UNIVERSITY OF OSLO
Department of Informatics

Wordnet Semantics From Dictionaries

Semi-automatic Extraction of Semantic Relations Between Verbs From a Dictionary

Master thesis

Rune Lain Knudsen

Autumn 2012

Acknowledgements

I am indebted to a number of people that, directly and indirectly, contributed to the creation, development and finalizing of this thesis. **Erik Velldal**, for being my patient and understanding supervisor for most of my time as a master's student, for providing me with a substantial amount of input on all aspects of my thesis, and for bringing me down to earth whenever I got too unrealistic as to what one can achieve during a well-defined timespan. I seem to miscalculate such things frequently. **Ruth E. Vatvedt Fjeld**, for being my co-supervisor throughout my master's programme, my vast knowledgebase for everything related to the field of lexicography, and for leading me onto the path of wordnets in the first place. **Lilja Øvrelid**, for being my co-supervisor for the first parts of my thesis, and for the last semester bravely stepping in as my main supervisor when Velldal was on child leave. A special thanks goes to **Julie Matilde Torjusen** and **Lilja Øvrelid** for being central participants in the annotation study. A very important part of this thesis relies on the post-processing work done by Torjusen and Øvrelid. **The Language Technology Group** at IFI, UiO, for a friendly and motivational environment, inspirational seminars and an important source of knowledge for everything related to the field of NLP. **The Text Laboratory**, UiO, for their work on the excellent OBT+Stat tagger.

I would also like to thank my fellow students and friends **Emanuele Lapponi** and **Lars-Erik Bruce**, for providing me with inspiration, help and companionship whenever I was around. I apologize for my absence at times.

Last but not least, I would like to thank my family and all my friends for constantly reminding me that there is in fact a world outside of my desktop computer.

Contents

List of Figures

List of Tables

Part I

Introduction

Wordnets are used as components in a wide range of applications, especially ones related to the many different tasks of natural-language processing. Information retrieval (Voorhees, 1998), machine translation, intelligent spell checking (Hirst and St-Onge, 1998), word-sense disambiguation (Banerjee and Pedersen, 2003), automatic text analysis, common-sense reasoning (Harabagiu and Moldovan, 1998) etc. are some examples of fields that benefit from wordnets. Princeton WordNet, a project that started its development under the direction of George A. Miller in 1985, is widely in use today and constantly undergoing further development. Several wordnets for other languages have followed since the incubation of Princeton WordNet (Lindén and Carlson, 2010; Pedersen et al., 2011; Åke Viberg et al., 2002), along with efforts to unify wordnets into even more complex structures that model inter-lingual relationships between different wordnets(Vossen, 2002; Tufiş et al., 2004).

1 Motivation and Goals

The motivation for this master thesis originates from Lars Nygaard's cand. philol. thesis from 2006 (Nygaard, 2006), from which a prototype for a Norwegian wordnet was generated by analyzing definitions for nouns in Bokmålsordboka (BOB)[1]. BOB is a dictionary for Norwegian Bokmål which is available both in book format and as an on-line resource.

This thesis will propose a set of components for a method that is in some respects an extension of Nygaard's method, in other respects a different approach altogether. An investigation will be done for every stage of the process, including the extraction and analysis of the dictionary data, the generation of a semantic network and the evaluation phase of such a semantic network. The proposed method will be tested on a dictionary for Norwegian Bokmål, and a thorough annotation study will be presented in order to clarify the terms for evaluating a semi-automatically generated semantic network.

The main focus of study is on the task of generating wordnet data from verb definitions. This is an area not covered by Nygaard's method, and is generally a subject of study not as frequently covered as nouns. Verbs exhibit different behavior than nouns in many respects, therefore they should present a somewhat different set of challenges.

The goal is to investigate to which extent a method for the described purpose can be automated, and to which extent it can be said to be general.

The observations made throughout the thesis will be analyzed in the attempt to gain insights into lexical semantics as well as verbs, both with regard to semantic properties and the challenges one encounters when modeling verbs in a wordnet.

2 Thesis Structure

The thesis is structured as follows:

Chapter 1 introduces some terminology and gives a general overview of wordnets and dictionaries that supplies us with some of the theoretical foundations needed for the rest of the thesis.

Chapter 2 presents two of the earlier attempts of semi-automatic generation of wordnets from dictionary information and remarks on those.

[1] Nygaard's method is covered in Section 2

Chapter 3 describes the proposed method through a series of steps, all of which are covered in the following chapters up to Chapter 8.

Chapter 8 presents a series of evaluations based on our observations from the earlier chapters.

Chapter 9 presents some concluding remarks about the method, the evaluation process and further work.

3 Clarifications and Caveats

There are some definitions and assumptions that must be stated before moving on to the rest of the thesis. The reader may consider the definitions and clarifications in this section to hold for the rest of the thesis unless specifically stated otherwise.

Definiendum and Definiens The terms *definiendum* (plural *definienda*) and *definiens* (plural *definientia*) need to be clarified. We define them as follows:

Definiendum That which is to be defined. Represented as the lemma form of some word.

Definiens That which defines. We restrict the notion of a definiens to the explanatory part of a definition for some definiendum.

Notation for Semantic Relations The notation for a semantic relation is presented with the example *springe (run)* HAS_SYNONYM *løpe (run)*. The left hand side of the relation is referred to as the LHS , while RHS refers to the right hand side of the relation. The LHS and RHS are always represented in lemma form.

To enhance the readability of tables and figures throughout the thesis, semantic relations are often abbreviated to three-letter representations. These abbreviations are listed in Table 1.

Full	Abbreviated
HAS_SYNONYM	SYN
HAS_HYPERONYM	HYP
INVOLVED	INV
CAUSES	CAU
ENTAILS	ENT
ANTONYM	ANT

Table 1: Abbreviations for semantic relations.

Using the previous example of the notation for a semantic relation, the abbreviated version of this relation is *springe* SYN *løpe*. The *set of relation symbols* corresponds to the instances listed in Table 1. The term *relation symbol* refers to a member of the set of relation symbols, both the full version and the abbreviated version.

Part II

Background

Chapter 1

Central Concepts for Wordnets and Dictionaries

This chapter provides background information and terminology on wordnets and dictionaries in general that is needed in order for the rest of the thesis to make sense. Section 1 describes wordnet terminology and presents some wordnets that are deemed relevant to this thesis. Section 3 presents lexicographic terminology and some general background information regarding dictionaries.

1 Wordnets

A wordnet is a lexical database whose structure was originally inspired by theories on how knowledge about words and concepts might be organized in the human mind (Fellbaum, 1998c, p. 29-34). The fundamental structure of a wordnet is modeled as a network of semantic relations, mostly between *lexicalized concepts*. A lexicalized concept is a concept that can be expressed by a word or phrase. Lexical items are single words, or chains of words, that make up the basic elements of a lexicon. Several *lexical items* may refer to the same lexical concept, which in turn can be referred to by an expression acting as a common word for the collection of lexical items in question.

1.1 Synsets

In a wordnet, a lexicalized concept is modeled as a set of words (usually represented by their lemma forms) that are considered to have loose synonymy. , e.g. { *help, assist, aid* }. Loose synonymy holds for words that *can be interchanged in some contexts* (Miller, 1998, p. 23-24), hence the notion of synonym sets, or *synsets*. Strict synonymy is regarded as too restrictive a definition to be used for this purpose. To claim that two concepts are strictly synonymous is to imply that the two concepts can replace eachother in all contexts without changing the meaning of the context. This will exclude a very large set of concepts that intuitively have a synonymous relationship, and it is therefore not a practical definition for this field of study.

One lemma can be part of several synsets as polysemous lemmas are split into their different meanings. As an example, the verb "call" has 28 senses in Princeton WordNet, three of which are { *call, telephone, call up, phone, ring* }, { *shout, shout out, cry, call, yell, scream, holler, hollo, squall* } and { *name, call* }.

1.2 Semantic Relations

To connect synsets in a meaningful way, a set of semantic relations are used that describe how two concepts increase and/or specialize eachothers meanings. Below is a list of some of the most common semantic relations used in wordnets (Saeed, 1997; Fellbaum, 1998c; Alonge et al., 1998). The examples for each relation are gathered from searches performed in the WordNet3.0 command line application available from `http://wordnet.princeton.edu/wordnet/download/`. Most words are part of larger synsets but are represented as one word only for the sake of clarity.

Synonymy Relates two concepts using the definition of synonymy as defined in Section 1.1. This is a symmetric and reflexive relation. Example: *kind* is a synonym of *benign*.

Hyperonymy Describes a typical *is-a* relationship, indicating that one concept subsumes another. Example: *interact* is a hyperonym of *communicate*, which is a hyperonym of *utter*. This is a transitive relation, meaning that *interact* also is a hyperonym of *utter*.

Hyponymy The opposite of hyperonymy. Example: *utter* is a hyponym of *communicate*.

Troponomy A relation between verbs that corresponds to the hyponymy relation, with some differences (see Section 2.1 for details.)

Antonomy Relates two lexical opposites. Example: *evil* is an antonym of *good*.

Meronymy Used to describe a part-whole relationship. Example: *pad* is a meronym of *paw*, which is a meronym of *feline*.

Holonomy The opposite of meronymy. Example: *feline* is a holonym of *paw*.

Entailment Denotes one concept as a prerequisite for another. Resembles logical entailment but with looser restrictions. Example: To *snore* entails to *sleep*.

Cause Implies a causal relationship between two concepts. Example: to *kill* causes something to *die*.

1.3 Ontology

The synsets and their relations give rise to an *ontology* describing world knowledge as a hierarchy of concepts, entities and ideas. In many wordnets a *top ontology*, or *upper ontology*, is explicitly defined in an attempt to enforce interoperability between the lower levels of the semantic network. The set of concepts that make up the top ontology are very general and in many cases quite abstract. Concepts like *physical*, *abstract*, *quantity*, *agent* and *relation*[1]tend to be a part of a top ontology in one way or another. Choosing an appropriate upper ontology is not trivial and has been subject to much debate. Part of the problem lies in the fact that there is no consentual, objective definition for what an ontology is, hence the expectations and requirements tend to differ between institutions and research areas. Princeton WordNet operate with 11 synsets that are defined as *unique beginners* for nouns; synsets that have no hyperonyms themselves and under which all other synsets are organized into hyponym hierarchies (Miller, 1998, p. 28-29). EuroWordNet, which is presented in Section 2.2, has only three, very abstract, unique beginners (1stOrderEntity, 2ndOrderEntity and 3rdOrderEntity).

[1]The examples are taken from the Suggested Upper Merged Ontology (SUMO) and can be explored in more detail at `http://www.ontologyportal.org/`. This is the largest public formal ontology available, it is owned by the IEEE and it is mapped to the whole of WordNet3.0.

1.4 Gloss

A *gloss* is usually attached to each concept, briefly explaining the meaning of the concept using natural language. This resembles the definition text for an entry in a conventional dictionary but does not contain the lexicographic notations usually found with it. In addition, one synset can have only one gloss, reflecting the fact that each synset refers to exactly one lexicalized or non-lexicalized concept (unlike a dictionary, where one lemma often has multiple definitions in the same lexical entry). As an example, the gloss for one of the meanings of the noun *car* in Princeton WordNet is "a motor vehicle with four wheels; usually propelled by an internal combustion engine".

