the 10th International Conference on Asian Digital Libraries: Looking Back 10 Years and Forging New Frontiers* (Berlin, Heidelberg, 2007), ICADL'07, Springer-Verlag, pp. 317–326.

[75] Nicomsoft Ltd. *Optical Character Recognition (OCR) – How it works*, February 2012. WWW page of the Nicomsoft: `https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/`. Accessed 29 Oct 2015.

[76] Nothman, J., Ringland, N., Radford, W., Murphy, T., and Curran, J. R. Learning multilingual named entity recognition from Wikipedia. *Artif. Intell. 194* (Jan. 2013), 151–175.

[77] Oksanen, A. Lainsäädänön ja oikeusäytännön mallintaminen ja julkaiseminen linkitettynä avoimena datana (modeling and publishing legislation and case law as linked open data). Master's thesis, Aalto University, Department of Computer Science, 2016.

[78] Oren, E., Möller, K., Scerri, S., Handschuh, S., and Sintek, M. What are semantic annotations. *Technical Report. DERI Galway* (2006).

[79] Paik, J. H. A novel TF-IDF weighting scheme for effective ranking. In *Proceedings of the 36th International ACM SIGIR Conference on*

Research and Development in Information Retrieval (New York, NY, USA, 2013), SIGIR '13, ACM, pp. 343–352.

[80] Patel, C., Patel, A., and Patel, D. Optical character recognition by open source OCR tool Tesseract: A case study. *International Journal of Computer Applications 55*, 10 (October 2012), 50–56.

[81] Pirinen, T. A., and Cliath, O. C. B. Á. Omorfi—free and open source morphological lexical database for Finnish. In *Nordic Conference of Computational Linguistics NODALIDA 2015* (2015), p. 313.

[82] Pirjo Mikkonen, S. P. *Sukunimet*. Otavan kirjapaino Oy, 2000.

[83] Pouliquen, B., Steinberger, R., and Ignat, C. Automatic annotation of multilingual text collections with a conceptual thesaurus. In *Proceedings of the Workshop 'Ontologies and Information Extraction'* (2003), EUROLAN'2003, pp. 8–28.

[84] Richardson, L. BeautifulSoup 4.4.0 documentation, 2004-2015. WWW page of the BeautifulSoup: `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`. Accessed 09 Oct 2016.

[85] Riloff, E., and Jones, R. Learning dictionaries for information extraction by multi-level bootstrapping. In *in AAAI'99/IAAI'99 – Proceedings of the 16th National Conference on Artificial Intelligence & 11th Innovative Applications of Artificial Intelligence Conference* (1999), AAAI Press & MIT Press, pp. 474–479.

[86] Sanner, M. F., et al. Python: a programming language for software integration and development. *J Mol Graph Model 17*, 1 (1999), 57–61.

[87] Semantic Computing Research Group. Finnish Wikipedia as Linked Data (DBpedia), 2013. `http://www.ldf.fi/dataset/dbpedia-fi/`. Accessed 02 Nov 2016.

[88] SFS 5471. Guidelines for the establisment and maintenance of Finnish language thesauri. SFS standard, Finnish Standards Association, 1988.

[89] Sinkkilä, R., Suominen, O., and Hyvönen, E. Automatic semantic subject indexing of web documents in highly inflected languages. In *Extended Semantic Web Conference* (2011), Springer, pp. 215–229.

[90] Smith, R. An overview of the Tesseract OCR engine. In *icdar* (2007), IEEE, pp. 629–633.

[91] Smith, R. W. History of the Tesseract OCR engine: what worked and what didn't. In *IS&T/SPIE Electronic Imaging* (2013), International Society for Optics and Photonics, pp. 865802–865802.

[92] Studer, R., Benjamins, V. R., and Fensel, D. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering 25*, 1-2 (Mar. 1998), 161–197.

[93] Suominen, O. *Methods for Building Semantic Portals*. PhD thesis, Aalto University, 2013.

[94] Tapanainen, P., and Järvinen, T. A non-projective dependency parser. In *Proceedings of the Fifth Conference on Applied Natural Language Processing* (Stroudsburg, PA, USA, 1997), ANLC '97, Association for Computational Linguistics, pp. 64–71.