2 Four Wordnets and Their Properties

Wordnets are being developed for a multitude of languages all over the world. The Global Wordnet Association (Glo) (GWA) maintains a list of wordnets that conform to their standards on `http://www.globalwordnet.org/gwa/wordnet_table.htm`. Currently 64 wordnets are on this list but the actual number is probably higher if related semantic networks and projects that have not been in contact with GWA is included. I will focus on the four wordnets mentioned in the introduction and a selection of their properties and methodology that I find relevant for the purpose of this essay and my thesis. Princeton WordNet is presented since it is the first wordnet ever made, and as thus has been a major influence for all subsequent wordnet projects. EuroWordNet is a major effort in the task of unifying wordnets for different languages, and has a substantial extension of relation types. DanNet is presented as it represents the Danish language, a language that is closely related to Norwegian Bokmål. In addition, it is one of the wordnets that are modeled in close relationship to an existing dictionary, in this case Den Danske Ordbog(Den). NorNet is the prototype for a Norwegian wordnet and has been generated from Nygaard's method, and as such is highly relevant.

2.1 Princeton WordNet

Princeton WordNet (PWN) has been under continuous development since its birth in 1985. It has served as the foundation for the development of the theory, architecture and methodology for later wordnet projects (Fellbaum, 1998b).

Verbs in PWN

The main relation for verbs in PWN is the troponymy relation, which is defined as the verb equivalent of the hyponymy relation. The reason for distinguishing between hyponymy and troponymy is discussed in (Fellbaum and Miller, 1990), and specifically related to PWN in (Fellbaum, 1998a). The troponymy relation holds if the sentence *to* V_1 *is to* V_2 *in some particular manner* is true (e.g. *shout* HAS_TROPONYM *bawl* implies that to *bawl* is to *shout* in some particular manner), creating the foundation for a hierarchy of more and more specific verbs much in the same fashion as for noun hyponyms. The troponymy relation also represents a special form of entailment from V_1 to V_2 as can be seen in e.g. the relation *talk* HAS_TROPONYM *whisper* where *whisper* entails *talk*. Lastly, a troponomy relation also should satisfy a condition of temporal coextensivity, meaning that the troponym of a word should occupy the same timespan as its hyperonym (e.g. *walk* HAS_TROPONYM *march*).

Unique Beginners Finding appropriate unique beginners is arguably even more difficult for verbs than for nouns. Likely candidates tend to have a high degree of polysemy and makes it difficult to determine which sense should be the unique beginner and which should be put below it in the hierarchy. PWN has partially solved this problem by separating lexicalized verb concepts into different domains, effectively creating several hierarchies for verbs according to top-level concepts like possession, social interaction, movement and so forth.

2.2 EuroWordNet

EuroWordNet (EWN) is a project which aim is to construct a multilingual database containing wordnets for European languages. EWN currently consists of language-specific wordnets representing the Dutch, Spanish, Italian, English, French, German, Czech and Estonian languages. The language-specific wordnets are linked to an *inter-lingual index (ILI)*, an unstructured list of concepts initially based on Princeton WordNet1.5 (Vossen, 2002). Each synset in the language-specific wordnets is mapped to one or more concepts in the inter-lingual index according to sense equality.

In addition to the ILI, a *domain ontology*, a set of *common base concepts* and a *top concept ontology* has been developed. The Domain Ontology is a hierarchical model of topics grouping concepts under terms like *traffic*, *hospital* and so forth. The Common Base Concepts are concepts that are derived from *base concepts* found in the various wordnets that make up EWN. The Base Concepts are synsets that are selected locally in every wordnet based on a high number of relations to other concepts in the same wordnet and a high position in the ontological hierarchy. These synsets are compared to the base concepts found in the other wordnets. Base concepts that are found in two or more wordnets and that are regarded to have a sufficiently equivalent meaning make up a collection of synsets called the common base concepts. According to the EWN General Document (Vossen, 2002, p. 55) there are 1310 common base concepts in total. The top concept ontology organizes the common base concepts into a hierarchy. By linking the language-dependent wordnets to the ILI, the ILI to the Common Base Concepts and the Common Base Concepts to the Top Concept Ontology, a framework for a common structure for several languages is formed. This framework is also easily extended with other language-neutral ontologies (e.g. expert systems, knowledge bases, common sense repositories etc.) which in turn extend the individual wordnets since they can access this information through their links to the ILI.

Some examples of synsets linked to the inter-lingual index are given in Table 1.1. The original table can be found in Vossen (2002, p. 41). As shown in the second example for the Dutch synset mapping, many-to-many relations are possible.

ILI	**Dutch**	**Spanish**	**Italian**
{ *office* }	{ *kantoor; werkkamer; werkruimte* }	{ *oficina* }	{ *ufficio; studio* }
{ *role; part; office; function* }	{ *functie; rol* }, { *emplooi* }	{ *funciòn; papel; officio* }	{ *ufficio; mansione; carica* }

Table 1.1: Examples of some synsets linked to synsets in the ILI.

Verbs in EuroWordNet

EWN has some interesting additions to the set of verb relations. One of these is a relation INVOLVED which can be said to govern a set of relations specifying the type of involvement being described. It describes a relationship between two concepts where one concept is directly involved in some way with the other. Some subtypes of this relation are listed below:

INVOLVED_AGENT E.g. *undervise (teach)* INVOLVED_AGENT *lærer (teacher)*

INVOLVED_PATIENT E.g. *undervise (teach)* INVOLVED_PATIENT *student (student)*

INVOLVED_INSTRUMENT E.g. *male () paint* INVOLVED_INSTRUMENT *pensel (paintbrush)*

INVOLVED_LOCATION E.g. *undervise (teach)* INVOLVED_LOCATION *skole (school)*

INVOLVED_RESULT True if the RHS can be seen as something that is the result of the LHS . E.g. *fryse (freeze)* INVOLVED_RESULT *is (ice)*

INVOLVED_MANNER Is true if the RHS says something about the manner of which the LHS is performed. E.g. *skrike (scream)* INVOLVED_MANNER *høy (loud)*

2.3 DanNet

DanNet is a fully operational wordnet for the Danish language. The latest version, DanNet 2.1, is released under an open source licence and can be downloaded from `http://www.wordnet.dk/`. This version contains 62.000 synsets. Approximately 2000 of these synsets are mapped to equivalent synsets found in Princeton WordNet. Some additional relations are defined in addition to the relations found in Princeton WordNet and EuroWordNet, like CONCERNS, USED_FOR and MADE_BY. A set of descriptive features are also defined to enrich synsets and relations. Some examples are connotation (positive or negative), sex (male or female) and the domain for which a synset is considered to belong to (e.g. archaeology, electronics, geography etc.), along with possible links to equivalent base concepts in EuroWordNet. For relations, possible features are disjunction, negation, orthogonality and restriction[2].

According to the specifications for DanNet (Pedersen et al., 2011), about 30% of the material in DanNet is produced in a semi-automatic way. Around 2% of this material has been validated, indicating a high level of consistency in hyperonym relations but varying levels in other types of relations.

The source for the semi-automatic aquisition of semantic information was "Den Danske Ordbog" (DDO), a corpus-based dictionary describing the modern Danish vocabulary from about 1955 up until today. About half of the vocabulary in DDO is represented in DanNet. The subset is selected according to word frequency and with a preference for concrete objects over abstract concepts.

DDO was intended from the start to be a machine-readable resource. A lot of information for definitions was explicitly encoded with this in mind, contributing to the process of building a wordnet. This information included, among other things, subject or domain (e.g. *art* being the domain for *painting*), synonyms, near-synonyms and antonyms, collocational information and citations/example sentences. In addition, an explicit distinction between the genus proximum and differentia specifica of a definition is encoded.

This is a way to define a concept inspired by the teachings of Aristoteles. The genus proximum assigns an entry to a general class while the differentia specifica specifies properties

[2]for more information, consult the table in (Pedersen et al., 2011, p. 7-8)

that separates/differentiates it from other instances of the same class. E.g. to explain what a *triangle* is, one can say that it is a *geometrical shape* (genus) *with three sides* (differentia).

The genus proximum for each sense were directly transferred into the DanNet encoding tool and subsequently adjusted where needed. The adjustments took form of e.g. disambiguating the sense for a genus expression (not specifically encoded in DDO), or changing a general hyperonym to a more specific hyperonym (or a synonym).

2.4 NorNet

NorNet is a prototype for a wordnet for Norwegian Bokmål initially based on Nygaard's experiments in his thesis from 2006 (Nygaard, 2006). The results have subsequently been subject to post-processing and editing(Fjeld et al., 2012).

In 2010 further development of NorNet was initiated under the direction of Ruth Vatvedt Fjeld, professor of lexicography in the Department of Linguistics and Scandinavian Studies.

NorNet in its current state is limited to nouns with synonym and hyponym/hyperonym relations binding them together. Synsets are inferred based on the synonym relations.

3 Dictionary Concepts

Dictionaries generally have a well-defined structure that conforms to certain standards. This section explains concepts that are central for the discussion in the various parts of the method directly related to dictionaries. This is based on a survey made by Hausmann and Wiegand (Hausmann and Wiegand, 1989) as well as the introduction in the Norwegian monolingual dictionary *Nordisk Leksikografisk Ordbok* (Bergenholtz et al., 1997).

3.1 Overview of Dictionary Structure

Lemma Signs The items in a dictionary that act as the search keys. They have many of the same properties as the definition of a lemma in linguistics. The lemma signs are usually basic uninflected forms of a word, and are ordered in some way by the macrostructure and outer access structures.

Macrostructure The structure that maps all lemmas to their lexical items according to some ordering paradigm. The ordering paradigm is often based on alphabetical information, and/or some thematic or conceptual hierarchy.

Article A lemma, with all information regarding that lemma presented with it.

- **Outer Access Structure** One or more structures that specify how to make the reader find the information that is sought after. The outer access structure and the macrostructure tend to coincide if there is only one specified way to order the lexical items.
- **Inner Access Structure** The structure that specifies how to find information within a lexical item.

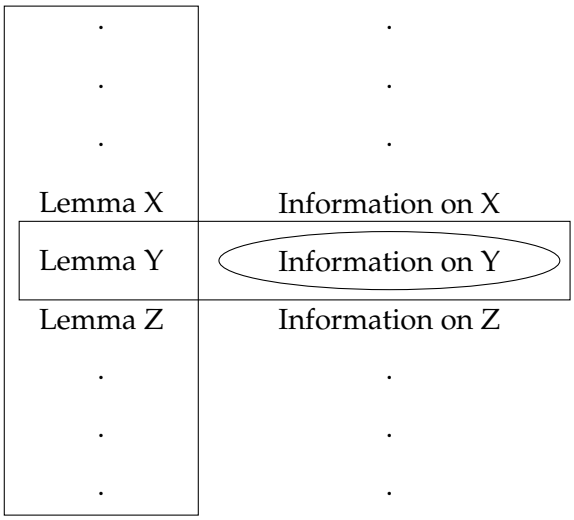


Figure 1.1: A simplified visual example of the macrostructure and microstructure of a dictionary, taken from (Hausmann and Wiegand, 1989, p. 329). The vertical box represents the macrostructure, the horizontal box represents an article and the ellipse represents the microstructure.

The Lexical Article

The lexical article is the most relevant part of the dictionary for this thesis. It is structured according to the notion of microstructures. A microstructure is a linear set of information types ordered according to to the inner access structure as specified by the dictionary. Some common information type classes are listed below, each with an explanation of the information one can expect to find within.

Synchronic Information Contains information about spelling, pronounciation and accentuation, part of speech, inflection and aspect.

Diachronic Information Etymological information.

Diasystematic Labelling Temporal labels, regional labels, borrowing labels, style labels, as well as technical field and group labels, attitude/connotation labels and usage labels.

Explanatory Information A short description of the item to be described, written in condensed natural language (the definition). May also contain linguistic or encyclopedic descriptions.

Syntagmatic Information Constructions, collocations, examples and quotations.

Paradigmatic Information Information about synonyms, antonyms, analogues, homonyms and paronyms.

Semantic Information Information types that point the reader to the specific sense of the lemma that is defined.

Usage Notes Snippets of texts examplifying the use of the lexical item in question.

Ordering devices represented as symbols such as numbers, letters and special characters often visualize the ordering and separation of the kinds of information types encountered in the

article, as well as acting as references (to information outside of the dictionary), cross-references (to other lexical items in the dictionary) and placeholders for the lemma (e.g. *h~* for *hunt*).

The core of the lexical article tends to be focused on the explanatory, syntagmatic and paradigmatic information, and in many dictionaries the microstructure is reduced even further, sometimes to the point where the only well-defined information category is the explanatory one (Hausmann and Wiegand, 1989, p. 342).

Chapter 2

Previous Work

This chapter presents some earlier and related work on the task of building a wordnet. Section 1 describes the two main approaches that are used - the merge approach and the expand approach. Sections 2 and 3 investigates two projects that makes use of a semi-automatic approach to extract semantic relations from a dictionary. The main focus in this chapter will be on the merge approach, as it is the approach taken by the method developed for this thesis.

1 Building a Wordnet

Various methods have been applied in the creation of wordnets. Princeton WordNet was created manually from scratch, much because of the fact that it was a pioneering project and as such could not rely on prior resources. This is probably the method that is the most demanding in terms of time and resources, but it does have some advantages: It reflects the properties of the target language right from the start and it allows one to sculpt the contents in any form and direction as one sees fit for the project in question. As a lot of ground has been covered in the last three decades this is normally not the approach used today. Aside from this approach, the methods are generally grouped into two classes (Vossen, 2002, p. 52):

Expand An already existing wordnet such as Princeton WordNet is used as the source material. The concepts are translated into the language for the new wordnet. Large parts of the semantic structure is inherited from the original network and the new wordnet is *expanded* with relations and concepts that differ from the original language.

Merge A wordnet is created based on local resources such as corpora or dictionaries. The resulting network is subsequently adapted to, or *merged* with, other wordnets (often Princeton WordNet) in order to ensure interoperability.

1.1 The Expand Approach

Creating a wordnet by the expand approach reduces the time and resources spent on creating a new semantic network as the existing relations and taxonomies are gained from the original. The disadvantage of this approach is that the new wordnet will be biased towards the original wordnet's representation of semantic knowledge. This may lead to problems such as:

- Missing concepts and relations occuring in the target language but lacking in the original language.

- Concepts and relations that are unnecessary or outright wrong as the result of concepts and relations that are only meaningful in the original language.
- Skewed ontologies and relational errors resulting from differences in the two languages regarding political, cultural and social structures.
- Differences in polysemous lemmas that mean the same in some contexts but not in all.

As a consequence, a substantial amount of post-processing and/or editorial work is usually required. FinnWordNet[1] has been created by translating Princeton WordNet 3.0. The translation was done manually by professional translators, increasing the labour during the creation of the wordnet but in turn decreasing the amount of post-editorial work needed.

Translating from a closely related language will probably reduce the problem area substantially. This is part of the motivation for a second wordnet for Norwegian Bokmål and Nynorsk which is under development by Lars Nygaard at Kaldera[2] for Språkbanken, a project held by The National Library of Norway[3]. This wordnet will be created semi-automatically by translating the resources in DanNet. Since the Danish language and Norwegian Bokmål is very closely related, the assumption is that the problems normally appearing in an expand approach will be minimized. The resulting wordnet for Norwegian Bokmål will then act as a source for an additional wordnet for Norwegian Nynorsk.

1.2 The Merge Approach

The merge approach tends to be based on dictionaries or corpora as resources. Wordnets created this way closely reflect the structure and quality of the source material. This section is mostly concerned with dictionary-based approaches since it is the most relevant one for this thesis.

Dictionary-based Approaches

The tasks of analyzing dictionary definitions and automatically extracting information from dictionaries have both been frequent subjects of study (Briscoe, 1989; Pedersen et al., 2009).

Research on extracting lexical and semantic information from conventional dictionaries have showed varying results. The conclusions were based on the observation of some properties of dictionaries that complicated the process, mainly:

- Inconsistent information within the dictionary.
- Mismatch between sense distinctions in a dictionary and sense distinctions in the natural language the dictionary describes.
- Implicit world/common-sense knowledge omitted in the dictionary.

A lot of the information needed for a wordnet is nonexistent in conventional dictionaries as commonsense knowledge is assumed to be known by the user. According to Pedersen et al. (2009, p. 272), definitions in monolingual dictionaries are usually phrased according to the substitution principle, which states that a lemma should be replaceable by its definition in

[1] http://www.ling.helsinki.fi/en/lt/research/finnwordnet/

[2] http://kaldera.no/

[3] http://www.nb.no/spraakbanken/

a given text. Because of this definitions tend to be short and incomplete, albeit usable in a sentence where the lemma occurs.

More encouraging results have been encountered from 2000 and onwards. DanNet was created initially by utilizing information using a Danish dictionary although about half of the total material has been created using other methods. Nygaard's experiments in his thesis (Nygaard, 2006) resulted in a large number of synonym and hyperonym relations between nouns and serves as the foundation for NorNet (see Section 2 for details).

A pilot study conducted during the creation of DanNet analyzed the definitions in Den Danske Ordbog (DDO) in an attempt to create more relations. The results from this pilot study was not used in DanNet however; translating and analyzing the semantic information encoded in the internal structure of the DDO database gave far better results (see Section 3 for details).

Corpus-based Approaches

The use of corpora as a source for extracting semantic data from natural language has been a popular approach when creating, enhancing and disambiguating semantic networks. Both rule-based and statistical methods have been used in this respect. The advantages of using corpora as a resource is that a huge amount of concepts and relations can be generated fairly quickly, resulting in a bigger semantic network than one could ever hope to create manually. On the other hand, this sometimes makes it harder to validate a large enough portion of it so to be certain of its consistency, unless a prior resource fit for this purpose can be used automatically. The need for a well-balanced corpus is apparent since the resulting network reflects the text it was extracted from, potentially biasing it towards the contexts found in the corpus. In any case, many interesting techniques has been developed. Some examples follow:

- Topic signatures (sets of topically related words) have been linked to WordNet synsets by using sense-tagged corpora and mining the web through queries built from concepts in WordNet (Agirre et al., 2001).
- BabelNet[4] treats Wikipedia as a kind of corpus by associating WordNet senses with Wikepedia pages utilizing hyperlink structure and information embedded in the pages in more or less natural language (Navigli and Ponzetto, 2010). Word similarity has been inferred using distributional methods (Pantel, 2005), which gives rise to e.g. automatic generation of synsets.
- A distributional method for automatically generating a thesaurus from text corpora was examined by Dekang Lin in 1998 (Lin, 1998). The source was a 64-million-word corpus containing text from newspapers. The resulting thesaurus was evaluated by comparing words with high frequencies to the equivalent words in WordNet1.5 and Roget's Thesaurus. The results indicated a strong agreement with WordNet1.5 synsets.

2 Automatic Extraction from Definitions in Bokmålsordboka

A semi-automatic extraction of hyperonym and synonym relations was explored by Nygaard in his cand. philol. thesis (Nygaard, 2006). Since his method serves as the inspiration for the method described in this thesis and shares the same goal (i.e. generating semantic relations for NorNet), it is examined and evaluated.

Nygaard's approach was to analyze the lexical entries in Bokmålsordboka (BOB)[5] in order

[4] `http://lcl.uniroma1.it/babelnet/`

[5] Both Bokmålsordboka and Nynorskordboka can be found at `http://www.nob-ordbok.uio.no/`

to create a set of rules for extracting semantic data useable for the generation of a wordnet. His method consisted of 3 steps (Nygaard, 2006, p. 45):

Preprocessing Handling and filtering of metainformation, alternative definitions indicated by parentheses and multiple definitions separated by a semicolon or comma.

PoS tagging Using the Oslo-Bergen tagger[6] for adding part-of-speech information to each definition entry.

Extraction Extraction of hyperonymy and synonymy relations based on POS tags and morphological features generated by the Oslo-Bergen tagger.

The ruleset consisted of heuristic lexicosyntactic rules of the forms shown in Table 2.1. The rules are taken from Nygaard's thesis (Nygaard, 2006, p. 51-53) and freely translated to English.

Rule	If a definition consist of a single noun, or several nouns separated by a comma, those nouns are synonyms for the definiendum
Example	**vidde** [...] område, areal
Result	*vidde (plateau)* HAS_SYNONYM *område (area)* *vidde (plateau)* HAS_SYNONYM *areal (area)*
Rule	If a definition contains other elements than single nouns, then the first noun in lemma form is a hyperonym of the definiendum, unless this noun is part of a list of stop words.
Example	**bistro** m1 (fr 'vertshus(holder)') liten resturant
Result	*bistro (bistro)* HAS_HYPERONYM *resturant (restaurant)*

Table 2.1: Examples of rules used by the algorithms developed by Nygaard. Bold words mark headwords; underlined words are the words that are selected by the corresponding rule.

To avoid erroneous extractions some filtering of the more problematic definitions were performed. A stop list of expressions was compiled and used to filter out definitions whose patterns failed to conform to the ruleset. These definitions were not considered by the algorithm. Some observations were made for erroneous decisions made by the morphological tagger. Based on these observations, definitions where the genus word was found at position 4 or higher in the sentences were excluded.

3 Automatic Extraction from Definitions in DDO

Another example of an attempt at developing an automatic extraction of semantic relations from a dictionary is an informal pilot study presented in Pedersen et al. (2009, p. 287-291), as part of the DanNet project. It presents a somewhat different approach to the Nygaard method, with its own set of challenges. The motivation for this study was to investigate whether a fully automated method could be developed for extracting semantic data from the definitions in DDO. DDO is structured in a way that strongly encourages forming definitions that conform to the principles of the genus proximum / differentia specifica. Specifically, the genus expressions are explicitly marked as such, giving a concrete distinction between that and the rest of the definition, which in turn should conform to the differentia as much as possible. Because of this, the assumption was that an automatic extraction of relations from the definitions would prove to be efficient and accurate.