[95] The Publications Office of the European Union. EuroVoc, the EU's multilingual thesaurus. WWW page of the Multilingual Thesaurus of the European Union: `http://eurovoc.europa.eu/drupal/?q=fi`. Accessed 20 Sep 2016.

[96] The W3C SPARQL Working Group. SPARQL 1.1 Overview. W3C Recommendation 21 March 2013., 2013. `https://www.w3.org/TR/sparql11-overview/`. Accessed 30 May 2016.

[97] Tjong Kim Sang, E. F., and De Meulder, F. 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* (2003), Association for Computational Linguistics, pp. 142–147.

[98] Tolvanen, M. Subject indexing, Helsinki City Library, 2016. Interviewed using email on 18.10.2016.

[99] Turney, P. D. Learning algorithms for keyphrase extraction. *Information Retrieval 2*, 4 (2000), 303–336.

[100] Van Rossum, G. Python programming language. In *USENIX Annual Technical Conference* (2007), vol. 41.

[101] Volk, M., Marek, T., and Sennrich, R. Reducing OCR errors by combining two OCR systems. In *ECAI-2010 workshop on language technology for cultural heritage, social sciences, and humanities* (2010), pp. 61–65.

[102] Wahlroos, M. Indeksointimetatiedon eristäminen ja arviointi (extraction and evaluation of index metadata). Master's thesis, University of Helsinki, Department of Computer Science, February 2013.

[103] Walker, D. D., Lund, W. B., and Ringger, E. K. Evaluating models of latent document semantics in the presence of OCR errors. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (2010), Association for Computational Linguistics, pp. 240–250.

[104] Wentland, W., Knopp, J., Silberer, C., and Hartung, M. Building a multilingual lexical resource for named entity disambiguation, translation and transliteration. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)* (Marrakech, Morocco, may 2008), European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[105] Yamashita, T., and Matsumoto, Y. Language independent morphological analysis. In *Proceedings of the sixth conference on Applied natural language processing* (2000), Association for Computational Linguistics, pp. 232–238.

[106] Yimam, S. M., Biemann, C., Eckart de Castilho, R., and Gurevych, I. Automatic annotation suggestions and custom annotation layers in WebAnno. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (Baltimore, Maryland, June 2014), Association for Computational Linguistics, pp. 91–96.

[107] zdenop. Tesseract(1). GitHub home page for the Tesseract OCR tool. `https://github.com/tesseract-ocr/tesseract/blob/master/doc/tesseract.1.asc`. Accessed 16 Apr 2016.