[6] `http://tekstlab.uio.no/obt-ny/`

All definitions were transformed into a special type of corpus, each token tagged with the lemma form. A set of hypotheses serving as the foundation for lexicosyntactic rules were made after analyzing the definition structure in a fashion similar to the Nygaard approach (Pedersen et al., 2009, p. 288) as exemplified below:

- Adjectives preceeding the genus denote general (physical) properties of the definiendum.
- VPs in a relative clause which are headed by *kan* 'can' specify the function or use of the definiendum, i.e. the USED_FOR relation.

These rules were generalized into patterns that tried to capture as many definitions as possible and extract semantic relations from the definitions that matched the rule. Some examples are given in Table 2.2.

Rule	genus expression *til at* VP-inf *med/på/i*
Example	*apparat til at afspille cd'er med*
Rule	genus expression *der/som* VP-fin
Example	*apparat der måler og viser et køretøjs hastighed*
Rule	genus expression *til* NP
Example	*apparat til optagelse og afspilning av lyd*

Table 2.2: Examples of rules used in the DanNet pilot study

This method queries explicitly for lemmas and can thus target constituents of a definition in a very specific way, but it relies on the assumption that a given lemma form belongs to a certain type of grammatical class, and that its surrounding context is more or less unambiguous when it comes to the grammatical categories of the tokens. Each rule targets a small number of definitions, which gives a good precision rate. They are however generated by time consuming inspection of lists of definitions at the lemma level. This means that in order to cover a substantial amount of the dictionary, a substantial amount of rules must be made.

The overall conclusion made in this study was that an automatic extraction from a given dictionary will be successful only if the definitions conform to a vocabulary and syntactic structure that is formal and predictable to the extent that it would be deemed unacceptable by most lexicographers. No relations other than hyperonymy was extracted using this approach. Seeing as these relations were directly available from the explicitly marked genus expressions, the lemma-based analysis of the definitions does not prove to be adequate.

4 Concluding Remarks

The main difference between the DDO approach and Nygaard's method is the kind of data used for pattern matching. Where Nygaard used PoS tags as the foundation for discovering patterns, DDO's method made use of lemma forms of the tokens found in a definition. Nygaard's method does not consider many types of information stored in a word other than the grammatical class it belongs to, along with stop-words. The DDO approach is not aware of anything else than the explicitly marked genus expression and the lemma forms of the differentia specifica, and as such it has to make many assumptions that will only hold for a small set of instances.

The Nygaard method and the method applied to DDO both suffer from the fact that the source material gives incomplete information to the analysis tools. In the case of DDO, this

results in the lack of ability to generalize the ruleset, thus giving little advantage over manual approaches. The Nygaard method takes the opposite direction by creating a small set of rules that account for a large number of definitions. It is however unable to specialize and vulnerable to corner cases, and as such it is forced to ignore much of the data that would otherwise lead to valuable parts of the semantic network.

Part III

The Method: Dict2WN

Chapter 3

Method Overview

This chapter describes DICT2WN, a proposed method for semi-automatic extraction of semantic relations based on definitions in a dictionary. As stated in the introduction, the goal is to find a method that is as general as possible, so that it can be used for any dictionary, creating a wordnet or other semantic network efficiently and with an acceptable error rate. This method bases itself on of the merge approach, using a dictionary as the source material. It attempts to remedy the shortcomings of the two methods described in Sections 2 and 3 by looking at both lexical and morphosyntactic information, as well as employing a more flexible set of algorithms.

A detailed explanation for all the steps involved in the process is presented, all the way from the extraction from the dictionary to the final semantic network. For each step, the implementation of the method is presented as well, together with observations and preliminary results that are examined further in the evaluation chapter (Chapter 8).

1 Overview

Chapter 4 presents the extraction and preprocessing step, along with details on the application of these steps to BOB.

Chapter 9 presents the concepts of transducers, operator words and transducer expansion, the three main aspects of the proposed method.

Chapter 6 presents the types of graphs generated by the method and describes some techniques that might increase the quality of said graphs before the final export.

Chapter 7 gives an overview of the manual post-processing/annotation done on the data exported from DICT2WN as well as an annotation study.

Chapter 8 presents an evaluation of the process based on the previous chapters, particularly Chapter 7.

Chapter 9 makes some concluding remarks based on the observations made throughout the thesis along with propositions for further work.

Chapter 4

Extraction and Preprocessing

The first task of the method is to extract the required data from the source material and preprocess it so that it is ready for the next stage. Section 1 presents the extraction, while Section 2 explains the preprocessing task.

1 Extraction

The extraction process depends on the format of the dictionary. The most likely candidates for the method explored in this thesis are machine-readable dictionaries where one has access to the back-end (i.e. the electronic database). The possibility for extracting data from other formats (e.g. physical dictionaries or ones where access to the underlying architecture is limited) could however in theory be a viable option, since each definition part is distinguished from the others visually, with differing typefaces. Certain keywords and tokens may also act as separators (e.g. slash, tilde, 'jf.', etc.). This means that the output of a dictionary query will contain some kind of markup (e.g. HTML) that allows one to split the definition up into its constituents. Even if this should turn out not to be the case for a dictionary, or if the dictionary in question is a physical human-readable one, OCR techniques might be applied to identify the parts. This is outside of the scope for this thesis, but worth mentioning for the sake of emphasizing the goal of finding an approach that is as general as possible.

The data from the dictionary is extracted and converted into an XML file acting as the source data for the rest of the method. During this xml-file generation, each article and definition should be given a unique identifier. The preferred way of generatig these unique identificators is to transfer it directly from the dictionary, provided that the dictionary actually contains such information. If this is not the case it must be generated or inferred, in which case some interconnectivity between the dictionary and the resulting semantic network might be lost.

1.1 Extraction From BOB

BOB is stored in a relational database; its articles organized into different tables where each constituent of a given article is structured separately. Upon a query, these constituents are joined together and presented to the user in a readable format (see Table 4.1 for an example). Extracting the different parts of the definition was therefore a matter of extracting relevant table data, and thus fairly trivial.

In BOB's database, each article, definition and lemma have a unique identifier in the form of an integer. These were transferred as they were defined in BOB to ensure as high a degree of interconnectivity as possible between BOB and the resulting wordnet. Listing 4.1 gives some

Original		Translated	
slette	**I slette** *sletta* el. *-n* sludd	**sleet**	**I slette** *sletta* or *-n* sleet
slette	**II slette** f1 el. m1 (norr slétta) **1** større flat landstrekning *byen er omgitt av vide s-r* **2** flatt, avgrenset parti *en liten s- i skogen / skihopperen svingte på sletta*	**plain/ clearing**	**II slette** f1 or m1 (norr slétta) **1** larger flat area *the town is surrounded by wide p-s* **2** flat, bounded lot *a small c- in the forest / the ski jumper turned on the clearing*
slette	**III slette** v1 (norr *slétta*) **1** gjøre slett, jevne *s- duken / s- over* også: gjøre godt igjen */ s- til, ut* **2** fjerne, stryke *regnet s-t* (ut) *alle spor / s- et lydbåndopptak / gjelden ble s-t*	**smooth/ erase**	**III slette** v1 (norr *slétta*) **1** make smooth *s- the cloth / s- over* also: do well again */ s- to, out* **2** remove, erase *the rain e-* (out) *all traces / e- an audio recording / the debt was e-*

Table 4.1: Example of the result of a query for *slette* (Eng. *sleet* (noun), *plain / clearing* (noun), *smooth / erase* (verb))

examples of the extracted data. Line 1, 5, 9, shows examples of the article id's extracted from BOB. Line 2, 5, 10, 13 and 38 shows examples of extracted definition id's, while line 19 and 44 shows examples of extracted lemma id's.

2 Preprocessing

For each definition with a definiens, the definiens is tagged with an appropriate Part-of-Speech (PoS) tagger. This is a crucial step, and as will be shown, somewhat of a non-trivial step. Getting correct PoS-tag sequences for the explanatory parts of definitions is important, but a certain error rate must be expected.

Natural language is ambigious. Every PoS tagger constructed so far gives an error rate, and it is unlikely that a perfect PoS tagger will be constructed in the near future, as even humans find it hard to disambiguate properly in difficult cases. We must also assume that the error rate for a tagger designed for a given natural language will be higher when used for definitions in a dictionary than the error rate being reported from tagging natural texts. This is a consequence of the way the definiens is written in most dictionaries. The language used in dictionary entries tend to be not entirely formed as natural sentences, and this complicates the tagging step (see the definientia in Table 4.1 for some examples). Dictionaries are traditionally released in physical book formats and need to compress every definition as much as possible to save space. This is not a problem for human readers, since they can 'uncompress' the information by applying their knowledge about how dictionaries are written, and by using general common-sense knowledge to infer the missing parts. It is unrealistic to expect a tagger designed for natural languages to be able to do this - the contextual information needed to make correct morphological inferences may simply not be there.

An on-line dictionary such as the English Cobuild dictionary (`http://dictionary.reverso.net/english-cobuild`) is an example of a dictionary that most probably will not cause this problem. This type of dictionary gives complete, detailed definitions without making use of any type of text condending. As more and more dictionaries are designed with on-line user interfaces, one can expect text condensing to be less of a problem in the future, but as of now, such dictionaries are a minority.

There are however at least two ways to improve the tagging process. Given a set of rules specialized for condenced text and the keywords used in dictionaries one might avoid this problem. This is an interesting subject which in my opinion deserves of a thorough

Listing 4.1: A small exerpt of the XML file generated from BOB. The content of some of the definitions are removed and marked with "...". The definitions used as examples in the text are shown with full content.

```
<article art_id="54418">
  <definition def_id="66087">
    ...
  </definition>
  <definition def_id="66088">
    ...
  </definition>
</article>
<article art_id="54419">
  <definition def_id="1066335">
    ...
  </definition>
  <definition def_id="66089">
    <lookup>slette</lookup>
    <etymology>
      <etymology>norrsl'etta</etymology>
    </etymology>
    <lemmas>
      <lemma lemma_id="58163" pos="V01" form="slette"></lemma>
    </lemmas>
    <glosses>
      <gloss>
        <text>gjøre slett, jevne</text>
        <tagged>
          <word position="0" lexeme="gjøre" lemma="gjøre">verb inf tr1 rl9 pr3</word>
          <word position="1" lexeme="slett" lemma="slett">adj nøyt ub ent pos</word>
          <word position="2" lexeme="," lemma="$,">komma</word>
          <word position="3" lexeme="jevne" lemma="jevne">verb inf pa1 pa2 pa1/til
              pa2/til</word>
        </tagged>
      </gloss>
    </glosses>
    <examples>
      <example>'s– til, ut'</example>
      <example>'s– over' også: gjøre godt igjen</example>
      <example>'s– duken'</example>
    </examples>
  </definition>
  <definition def_id="66090">
    <lookup>slette</lookup>
    <etymology>
      <etymology>norrsl'etta</etymology>
    </etymology>
    <lemmas>
      <lemma lemma_id="58163" pos="V01" form="slette"></lemma>
    </lemmas>
    <glosses>
      <gloss>
        <text>fjerne, stryke</text>
        <tagged>
          <word position="0" lexeme="fjerne" lemma="fjerne">verb inf tr1 rl4 d5 rl9</word>
          <word position="1" lexeme="," lemma="$,">komma</word>
          <word position="2" lexeme="stryke" lemma="stryke">verb inf tr1 tr11 a11 rl14
              rl15</word>
        </tagged>
      </gloss>
    </glosses>
    <examples>
      <example>'s– et lydbåndopptak'</example>
      <example>'regnet s–t (ut) alle spor'</example>
      <example>'gjelden ble s–t'</example>
    </examples>
  </definition>
</article>
```

investigation, but it is unfortunately outside the scope of this thesis.