Appendices

Appendix A

Regular Expressions for OCR Post-processing

```
find - linux command (http://linux.die.net/man/1/find)
sed - linux command (http://linux.die.net/man/1/sed)
Find with find command a file with a given file pattern. Then using
    sed replace (using flag i edits in place ( on the spot )) a
    searched pattern with a new string. This is where the regex
    pattern is added. It starts with s/ and followed by the replacing
    pattern after next /. Finally the regex ends with a /g. Add {} and
    \; to the end.

Examples below.

 find . -name "*.c" -exec sed -i "s/ SO /50/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/ SO /50/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/ SO /50/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l4.D/14.D/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zn/:n/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/Jspz/Jsp:/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/i //g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/kmz/km:/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l9/19/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l8/18/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l7/17/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l6/16/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l5/15/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l4/14/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l3/13/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/l2/12/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/I9/19/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/I7/17/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/I8/18/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/I6/16/g" '{}' \;
```

```
find . -name "*.tes.txt" -exec sed -i "s/I5/15/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/I4/14/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/|7/17/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/|8/18/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/|9/19/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/Â»/-/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/18l/181/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/19l/191/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/17l/171/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/16l/161/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/15l/151/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/13l/131/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/|3/13/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/I3/13/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/ ELP / Er.P /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/El".P / Er.P /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/EI".P / Er.P /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/jR /JR /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/lÅ¡r.P /JR /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/. kun /, kun /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/EI'.P / Er.P /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/ ELP/ Er.P /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/ .ja / ja /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/. mutta /, mutta /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/_ /- /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/:5tä /:stä /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/:5ta /:sta /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/-- /- /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/_- /- /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/H /- /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/I-I/H/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/. jolla /, jolla /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/ .jolla / jolla /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/ .kun / kun /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/ .ja / ja /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/ ia / ja /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/IMo /No /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/ rn / m /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/, j, kun/, ja kun/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/--/-/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/- /-/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/PUNAlSEN/PUNAISEN/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/l\/JR /I\/JR /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/zlle/:lle/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kum /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -lmm /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -ktwa /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Å-.um /SA-kuva /g" '{}'
    \;
```

```
find . -name "*.tes.txt" -exec sed -i "s/SA -ltm *a /SA-kuva /g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuw /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -imm /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -k u va /SA-kuva /g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kkt'a/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuva/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -ktwa/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -ltm */SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Å'M'a/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Å-.um/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -hu/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Å:n va/SA-kuva /g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Iiwa/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -luma/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Äwaa/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuva/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -lmm/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kum/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kwa/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Äwra/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Å'Jn'a/SA-kuva /g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/SA -bu'a/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kmu/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA - kuva/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA - lnwa/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -lnwa/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -Ån a/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuva /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -hwa/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA - lmm/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -k u va/SA-kuva /g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuva /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -hwa /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuma' /SA-kuva /g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuma /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA - lmm /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -ktwa /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuw /SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SA -kuw/SA-kuva /g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/16zn/16:n/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/39zn/39:n/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/26zn/26:n/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/6zn/6:n/g" '{}' \;
```

```
 find . -name "*.tes.txt" -exec sed -i "s/8zn/8:n/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/mmzn/mm:n/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zssa/:ssa/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zksi/:ksi/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zssä/:ssä/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zsta/:sta/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zeen/:een/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zlla/:lla/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zlle/:lle/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zllä/:llä/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zää/:ää/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zö/:ö/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zään/:ään/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/zstä/:stä/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/MO'I'I1/MOTTI/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/MO'I'T I/MOTTI/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/M AA LA IS ET/MAALAISET/g"
     '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/Iinnak/linnak/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/Iihaski/lihaski/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/YIi/Yli/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/Iinnoit/linnoit/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/Iievit/lievit/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/Iitteinä/litteinä/g" '{}' \;
 find . -name "*.tes.txt" -exec sed -i "s/Ii ää/lisää/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/l0ppu/loppu/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i
    "s/jalkaväkirykmcn/jalkaväkirykmen/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/J
    alkaväkirykmentti/Jalkaväkirykmentti/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/paranitar/paronitar/g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/Iahjapak/lahjapak/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/mikäbä/mikäpä/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/PAAMAJASSA/PÄÄMAJASSA/g"
    '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/PAAMAJA/PÄÄMAJA/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/JR l 1:n/JR 11:n/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/tchty/tehty/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/tch-ty/tehty/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/jätib/jätin/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/Iähi/lähi/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/SAKSALAISEl/SAKSALAISET/g"
    '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/I-vaää/Hyvää/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/tehOStetaan/tehostetaan/g"
    '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/AluejärjeStö/Aluejärjestö/g"
    '{}' \;
```

```
find . -name "*.tes.txt" -exec sed -i
    "s/se105tetuiksi/selostetuiksi/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i
    "s/Rannik-k0tyki5to'/Rannikkotykistö/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/nu0tioita/nuotioita/g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/su-juvasri/su-juvasti/g"
    '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/Äänettömänä/Äänettömänä/g"
    '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/divisioonicn/divisioonien/g"
    '{}' \;
find . -name "*.tes.txt" -exec sed -i
    "s/pääh yökkäykm/päähyökkäykset/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i
    "s/kenraali-luutnanttiA./kenraali-luutnantti A./g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/alikersantti
    MaunoAu-[a/alikersantti Mauno Aula/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/se-Iustaan/selustaan/g" '{}'
    \;
find . -name "*.tes.txt" -exec sed -i "s/TçHTÄVÄ/TEHTÄVÄ/g" '{}' \;
find . -name "*.tes.txt" -exec sed -i "s/l./1./g" '{}' \;
```

Appendix B

Kansa Taisteli Magazine: SPARQL Queries for ARPA

In this Appedix the SPARQL queries used in the Kansa Taisteli magazine articles annotation process are documented.

B.1 DBpedia

```
PREFIX text: <http://jena.apache.org/text#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dbpfi: <http://fi.dbpedia.org/resource/>
SELECT ?id ?label ?ngram ?source {

  # Values contains the query strings for this query.
  VALUES ?ngram {
    <VALUES>
  }

  # Capitalizing the initial letter,
  # and adding a language code (LANG) that
  # is defined in ARPA.
  BIND(STRLANG(CONCAT(UCASE(SUBSTR(?ngram,1,1)),
       LCASE(SUBSTR(?ngram,2))),<LANG>) AS ?label)

  ?id rdfs:label ?label .

  # filtering targeted categories of DBpedia ontology
```

```
  # to include war related categories such as WW2,
  # warfare, and war history categories.
  { ?id dct:subject/skos:broader*
       dbpfi:Luokka:Toinen_maailmansota . }
  UNION {
   ?id dct:subject/skos:broader*
      dbpfi:Luokka:Sodankäynti .
  } UNION {
    ?id dct:subject/skos:broader*
       dbpfi:Luokka:Sotahistorian_teemasivun_luokat .
  }

 # removing matches that are categories or properties.
  FILTER(!STRSTARTS(STR(?id),
         "http://fi.dbpedia.org/resource/Luokka:"))
  FILTER(!STRSTARTS(STR(?id),
         "http://fi.dbpedia.org/property/"))

}
```

B.2 KOKO Ontology

```
PREFIX text: <http://jena.apache.org/text#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?id ?label ?ngram ?upperClass ?upperClassLabel {
  VALUES ?ngram {
     <VALUES>
  }
  FILTER(!REGEX(?ngram, "^[0-9]+$"))
  BIND(STRLANG(LCASE(?ngram),"fi") AS ?label)

  GRAPH <http://www.yso.fi/onto/koko/> {
    ?id skos:prefLabel ?label .
  }
}
```

B.3 Military Units

```
PREFIX text: <http://jena.apache.org/text#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

```
PREFIX actors: <http://ldf.fi/warsa/actors/>
PREFIX atypes: <http://ldf.fi/warsa/actors/actor_types/>
SELECT ?id ?label ?ngram
WHERE
{
  # Values contains the query strings for this query.
  VALUES ?ngram {
    <VALUES>
  }

  # Filtering out the query strings shorter than two
  # characters, do not start with a capital letter
  # and remove special characters
  FILTER(STRLEN(?ngram)>2 &&
        UCASE(SUBSTR(?ngram,1,1))=SUBSTR(?ngram,1,1))
  BIND(CONCAT('"',REPLACE(?ngram,
       "([\\+\\-\\&\\|\\!\\(\\)\\{\\}\\[\\]\\^\\"\\~\\*\\?\\:])",
       "\\\\$1"),'"') AS ?qstring)

  # Quering for matches from a specific graph, checking
  # the military unit is in a sub category of the Military
  # Unit type and query for the label.
  GRAPH <http://ldf.fi/warsa/actors> { ?id text:query ?qstring . }
  ?id a/rdfs:subClassOf* atypes:MilitaryUnit .
  ?id rdfs:label|skos:prefLabel|skos:altLabel ?label .

  # Filter out the matches that do not match the original
  # query string in lower case.
  FILTER(LCASE(STR(?label))=LCASE(STR(?ngram)))
}
```

B.4 Person

```
PREFIX text: <http://jena.apache.org/text#>
PREFIX apf: <http://jena.hpl.hp.com/ARQ/property#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX actors: <http://ldf.fi/warsa/actors/>
PREFIX etype: <http://ldf.fi/warsa/events/event_types/>
PREFIX crm: <http://www.cidoc-crm.org/cidoc-crm/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX cas: <http://ldf.fi/schema/narc-menehtyneet1939-45/>

SELECT DISTINCT ?id ?ngram (COALESCE(?nlabel, ?plabel) AS ?label)
       ?rank_label ?etunimet ?sukunimi
```

```
WHERE
{
    # Values contains the query strings for this query.
    VALUES ?ngram {
        <VALUES>
    }
   FILTER(STRLEN(?ngram)>3 && REGEX(?ngram,"^[A-ZÅÄÖ]"))
    # A person can be found only by using the
    # forenames/ranks and surname.