A simpler and somewhat less rigorous method is to add some extra information to the sentences before the tagging step, making incomplete sentences more syntactically coherent. This was done for the extracted data from BOB and is explained in detail in Section 2.1.

The XML data is subsequently preprocessed and inserted into an appropriate database format for further analysis and wordnet generation.

2.1 Preprocessing the Extracted Data from BOB

To aquire PoS tag sequences for the explanatory part of each extracted definition, OBT+Stat (Johannessen et al., 2011) was used. OBT+Stat is a morphological and syntactic tagger for Norwegian Bokmål and Nynorsk. It is based on the Constraint Grammar formalism, employing a CG3 ruleset developed at the Text Laboratory[1] at UiO, as well as making use of the HunPos Hidden Markov Model tagger (Halácsy et al., 2007) in order to remove any ambiguity left by the constraint-based tagger. OBT+Stat is available at `http://tekstlab.uio.no/obt-ny`.

Listing 4.2: An example of the output from OBT+Stat.

```
"<gjøre>"
        "gjøre" verb inf tr1 rl9 pr3
"<slett>"
        "slett" adj nøyt ub ent pos
"<,>"
        "\$," <komma>
"<jevne>"
        "jevne" verb inf pa1 pa2 pa1/til pa2/til
```

Each explanatory part was sent to OBT+Stat. An example of the output from OBT+Stat when given the sentence "gjøre slett, jevne" ("make smooth, smooth") for the explanatory part from one of the definitions for *slette (smooth)* can be seen in Listing 4.2. The output from OBT+Stat was processed and added to the extracted data, as can be seen on lines 25-28 and 50-52 in Listing 4.1.

DICT2WN currently makes use of the coarse PoS tags and lemma forms only, but all information from the tagger is included in the source material to ensure that the information will be available for any future improvement of the method. A list over the most relevant coarse PoS tags used by OBT+Stat can be seen in Table 4.2

Increasing Tagger Performance by Adding Context

OBT+Stat generally performs very well for Norwegian Bokmål (Johannessen et al., 2011, p. 29-31). Even so, due to the problem of the condensed form definientia it makes a number of erroneous decisions on the material extracted from BOB. By looking at the most frequent PoS pattern classes, a high degree of consistency in the types of errors made by the tagger was discovered. The most common tagging errors were located in the first token of some definitions where a verb was erroneously tagged as `SUBST` or `ADJ`, as well as in verb tokens occurring after commas. Special dictionary keywords like *overf* (eng. *fig.*) were mostly interpreted as nouns, cluttering the PoS patterns. In addition, incorrect PoS tag decisions sometimes propagated into the lemma inferences made by the tagger, giving incorrect lemma-forms for a number of tokens.

Based on the observed consistency in the erroneous decisions made by the tagger, an experiment was performed to see if the tagger performance could be increased. The infinitival

[1] `http://www.hf.uio.no/iln/om/organisasjon/tekstlab`

ADJ	adjective
ADV	adverb
DET	determiner
INF-MERKE	infinitival mark
INTERJ	interjection
INTERJ	conjunction
PREP	preposition
PRON	pronoun
SBU	subjunction
SUBST	noun
UKJ	unknown
VERB	verb
CLB	sentence boundary

Table 4.2: A list over the most relevant coarse PoS tags used by OBT+Stat. Short explanations in English for each tag is given in the second column

marker 'å' (equivalent to the English infinitival marker 'to') was prepended to every definiens for a definiendum that was assigned a grammatical code equivalent to a verb (see Table 4.4). The infinitival marker was removed from the tagged PoS sequence before inserting it into the XML-file by specifying an offset equivalent to the number of tokens added (in this case, one).

Two versions of the XML-file were generated, one that utilized the prepended infinitival marker and one that didn't. By examining the differing PoS tag sequences, the conclusion was that the experiment did in fact cause a substantial change in the decisions made by the tagger. 5060 PoS tag sequences for verb definitions were tagged differently when prepending the infinitival marker. Of these, 3593 changes were for the coarse PoS tags.

In order to evaluate the changes, 100 PoS tag sequences were investigated, selected at random from the set of the 3593 changes in coarse PoS tag sequences. The results were very encouraging. A total of 89 sequences went from partially incorrect to completely correct, 7 sequences were improved but still contained errors, while 4 were still incorrect. None were given a sequence that could be considered to be more incorrect than the original.

The most significant changes were from tokens incorrectly tagged as nouns or adjectives to tokens correctly tagged as verbs. This happened most often in the beginning of the definitions, but the added context also influenced tokens appearing e.g. after commas further out in the sentence. Incorrectly inferred lemma forms of the tokens were also changed into correct lemma forms in many cases upon being assigned the correct PoS tag. This even happened in cases where the PoS tag was correctly tagged but incorrectly lemmatized.

We interpret the amount of sequences that were correctly tagged when the infinitival marker was prepended as the proportion of success in a bernoulli trial process. By this interpretation, we can use a confidence interval equation for binomial proportions to estimate how much of the whole data set will be tagged correctly with added context:

$$\hat{p} \pm z_{1-\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{4.1}$$

where $\hat{p}$ is the success ratio of the bernoulli trial process, $z_{1-\alpha/2}$ gives the percentile rank for our confidence level and n is the sample population size.

With a success rate of 0.89, a population size of 100, using a confidence level at 95%, we

calculate

$$0.89 \pm 1.96\sqrt{\frac{0.89(1-0.89)}{100}} \tag{4.2}$$

which gives us the confidence interval 0.89 ± 0.0613. We can thus assume with 95% confidence that between 82.87% and 95.13% of incorrect PoS tag sequences will be tagged correctly with added context. This contributes substantially to the consistency of the source data. Table 4.3 gives some examples of erroneously tagged definitions and the improvement gained from prepending the infinitival marker.

def.	late	som	,	hykle	
trans.	*pretend*	*to*	,	*act hypocritical*	
lemmas	~~lat~~ late	som	$,	hykle	
PoS	~~adj~~ VERB	PREP	KOMMA	VERB	
def.	innebære	,	by	på	
trans.	*involve*	,	*present*		
lemmas	innebære	$,	by	på	
PoS	VERB	KOMMA	~~subst~~ VERB	PREP	
def.	lage	,	forme		
trans.	*create*	,	*shape (* VERB *)*		
lemmas	lage	$,	forme		
PoS(err)	~~subst~~ VERB	KOMMA	VERB		
def.	blande	i hop	,	røre	sammen
trans.	*mix*	*together*	,	*stir*	*together*
lemmas	blande	i_hop	$,	røre	sammen
PoS	~~subst~~ VERB	ADV	KOMMA	~~subst~~ VERB	ADV

Table 4.3: Some examples of erroneously tagged definitions that were correctly tagged when prepending an infinitival marker. An English translation of the definition is given in the *trans.* rows. Erroneous decisions by the tagger are represented by crossed-out lemmas or PoS tags while the correct decisions proceed it.

Based on the results of the tagger experiment, the data containing the improved tagger decisions was decided upon as the best data source for the rest of the method. This data was subsequently inserted into DICT2WN's own database.

Conversion of Grammatical Codes

After inserting all the information from the XML file from the previous step into Dict2WN's database, a set of regular expressions were defined, converting the grammatical codes found for definitions in the dictionary to equivalent PoS tags used by the Oslo-Bergen tagger (see Table 4.4 for the regular expressions and corresponding PoS tags). This was done to simplify some of the later inference steps.

Not every definition within an article is assigned a lemma with a grammatical code, but since every article contains definitions for the various senses of a wordform for a particular grammatical class, one can safely infer the appropriate coarse-grained PoS tag for all the definitions the article contains - given that one or more of them are assigned a lemma with a grammatical code.

A statistical summary of all the data extracted from BOB is presented in Table 4.5. We can see that of all the extracted articles from BOB, verbs articles constitute about 10.6%. A total of 26235 definitions belong to these articles; out of these, 11414 have an explanatory part. These 11414 definitions are our main targets for extraction of semantic relations.

Regex	PoS
$[vV][S\|\backslash d]+$	verb
$[FMmN]\backslash d+$	subst
$[aA].+$	adj
$P[p\|P]$	prep
I	interj
$symb$	symb
T	tall
pn	pron
vr	vr
X	prefix
$fork$	fork
^$	none

Table 4.4: Conversion from grammatical codes to equivalent PoS tags compatible with the Oslo-Bergen tagger.

	Total	**Verbs**	**Expl**
Article ID's	66086	7005	6904
Definition ID's	219169	26235	11414
Lemmas	60949	6689	6600

Table 4.5: Statistics for the extracted data from BOB regarding **Article ID's**, **Definition ID's** and **Lemmas**. The column **Total** lists the total amount of unique identifiers extracted from the dictionary, **Verbs** restricts the amount to the number of unique identifiers describing some verb (**Verbs**), while **Expl** further restricts the amount to the ones actually having some explanatory information attached.

Chapter 5

Transducer Generation

The preprocessing step supplies us with sequences of PoS tags and lemma forms for each definiens we have imported into DICT2WN. In this section we start making use of this information. We begin by introducing the term *PoS pattern class* in Section 1. We then move on to a general description and our application of finite state transducers in Section 2, as they are one of the main components of DICT2WN. Section 4 describes the observations and assumptions leading to the manual creation of the initial transducer. Section 5 describes an augmentation of the transducer output that increases the flexibility of the method and a description of the second manually created transducer. Section 6 describes how the manually created transducers are automatically expanded in order to catch more of the source material. **??** describes the rationale for the third and final manually created transducer. Finally, gives an overview of all transducers along with some statistics.

1 PoS Pattern Classes

The PoS tag sequences for our definientia are used to create a set of *PoS pattern classes*. A PoS pattern class is defined by a unique sequence of PoS tags. A *member* of a PoS pattern class is a definition whose definiens has a PoS tag sequence identical to that of the PoS pattern class. We define the *size* of a PoS pattern class in terms of the number of members. A frequency list for the largest PoS pattern classes is presented in Table 5.1.

A total of 4007 PoS pattern classes from verb definientia were generated. We can see from the frequency list in Table 5.1 that the largest PoS pattern class contains approximately 7.1% of the extracted definentia, while the second largest contains about 5.7%. After the first few classes the size of PoS pattern classes drops quickly. There is a long tail of very small PoS pattern classes, 3395 of which have a size of one.

Much of the initial phase of the development of DICT2WN was spent on inspecting the members of the largest PoS pattern classes. The observations done during this work backed up the assumption that similar definientia exhibit similar behaviour to the extent that it was deemed appropriate to continue the development of the method. This could be seen as an initial step to determine if the dictionary in question is consistent enough for this method. BOB is created for human readers and is not targeted towards machine-readability to the extent that e.g. DDO is. Thus, if an adequate set of relations can be created based on the data extracted from BOB, a similar treatment of dictionaries with an equal or bigger focus on machine-readability should be considered useful.