    # Assumption:  surname is always in the end
    # underscrore can be added to the surnames
    # in order to identify if the surname has two parts
    BIND("^((?:[a-zA-ZäÄåÅöÖ-]\\.[ ]*)|
        (?:[a-zA-ZäÄöÖåÅèü-]{3,}[ ]+))
        ((?:[a-zA-ZäÄåÅöÖ-]\\.[ ]*)|
         (?:[a-zA-ZäÄöÖåÅèü-]{3,}[ ]+))?
        ((?:[a-zA-ZäÄåÅöÖ-]\\.[ ]*)|
         (?:[a-zA-ZäÄöÖåÅèü-]{3,}[ ]+))*
        ([_a-zA-ZäÄöÖåÅèü-]{3,})$" AS ?nimiREGEX)
    BIND(UCASE(REPLACE(REPLACE(?ngram, ?nimiREGEX, "$4"),
         "_", " ")) AS ?sukunimi)

    # First forename / rank or second forename/first
    # forename (and removing extra spaces)
    BIND(REPLACE(REPLACE(?ngram, ?nimiREGEX, "$1"),
         "^(.*?)[ ]*$", "$1") AS ?ngrametu)
    BIND(REPLACE(REPLACE(?ngram, ?nimiREGEX, "$2"),
         "^([.a-zA-ZäÄöÖåÅ-]*?)[ ]*$", "$1") AS ?ngramkeski)

    # Must have forename / rank or initials of the forename
    # Name must be at least 3 letters long.
    FILTER(REGEX(?ngrametu,
          "(^[A-ZÄÖÅ]\\.$)|(^[a-zA-ZäÄöÖåÅèü-]{3,}$)"))

    # Either only one forename or the same terms as for the
    # others.
    FILTER(REGEX(?ngramkeski,
          "(^$)|(^[A-ZÄÖÅ]\\.$)|(^[a-zA-ZäÄöÖåÅ-]{3,}$)"))
    BIND(UCASE(?ngrametu) AS ?etu)

    #Cannot accept the form E. forename surname.
    FILTER(?ngramkeski="" || !(STRENDS(?ngrametu, ".") &&
           !STRENDS(?ngramkeski, ".")))
    BIND(UCASE(?ngramkeski) AS ?keski)
    BIND(CONCAT('"',?sukunimi,'"') AS ?qstring)

    GRAPH <http://ldf.fi/warsa/actors>
          { ?id text:query ?qstring . }
```

```
 ?id foaf:familyName ?familyName .
 FILTER(?sukunimi = UCASE(?familyName))
 ?id skos:prefLabel ?plabel .
 OPTIONAL { ?id foaf:firstName ?etunimet . }
 BIND(CONCAT(?etunimet, ' ', ?familyName) AS ?nlabel)
 OPTIONAL {
     ?promotion_id a etype:Promotion ;
         crm:P11_had_participant ?id ;
         actors:hasRank ?promotion_rank_id ;
         crm:P4_has_time-span ?timespan_id .
     ?promotion_rank_id skos:prefLabel ?promotion_rank .
     ?timespan_id crm:P82a_begin_of_the_begin
                 ?earliest_promotion_time .
 }

 # The forenames must be found from the list of forenames
 # The initials must be the first letter of any forename
 # or the forename must be a rank.
 BIND(CONCAT("(^|[ ])", substr(?etu, 1, 1)) as ?etukirjainre)
 BIND(CONCAT("(^|[ ])", substr(?keski, 1, 1))
      as ?keskikirjainre)
 BIND(CONCAT("(^|[ ])", ?etu, "($|[ ])") AS ?etunimire)
 BIND(CONCAT("(^|[ ])", ?keski, "($|[ ])") AS ?toinennimire)
 BIND(IF(STRLEN(?keski)=2, ?keskikirjainre,
      IF(?keski="", ".", ?toinennimire)) AS ?keskire)
 BIND(IF(STRLEN(?etu)=2, ?etukirjainre, ?etunimire)
      AS ?eture)
 BIND(CONCAT("(^|[ ])", ?etu, " ", ?keski, "($|[ ])")
      AS ?longrankre)
BIND(REGEX(?etu, "MINISTERI$") AS ?minister_test)
FILTER(IF(?minister_test,
   NOT EXISTS { ?id a
   <http://ldf.fi/warsa/actors/actor_types/MilitaryPerson> .
   },TRUE)
)
BIND((REGEX(?promotion_rank, ?eture, "i") ||
     REGEX(?rank_label, ?eture, "i")) AS ?rank_test)
BIND((REGEX(?promotion_rank, ?longrankre, "i") ||
     REGEX(?rank_label, ?longrankre, "i")) AS ?long_rank_test)
 FILTER(
     IF(STRENDS(?etu, "."),