PoS Pattern						Size	Perc	Transducer Match
VERB	KOMMA	VERB				809	7.0878	1, 3.1
VERB						650	5.6948	1, 2, 2.1, 2.1.1, 2.2, 2.2.1, 2.3, 2.3.1
VERB	PREP	SUBST				295	2.5845	2, 2.1, 2.1.1, 2.2.1, 2.3, 2.3.1
VERB	SUBST					258	2.2604	2, 2.1, 2.1.1, 2.2, 2.2.1, 2.3, 2.3.1
SUBST	PREP	UKJ	UKJ	CLB		244	2.1377	
ADJ	CLB					241	2.1114	
VERB	ADJ					232	2.0326	2, 2.1, 2.1.1, 2.2, 2.2.1, 2.3, 2.3.1
SUBST	PREP	PREP	UKJ	CLB		221	1.9362	
UKJ	CLB					206	1.8048	
VERB	PREP					129	1.1302	2, 2.1, 2.1.1, 2.2, 2.2.1, 2.3, 2.3.1
VERB	KOMMA	VERB	KOMMA	VERB		125	1.0952	1
VERB	ADJ	KOMMA	VERB			112	0.9813	
VERB	SUBST	PREP				94	0.8236	2, 2.1, 2.1.1, 2.2, 2.2.1, 2.3, 2.3.1
VERB	KOMMA	VERB	PREP			76	0.6658	3.1
VERB	KOMMA	VERB	PREP	SUBST		75	0.6571	3.1
FORK						73	0.6396	
VERB	PREP	KOMMA	VERB			71	0.6220	
VERB	KOMMA	VERB	ADJ			70	0.6133	3, 3.1
VERB	SUBST	KOMMA	VERB			63	0.5520	
VERB	PREP	SUBST	KOMMA	VERB		49	0.4293	
VERB	SUBST	PREP	KOMMA	VERB		46	0.4030	
VERB	ADJ	INTERJ	ADJ			45	0.3943	
VERB	SUBST	PREP	SUBST			43	0.3767	2.1.1, 2.2.1, 2.3.1
PREP	SUBST					43	0.3767	
VERB	KOMMA	VERB	SUBST			42	0.3680	2.1.1, 3.1
PREP	UKJ					41	0.3592	
VERB	PREP	PREP	SUBST			36	0.3154	2.1.1, 2.2.1, 2.3.1
PREP	SUBST	CLB				34	0.2979	
SUBST	CLB					34	0.2979	
VERB	ADV					33	0.2891	2, 2.1, 2.2, 2.3 2.1.1, 2.2.1, 2.3.1
VERB	PREP	ADJ	SUBST			31	0.2716	2.1.1, 2.2.1, 2.3.1
VERB	VERB					30	0.2628	2, 2.1, 2.1.1, 2.2.1, 2.3.1
VERB	KOMMA	VERB	SUBST	PREP		29	0.2541	3.1
PREP	PREP	UKJ	CLB			26	0.2278	
VERB	ADJ	KOMMA	ADJ			26	0.2278	
VERB	SUBST	PARST	DET	PAREN		25	0.2190	
VERB	ADV	KOMMA	VERB			24	0.2103	
VERB	PREP	KOMMA	VERB	PREP		24	0.2103	
VERB	PREP	SUBST	PARST	DET	PAREN	23	0.2015	
VERB	SUBST	PAREN				22	0.1928	2, 2.1.1, 2.2, 2.2.1, 2.3.1

Table 5.1: Frequency list for the largest PoS pattern classes. The columns show the actual PoS pattern (**PoS Pattern**), the membership frequency (**Size**), the percentage of all verb definienda that are members of the class (**Perc**), and the subset of the transducers that matched the class (**Transducer Match**). Non-explanatory PoS patterns (see Section 1.1) are marked using gray text.

1.1 Non-explanatory PoS Pattern Classes

Upon inspection of the largest PoS pattern classes, some turn out to show different properties from the others in terms of how their definientia are formed. They are not condensed sentences of natural language; instead they belong to a class that contain sequences of tokens that describe their definiendum either as being a morphosyntactic variation of another definiendum, referring to the syntagmatic part of the definition; or definientia that consistently refer to the domain that the definiendum belongs to.

We refer to such PoS pattern classes as *non-explanatory PoS pattern classes*, and their members as *non-explanatory definientia*. Table 6.1 gives some examples of non-explanatory definientia. The reader can assume that unless explicitly referred to as non-explanatory, PoS pattern classes and definientia will refer to the set of explanatory PoS pattern classes and explanatory definientia.

SUBST SUBST	PREP PREP	UKJ PREP	UKJ UKJ	CLB CLB
adj	i	pf	pt	:
subst	i	pf	pt	:
adj	i	pr	pt	:
ADJ UKJ	CLB CLB			
refl	:			
mat.	:			
bot.	:			
overf	:			
idr	:			

Table 5.2: Examples of non-explanatory PoS pattern classes. The first non-explanatory PoS pattern classes show examples where the definienda refer to the syntagmatic parts of the definition. The third and fourth contain special keywords used in dictionaries to denote domain, metaphor, etc.

These classes should in theory provide access to information about e.g. domain and relationships between verbs and adverbs. However, the strategy for extracting this information should be different from that of explanatory definentia. Since this study is focused on the task of processing condensed text from explanatory definientia, they are not being treated in any way in this study.

2 Finite-state Transducers

A finite-state transducer is a type of finite automaton that can, in addition to recognizing some set of sequences of strings or symbols, compute relations between two sets of strings or symbols and generate a corresponding output sequence based on these relations and the input sequence(Jurafsky and Martin, 2008, p. 91-94). Since we want to find relation between PoS tag sequences and relation symbols, this should be an appropriate choice. To simplify some later steps in the method, *sequential finite-state transducers* as defined in Mohri (1997) are used. This type of transducer is deterministic on its input and linear in terms of time complexity. Although the output of such transducers are not guaranteed to be unambiguous, no ambiguity was observed when developing and testing transducers for this method. It is therefore considered adequate for this particular study.

Definition: Let P be the finite set of all PoS tags generated by the tagger, and R the finite set of relation symbols.
Our transducer for DICT2WN is a sequential symbol-to-symbol transducer where:

- Q is a finite set of N states $\{q_0, q_1, \ldots, q_n\}$,
- $i \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states,
- $P' \subseteq P$ is the input alphabet,
- $R' \subseteq R$ is the output alphabet,
- $\delta(q, p)$ is the state transition function $Q \times P' \rightarrow Q$, and
- $\sigma(q, p)$ is the output function $Q \times P' \rightarrow R'^{*}$

From now on, term *transducer* is restricted to this definition unless explicitly stated otherwise. Using our transducer definition, we can analyze PoS pattern classes and assign relations between a definiendum and tokens in its corresponding definiens by creating transducers that accept some set of PoS pattern classes. We specify both accepted input sequences and resulting output sequences using regular expressions (regexes). The notation used for representing this way of specifying a transducer is introduced with the following example:

I: VERB (KOMMA VERB)*
O: SYN (NIL SYN)*

The first row is the specification of input sequences accepted by the transducer, while the second row shows the resulting output. This is one of the actual transducers that will be described further in Section 4.1. For now, suffice to say that the transducer will accept one VERB symbol followed by $k \geq 0$ instances of a KOMMA VERB sequence; giving as its output a corresponding sequence of one SYN symbol followed by sequences of NIL SYN symbols of length equal to k. From the PoS tagging step we have the lemma sequence of the definiens for any member of an accepted PoS pattern class. This gives us three aligned sequences for a definiens accepted by some transducer which will be referred to as a *triple sequence*. The notation for a triple sequence resulting from an accepting transducer and the corresponding PoS tag sequence and lemma sequence is as follows:

L:	l_1	l_2	...	l_n
P:	p_1	p_2	...	p_n
R:	r_1	r_2	...	r_n

where n is the token length of the definiens in question, $l_1\ l_2 \ldots l_n$ are members of the set of lemmas generated by the tagger, $p_1\ p_2 \ldots p_n$ are members of the set of PoS tags in use, and $r_1\ r_2 \ldots r_n$ are members of our set of relation symbols. The combination of a lemma, PoS tag and relation symbol at any index in the triple sequence will be referred to as a *triple*. We refer to tokens in the lemma sequence using the term *target words*, implying that they are potential targets of a relation assignment from their corresponding definiendum to themselves.

3 1-to-n Target Ambiguity

Before moving on the task of creating transducers, we define the term *sense* as the specific meaning of a lemma assigned explicitly by a definition ID as extracted from BOB. For each relation created from a definiendum to a target word, the sense for the definiendum is unambigious since it is attached to one and only definition ID in the preprocessing step.

This is not the case for target words in the definiens since the only information available from the original data is the PoS tagging done in the preprocessing step. There is no information in the extracted data from the dictionary that point to the correct sense for any target word of any definiens.

We thus end up with lists of possible definition IDs for each target word in the definiens, as can be seen in Figure 5.1. This type of sense-ambiguity is defined in this thesis as *1-to-n target ambiguity* (where 1 stands for the unambiguous definiendum sense and *n* stands for the *n* possible senses for a target word in the definiens).

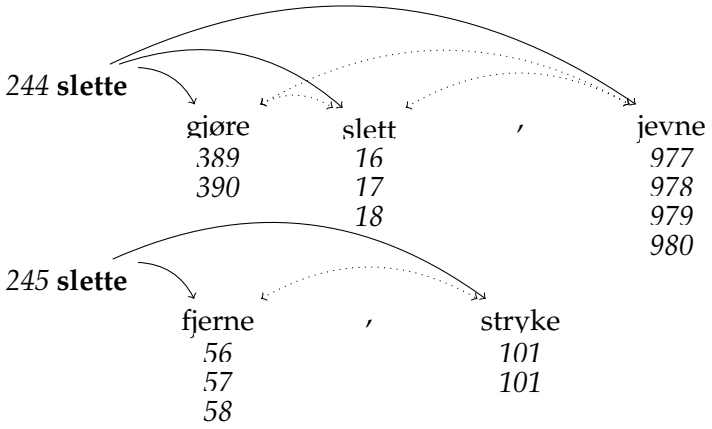


Figure 5.1: Graphical representation of the definiendum-definiens relation. The definienda are in boldface, with edges to possible target words. Definition IDs for both definienda and target words are emphasized (the actual id's are simplified for the sake of readability and do not correspond to the actual IDs used). This examplifies the 1-to-n target ambiguity - the sense for the definiendum is disambiguated by the definition id, while the possible senses for each target word is the set of possible definition id's shown as a list beneath each token in the definentia.

4 Manual Transducer Generation

After excluding the non-explanatory PoS pattern classes, we start looking for the simplest PoS pattern classes with the highest number of members. By ordering the list of PoS pattern classes in descending order based on their number of members, the most common patterns are discovered. These are short, simple sequences of PoS tags that act as the foundation for the initial round of transducer generation.

At this point it is important to identify the basic formal guidelines that the dictionary follows. If no such thing is explicitly defined for the dictionary, one can assume a general genus proximum / differentia specifica guideline, along with basic comma separated definientia describing their definiendum in terms of synonyms. Adhering to these guidelines, regular expressions for transducer inputs (accepted PoS patterns) and outputs (relations) are created.

4.1 Initial Transducer Generation for BOB

One of the rules in Nygaard's thesis performs the task of assigning synonym relations to all nouns encountered in single-noun and comma-separated single-noun definientia (Nygaard, 2006, p. 51-53). Upon manual inspection of similar PoS pattern classes for verbs, it turns out that one can make the same general assumption when similar patterns are encountered.

We thus create a transducer that captures single-verb and comma-separated single-verb PoS pattern classes. This transducer accepts four distinct PoS pattern classes, assigning HAS_SYNONYM relations between a definiendum and all verbs found in its definiens. We specify this by applying the regex ' VERB (KOMMA VERB)*' for the transducer input, and the regex ' SYN (NIL ssyn)*' for the transducer output. The NIL symbol means that no attempt of relation assignment will be done at that position. The SYN symbol specifies that a HAS_SYNONYM relation should be assigned between the definiendum and the lemma found at that position in the lemma sequence. As an example, consider the definition:

forbause:	forbløffe, forundre, overraske
astonish:	baffle, puzzle, suprise

The specified transducer will generate a triple sequence of the form:

L: forbause	,	forundre	,	overraske
P: VERB	KOMMA	VERB	KOMMA	VERB
R: SYN	NIL	SYN	NIL	SYN

We subsequently iterate through each triple in this sequence and generate relations from the sense id found for the definiendum and the sense id(s) found for the lemmas at each position - unless the relation symbol is NIL . The results for transducer 1 can be seen in Figure 5.2.

Transducer 1: Assignment of HAS_SYNONYM relations for all definitions consisting of single verbs and comma-separated verbs

Regex:

I: VERB (KOMMA VERB)*
O: SYN (NIL SYN)*

Statistics

Hits		1-to-n Ambiguity	
PoS	4	**Min**	3
Def	1866	**Max**	32
Lem	2852	**Mean**	6.8138
Sen	9642	**StdDev**	4.8238

Rel	Freq
HAS_SYNONYM	20864

Figure 5.2: General data for transducer 1

As we can see, we generate a fairly high number of HAS_SYNONYM relations in this first step since some the captured PoS pattern classes are among the largest ones. These relations will act as the foundation for generating synsets, which is explained in Chapter 6.

There are 1866 members in total in these four PoS pattern classes. The total amount of distinct lemmas found in both definienda and definentia are 2852. These lemmas are tied to 9642 distinct senses. Due to the 1-to-n target ambiguity, the total number of generated

HAS_SYNONYM relations must be taken with a grain of salt. As we can see from the statistics for the 1-to-n target ambiguity, the average ambiguity is at 6.8 possible senses for a target word. The maximum number of possible senses for a lemma is at 32, and we can see that no lemma is completely unambiguous. If we assume that one relation is correct for each target word in the definiens, we can assume that the minimal amount of unambiguous relations is at least proportional to the sum of all target words for the captured definientia.

PoS Pattern Class Matches							
	VERB	KOMMA	VERB	KOMMA	VERB	KOMMA	VERB
more	muntre	,	oppmuntre	,	underholde	,	glede
'amuse'	'cheer'	,	'encourage'	,	'entertain'	,	'delight'
	VERB	KOMMA	VERB	KOMMA	VERB		
kompensere	utjevne	,	godtgjøre	,	erstatte		
'compensate'	'equalize'	,	'indemnify'	,	'compensate'		
	VERB	KOMMA	VERB				
støte	fornærme	,	krenke				
'offend'	'insult'	,	'violate'				
	VERB						
respondere	svare						
'respond'	'answer'						

Table 5.3: Some examples of members of PoS pattern classes captured by transducer 1. Definienda are shown in the first column.English translations for every example are shown in quotes.

5 Operator Word Generation

During the initial phase of the development of DICT2WN, a lot of time was spent compiling lists of definitions, looking for patterns and indications of some special behaviour in the definientia. One of the things that became evindent was that tokens in a definiens could be designated to three different classes of words, each with differing properties and behaviour. These three classes of constituents for definientia are defined below and used throughout the rest of this thesis. The term *target word* was introduced in Section 2 and is now restricted further.

Target Word A token in some definiens that contains semantic information about the definiendum it belongs to, represented by its lemma. The set of lemmas belonging to this class, as well as the set of lemma signs found in definienda, are the sets of words that we want to generate semantic relations between.

Operator Word A token in the definiens that contains little or no semantic information about its definiendum, but that instead tends to change the meaning of one or more tokens in its surrounding context. An operator word is represented by its word-form just like target words.

Structural Token A token that provides additional structure to the definiens (e.g. commas, semicolons, parentheses etc.), but that neither adds semantic information to the definiendum nor changes the meaning of any surrounding tokens in any significant way.

As an example, consider the definition "**intensivere**: forsterke, gjøre mer effektiv" (en. "**reinforce**: make more effective"). The tokens *forsterke* and *effektiv* are definitely target words

according to our definition, while *gjøre* is more of a candidate for the class of operator words. The comma belongs to the class of structural tokens. The last token, *mer*, is a trickier case which, as we shall see later, is a general case for a lot of adjectives and adverbs.

It is clear that *gjøre* affects the semantic value of *effektiv* by specializing it, implicating that the definiendum has a causal relationship to something, effective being part of the outcome in some way. If we remove *gjøre*, we no longer know if effective is e.g. an inherent property of that something, or the effect of some cause.

The set of words belonging to these three classes are not neccesarily mutually exclusive. Very general verbs can sometimes act as both target words and operator words. The class of operator words generally contain prepositions, verbs that appear near the top of a hyperonym/troponym hierarchy and what is commonly referred to as light verbs (Butt, 2003).

It seems that we cannot always know the true semantic value of every token in some definiens. We must therefore assume that there might be tokens elsewhere that must be taken into consideration. To account for this in DICT2WN, the notion of operator words is formalized and an algorithm created based on the formalization.

5.1 Operator Word Definition and Example

> **Definition:** Let L be the finite set of all lemmas in use, P the finite set of all PoS tags in use, R the finite set of all relation types in use, and T a finite set $T = \{self, left, right\}$.
> An operator word is a function $\lambda : T \times P \rightarrow R$.
> For some set O of operator words $O = \{o_1, o_2, \ldots, o_n\}$, we define a function $\Lambda : L \rightarrow O$ that maps a lemma to a certain operator word. We refer to a specific operator word function returned by Λ as $o.\lambda$.

Based on this definition, we define an algorithm that makes use of operator words in order to transform the output from some transducer. This algorithm is described in pseudocode in Algorithm 1.

The algorithm takes the output from a transducer and searches for an OPER symbol in the relation sequence, starting at the end of the relation sequence and proceeding towards the operator word. If such a symbol is found, the Λ function is called with the lemma in the same position as the OPER symbol as the argument (line 2-4). If Λ returns an operator word, the algorithm proceeds by searching for an ARG symbol to the right of the operator word; then to the left of the operator word. In the case that an ARG symbol is found, the operator word function $o.\lambda$ is called and the relation symbol at the argument position is replaced with the symbol returned from $o.\lambda$. If there is no defined mapping from the PoS tag/lemma pair found at the argument position, the ARG relation is kept as it is. Finally, $o.\lambda$ is called on the operator word itself according to the same criteria.

Example

We start by defininining a very limited set of two operator words. We define an operator word function $o_1.\lambda$:

$$0_1.\lambda(t,p) = \begin{cases} \text{NIL} & \text{if} \quad (t,p) = (\text{self}, \texttt{VERB}) \\ \text{CAUSES} & \text{if} \quad (t,p) = (\text{right}, \texttt{ADJ}) \\ \text{INVOLVED} & \text{if} \quad (t,p) = (\text{right}, \texttt{SUBST}) \end{cases} \tag{5.1}$$

We then define another operator word function $o_2.\lambda$:

$$0_2.\lambda(t,p) = \begin{cases} \text{NIL} & \text{if} \quad (t,p) = (\text{self}, \texttt{PREP}) \\ \text{INVOLVED} & \text{if} \quad (t,p) = (\text{left}, \texttt{SUBST}) \\ \text{HAS\_HYPERONYM} & \text{if} \quad (t,p) = (\text{left}, \texttt{VERB}) \\ \text{ENTAILS} & \text{if} \quad (t,p) = (\text{right}, \texttt{ADJ}) \\ \text{CAUSES} & \text{if} \quad (t,p) = (\text{right}, \texttt{SUBST}) \end{cases} \tag{5.2}$$

Finally, we define the function Λ:

$$\Lambda(l) = \begin{cases} o_1 & \text{if} \quad l = \textit{gjøre} \\ o_2 & \text{if} \quad l = \textit{til} \end{cases} \tag{5.3}$$

Consider the following definition:

likerette: gjøre vekselstrøm om til likestrøm
rectify: make alternating current into direct current

We assume that the definiens has been tagged with the PoS tag sequence `VERB SUBST PREP PREP SUBST` , which is matched by a transducer of the form:

I:	VERB	SUBST	PREP	PREP	SUBST
O:	OPER	ARG	OPER	OPER	ARG

Our resulting triple sequence from the definiens thus becomes:

L: gjøre	vekselstrøm	om	til	likestrøm
P: VERB	SUBST	PREP	PREP	SUBST
R: OPER	ARG	OPER	OPER	ARG

We now have what we need to run Algorithm 1 on our definiens, transforming the relation sequence in our triple according to our set of operator words. This is shown in Table 5.4.

5.2 Candidate Operator Words

Candidate operator words are found by making a frequency list over all lemma forms of the words found for verb definitions. The most frequent lemmas tend to either belong to a class of very general concepts close to the top of a hyperonym hierarchy (e.g. *person*, *country*, *do*, *become*, etc.), or the class of words that should be treated as operator words. There is a fuzzy limit between some lemmas that could belong to both classes. The best way to handle these is to give them a status of an operator word.

One important question materializing from this step is whether the operator words and their arguments can be found with similar patterns in natural language. If this is the case, this step can be generalized further by employing linguistic knowledge instead of manually searching for patterns in a slightly ad-hoc way.

Lacking some concrete evidence to suggest a significant correlation between general linguistic theory and my definition of operator words, my choice of strategy fell upon simply compiling lists of the most frequent contexts for the most frequent lemmas in my data. The context for some lemma was defined as the n PoS tags to the left and right of the lemma, i.e. a PoS tag window of size n.