 # The initials of the forename (this must be taken
 # separately because initials are not comparable with the
 # ranks.
     REGEX(?etunimet, ?etukirjainre) &&
     REGEX(?etunimet, ?keskire),
     # else
     # Full name or rank or "ministeri"
```

```
            (REGEX(?etunimet, ?keskire, "i") &&
            (REGEX(?etunimet, ?eture, "i"))) ||
                IF(?keski != "",
                  ?rank_test &&
                    REGEX(?etunimet, ?keskire, "i")
                     || ?long_rank_test ||
                    (?minister_test &&
                    REGEX(?etunimet, ?keskire, "i")),
                # else
                ?minister_test ||
                    ?rank_test ||
                    ?long_rank_test
                )
        )
    )
}
```

B.5 Places

```
PREFIX text: <http://jena.apache.org/text#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX ptype: <http://ldf.fi/warsa/places/place_types/>
PREFIX hipla: <http://ldf.fi/schema/hipla/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?id ?label ?ngram ?type
FROM <http://ldf.fi/warsa/places/municipalities>
FROM <http://ldf.fi/warsa/places/karelian_places>
{
  # Values contains the query strings for this query.
  VALUES ?ngram {
    <VALUES>
  }

  # Filtering out the query strings that are shorter
  # than two characters, remove special characters,
  # and allow only terms with a capital first letter.
  FILTER(STRLEN(?ngram)>2 && REGEX(?ngram,"^[A-ZÅÄÖ]"))
  BIND(REPLACE(?ngram,
  "([\\//\\#\\'\\<\\>\\+\\-\\&\\|\\!\\(\\)\\{\\}\\[\\]\\^\\"\\~\\*\\?\\:\\\\])",
  "\\\\$1") AS ?qstring)

  # Query for matching labels that are of the type village,
  # town/city or a municipality.
  ?id text:query ?qstring.
  ?id a ?type .
```

```
  FILTER(?type=ptype:Kyla||
  ptype:Kirkonkyla_kaupunki||
  ?type=<http://www.yso.fi/onto/suo/kunta>)
  ?id rdfs:label|skos:prefLabel ?label .

  # Filter out the matches that do not match the original
  # query string in lower case.
  FILTER(LCASE(STR(?label))=LCASE(STR(?qstring)))
}
```

Appendix C

Semantic Finlex: SPARQL Queries for ARPA

In this Appedix the SPARQL queries used in the Semantic Finlex annotation process are documented.

C.1 Combined Legal Concept Ontology

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dbpfi: <http://fi.dbpedia.org/resource/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?id ?label ?ngram ?source ?len {
  {
    {
      # Values contains the query strings for this query
      VALUES ?ngram {
      <VALUES>
      }

      # Filtering out the querystrings that contain
      # only numbers, capitalizing the initial letter,
      # and adding a language code (LANG) that
      # is defined in ARPA.
      FILTER(!regex(?ngram, "^[0-9]+$"))
      BIND(STRLANG(CONCAT(UCASE(SUBSTR(?ngram,1,1)),
           LCASE(SUBSTR(?ngram,2))),<LANG>) AS ?ungram)

      # Query for upper case matches
      ?id skos:prefLabel ?ungram .
      BIND(?ungram AS ?label)
```

```
  #Calculating the length of label
  BIND(STRLEN(?label) AS ?len)
} UNION {
  # Query for lower case matches
  VALUES ?ngram {
  <VALUES>
  }
  FILTER(!regex(?ngram, "^[0-9]+$"))
  BIND(STRLANG(LCASE(?ngram),<LANG>) AS ?lngram)
  ?id skos:prefLabel ?lngram .
  BIND(?lngram AS ?label)

  #Calculating the length of label
  BIND(STRLEN(?label) AS ?len)
} UNION {
  # Query for upper case matches of equivalent classes
  VALUES ?ngram {
  <VALUES>
  }

  FILTER(!regex(?ngram, "^[0-9]+$"))
  BIND(STRLANG(CONCAT(UCASE(SUBSTR(?ngram,1,1)),
       LCASE(SUBSTR(?ngram,2))),<LANG>) AS ?ungram)

  ?id2 skos:prefLabel ?ungram .
  ?id owl:equivalentClass ?id2 .
  BIND(?ungram AS ?label)

  #Calculating the length of label
  BIND(STRLEN(?label) AS ?len)
} UNION {
# Query for lower case matches of equivalent classes
  VALUES ?ngram {
  <VALUES>
  }
  FILTER(!regex(?ngram, "^[0-9]+$"))
  BIND(STRLANG(LCASE(?ngram),<LANG>) AS ?lngram)
  ?id2 skos:prefLabel ?lngram .
  ?id owl:equivalentClass ?id2 .
  BIND(?lngram AS ?label)

  #Calculating the length of label
  BIND(STRLEN(?label) AS ?len)
}
# Clarifying URLs of the found matches to
# differentiate between sources.
BIND(IF(STRSTARTS(STR(?id),
    "http://www.yso.fi/onto/laki/"), 1,
```

```
        IF(STRSTARTS(STR(?id),"http://ldf.fi/ttp/"),2,0))
        AS ?source)
    FILTER(?source!=0)
  }
} ORDER BY DESC(?len)
LIMIT 1
```

C.2 DBpedia

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dbpfi: <http://fi.dbpedia.org/resource/>

      # Values contains the query strings for this query
      SELECT ?id ?label ?ngram ?source {
        VALUES ?ngram {
          <VALUES>
        }

        # Filtering out the querystrings that contain
        # only numbers, capitalizing the initial letter,
        # and adding a language code (LANG) that
        # is defined in ARPA.
      FILTER(!regex(?ngram, "^[0-9]+$"))
        BIND(STRLANG(CONCAT(UCASE(SUBSTR(?ngram,1,1)),
             LCASE(SUBSTR(?ngram,2))),<LANG>) AS ?label)

        # Quering for matches from the law category
        ?id rdfs:label ?label .
        ?id dct:subject/skos:broader* dbpfi:Luokka:Oikeustiede .
        # QUERY
        BIND(3 AS ?source)
}
```

C.3 EuroVoc

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX skosxl: <http://www.w3.org/2008/05/skos-xl#>
SELECT ?id ?label ?ngram
# Targetting a specific graph
FROM <http://data.finlex.fi/voc/eurovoc> {
```

```
  # Values contains the query strings for this query
  VALUES ?ngram {
    <VALUES>
  }

  # Query for the matches that have a
  # language code specified in ARPA.
  BIND(STRLANG(?ngram,<LANG>) AS ?label)
  ?id skos:prefLabel|skos:altLabel ?label .
}
```

C.4 KOKO Ontology

```
PREFIX text: <http://jena.apache.org/text#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?id ?label ?ngram {
  # Values contains the query strings for this query
  VALUES ?ngram {
     <VALUES>
  }

  # Filtering out the querystrings that contain
  # only numbers, capitalizing the initial letter,
  # and adding a language code (LANG) that
  # is defined in ARPA.
  FILTER(!regex(?ngram, "^[0-9]+$"))
  BIND(STRLANG(LCASE(?ngram),"fi") AS ?label)

  # Targetting a specific graph to query for matches
  GRAPH <http://www.yso.fi/onto/koko/> {
    ?id skos:prefLabel ?label .
  }
}
```

C.5 FinlexVoc

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX afn: <http://jena.hpl.hp.com/ARQ/function#>
```

```
SELECT ?id ?label ?ngram ?len {
  # Values contains the query strings for this query
  VALUES ?ngram {
    <VALUES>
  }
  # Capitalizing the initial letter,
  # and adding a language code (LANG) that
  # is defined in ARPA.
  BIND(strlang(concat(ucase(substr(?ngram,1,1)),
       lcase(substr(?ngram,2))), <LANG>) AS ?label)

  # Targetting a specific graph to query for matches
  # and filtering matches based on namespace.
  GRAPH <http://ldf.fi/finlex/voc> {
    ?id skos:prefLabel ?label .
    FILTER (afn:namespace(?id) =
           "http://data.finlex.fi/voc/finlex/")

    #Calculating the length of label
    BIND(STRLEN(STR(?label)) AS ?len)
  }
} ORDER BY DESC(?len)
```