Algorithm 1 Relation Sequence Transformation

Require: lem ▷ A sequence of lemma strings
Require: pos ▷ A sequence of PoS tag symbols
Require: rel ▷ A sequence of relation symbols

function TRANSFORM(lem, pos, rel)
  for $i \leftarrow |l|, 0$ **do**
    if $rel[i] =$ OPER **then**
      $o \leftarrow \Lambda(lem[i])$
      if o **then**
        $iRight \leftarrow$ FINDARGPOS$(i+1, |l|)$
        $iLeft \leftarrow$ FINDARGPOS$(i-1, 0)$
        if $iRight$ **then**
          UPDATEREL$(iRight, o, right)$
        end if
        if $iLeft$ **then**
          UPDATEREL$(iLeft, o, left)$
        end if
        UPDATEREL$(i, o, self)$
      end if
    end if
  end for
end function
function UPDATEREL(i, o, t)
  $rel[i] \leftarrow o.\lambda(t, pos[i])$
end function
function FINDARGPOS($start, end$)
  for $i \leftarrow start, end$ **do**
    if $rel[i] =$ ARG **then return** i
    end if
  end for
end function

gjøre	*vekselstrøm*	*om*	*til*	*likestrøm*	Initial input lemma sequence
VERB	SUBST	PREP	PREP	SUBST	Initial input PoS tag sequence
OPER	ARG	OPER	OPER	ARG	Initial input relation sequence
OPER	ARG	OPER	**OPER**	ARG	Search for OPER symbol
gjøre	*vekselstrøm*	*om*	o_2	*likestrøm*	Apply $\Lambda(til) = o_2$
OPER	ARG	OPER	o_2	**ARG**	Search for ARG symbol to the right
VERB	SUBST	PREP	o_2	**CAU**	Apply $o_2.\lambda(right, \mathsf{SUBST}) =$ CAU
OPER	**ARG**	OPER	o_2	CAU	Search for ARG symbol to the left
VERB	**INV**	PREP	o_2	CAU	Apply $o_2.\lambda(left, \mathsf{SUBST}) =$ INV
VERB	INV	PREP	**NIL**	CAU	Apply $o_2.\lambda(self, \mathsf{PREP}) =$ NIL
OPER	ARG	**OPER**	NIL	CAU	Search for OPER symbol
gjøre	*vekselstrøm*	**soper**	*til*	*likestrøm*	Apply $\Lambda(om) = undefined$
OPER	INV	OPER	NIL	CAU	Search for OPER symbol
OPER	INV	OPER	NIL	CAU	Apply $\Lambda(til) = o_1$.
o_1	INV	OPER	NIL	CAU	Search for ARG symbol to the right
o_1	INV	OPER	NIL	CAU	Search for ARG symbol to the left
NIL	INV	OPER	NIL	CAU	Apply $o_1.\lambda(self, \mathsf{VERB}) =$ NIL .
NIL	INV	OPER	NIL	CAU	Final output relation sequence

Table 5.4: Example of relation sequence transformation using operator words.

5.3 Second Transducer Generation for BOB

After defining a set of operator words, a second transducer is manually created, targeting simple two-word definientia consisting of a verb followed by either a noun, adjective or preposition. The output is now assigned assigned operator words and arguments, instead of explicit relations. The general data for transducer 1 can be seen in Figure 5.3.

Transducer 2: Assignment of operator words and arguments for definitions consisting of a verb, followed by either a noun, adjective or preposition.

Regex:

I: VERB (SUBST | ADJ | PREP)
O: OPER (ARG | ARG | OPER)

Statistics

Hits		1-to-n Ambiguity	
PoS	3	**Min**	3
Def	701	**Max**	40
Lem	1177	**Mean**	7.8643
Sen	4257	**StdDev**	5.7572

Rel	Freq
OPER	2105
ARG	1657
HAS_HYPERONYM	575
INVOLVED	353
CAUSES	936
ENTAILS	343

Figure 5.3: General data for transducer 2

When looking at the statistics for generated relations, we see that the largest classes are OPER and ARG . These are symbols in the transducer output sequence that have not been transformed. A high number of OPER relations is interpreted as an indication of an incomplete set of operator words. A high number of ARG relations is interpreted as an indication of an incomplete set of operations performed by the operator words.

By this interpretation, we have an incomplete set of operator words and argument support.

This was to be expected since our method for operator word assignment is somewhat trivial as of now. A number of other semantic relations are generated however; in addition, the set of OPER and ARG relations can be turned into a dataset for a further analysis of candidate operator words and arguments. Such an analysis is not considered in this study.

Some examples of definientia captured by transducer 2 can be seen in Table 5.5.

PoS Pattern Class Matches		
	VERB	SUBST
regne	utføre	talloperasjon
'calculate'	'perform'	'numeric operation'
dekontaminere	fjerne	smittestoff
'decontaminate'	'remove'	'infectious agent'
	VERB	ADJ
fortjene	være	verdig
'deserve'	'be'	'worthy'
skjerpe	gjøre	skarp
'sharpen'	'make'	'sharp'
	VERB	PREP
donere	gi	bort
'donate'	'give'	'away'
søke	lete	etter
'seek'	'look'	'after'

Table 5.5: Some examples of members of PoS pattern classes captured by transducer 2. Definienda are shown in the first column. English translations for every example are shown in quotes.

6 Semi-automatic Transducer Expansion

Using simple transducer patterns along with operator words, we manage to generate a substantial amount of relations. Still, this only accounts for a very limited subset of the dictionary, the size of which depends on the variety the of ways that the explanatory parts of the definitions are formed. PoS pattern classes are not analyzed unless they directly match one of the regexes devised in the initial step. As we recall from tbl:pos-pattern-list-01 in Section 1, we have a large number of small PoS pattern classes. To manually create transducers catching most of these reduces the usability of this method to the point where similar conclusions to the ones made in the DDO pilot study should be done.

We must therefore find a way to give our algorithm some kind of fuzzy matching of PoS pattern classes. Section 6.1 presents some observations that lead up to the devised strategy for transducer expansion explained in Section 6.2 and Section 6.3.

6.1 Observations

We have collected a set of PoS pattern classes P, each representing a unique sequence of PoS tags. Assuming that similar PoS patterns exhibit similar semantic traits, we want for every PoS pattern class $p \in P$ a set of other PoS pattern classes $\{q_1, q_2, \ldots, q_n\} \subseteq P$ that can be said to have the most similar sequence to that of p. To get a measure for this kind of similarity, a local alignment algorithm is applied which is explained in detail in Section 6.2.

We also have a set of operator words $O \subset V$ where V is the vocabulary of our data set, that operate on their surrounding context in a fairly consistent manner. Due to this consistency, one

can assume that sentences that share an operator word behave similarly in some way, at least in the vincinity of that operator word. We can further assume that the degree of similar behaviour will increase as the number of shared operator words between two sentences increase. Since the sentences we are concerned with are grouped into PoS pattern classes based on identical PoS sequences, we should be able to generalize this assumption, transferring these properties of sentences according to operator word occurrence to the class of sentences sharing a PoS sequence. In other words, we can compare PoS pattern classes much in the same way we compare sentences, using the operator words as the members of a bag-of-words vector and comparing the relative frequencies found for each class. The choice of measurement for this kind of similarity is explained in detail in Section 6.3.

After attaining a combined similarity metric, we can expand the initial transducers automatically by analyzing the n most similar PoS pattern classes, creating new input/output regexes that capture more PoS pattern classes.

6.2 The Smith-Waterman Algorithm

The Smith-Waterman algorithm is a local alignment algorithm originally developed to identify similar regions between two nucleotide or protein sequences of different lengths. It has also been applied to similar problems in the NLP fiels (Katrenko et al., 2010). The algorithm takes two sequences A and B as its input and outputs two aligned sequences A' and B' along with an alignment score s defined by the number of operations that was needed to get the optimal local alignment. The algorithm is described in detail in Smith and Waterman (1981).

The algorithm starts by filling a n-by-m matrix M where $n = |A| + 1$ and $m = |B| + 1$. The matrix is initialized by filling the first row and column with 0:

$$M(0, j) = 0 \text{ where } 0 < j < n$$
$$M(i, 0) = 0 \text{ where } 0 < i < m$$

The cells are then traversed row by row and given a score according to the function shown in Equation (5.4). This function makes use of a scoring function w that returns a value based on the operation performed at this point (either match, mismatch, insertion or deletion). In this case, the values returned are as defined in Equation (5.5). The motivation for choosing this particular scoring function is based on the fact that we usually start defining transducers for very short sequences. There is thus little value in looking for shorter sequences. A mismatch is given a lower score than an insertion for the same reason, making sure that the expansion looks for sequences increasing in size and tries to match PoS tags as often as possible.

Additionally, information about the cell with the highest score from the choices in Equation (5.4) is stored in order to create a path for the subsequent traceback.

$$M(i,j) = \max \left\{ \begin{array}{lll} 0 & & \\ M(i-1,j-1) & +w(a_i, b_j) & \text{Match/Mismatch} \\ M(i-1,j) & +w(a_i, -) & \text{Deletion} \\ M(i,j-1) & +w(-, b_j) & \text{Insertion} \end{array} \right\}, 1 \leq i \leq m, 1 \leq j \leq m \tag{5.4}$$

$$w(a_i, b_j) = \left\{ \begin{array}{ll} \text{Match} & = 1 \\ \text{Mismatch} & = -2 \\ \text{Insertion} & = -1 \\ \text{Deletion} & = -3 \end{array} \right. \tag{5.5}$$

When the matrix is completely filled, a traceback through the matrix is performed starting from the cell with the highest score. The traceback follows the path according to the direction stored in the cell (i.e. the cell that was found to have the highest score according to Equation (5.4)), and terminates either when it encounters a score of 0, or when the cell at position $(0, 0)$ is reached. The traceback gathers the coordinates found for every cell reached, inferring insertions, deletions and match/mistmach operations on the original sequences based on the path taken. In addition, the scores for each cell that was traversed in the traceback procedure are accumulated, giving an overall alignment score.

Alignment				
VERB	——	PREP	——	——
VERB	PRON	PREP	KOMMA	VERB
OPER	NIL	OPER	NIL	NIL

Table 5.6: The result of an alignment of VERB PREP and VERB PRON PREP KOMMA VERB along with the aligned relation sequence.

Some modifications are done to the algorithm in order to use its output for transducer expansion. The returned score is normalized and interpreted as a distance measure between the two original sequences. In addition, the relations defined in the transducer output is aligned using the same traceback coordinates as was found for sequence A, giving a third sequence C'. An example is shown in Table 5.6. By displacing the relation symbols the same way as the aligned sequence of PoS tags, the assumption is that patterns containing similar semantic content will be discovered, while parts of the definientia that might cause trouble or demand additional specifications of transducer and/or operator words are ignored.

6.3 Augmenting the Similarity Scores: Bag-of-Words

The scores acquired from the alignment have a low resolution, meaning that a lot of PoS pattern classes will have the exact same score. To add some more nuance to the scores, an additional similarity metric is used. This metric is based on the Jensen-Shannon divergence (Lin, 1991), also known as Information Radius.

A vector is created for each PoS pattern class, acting as a bag-of-words. The vector is constructed from the list of operator words made in the prior step. For each PoS pattern class, the lemma forms of the definientia belonging to that class are counted; given that they also exist in the list of operator words. The PoS pattern classes are then compared using the Jensen-Shannon divergence formula, and the resulting similarity scores are stored alongside the alignment scores. To get the n-best PoS pattern classes for a given PoS pattern class, its potential n-best matches are sorted in descending order using both the alignment score and the similarity score before picking the n PoS pattern classes at the top.

6.4 Example: Expansion of Transducer 2

To show how the algorithm behaves, a 5-best expansion transducer 2 is presented below. A second iteration of expansions is performed, but only the results of the second iteration is shown together with the first iteration in Table 5.10.

The transducer initially matches three PoS pattern classes: " VERB PREP ", " VERB SUBST " and " VERB ADJ ". For each of these classes, the 5 best PoS pattern classes, i.e. the 5 top scoring classes according to the alignment score and Jensen-Shannon divergence are collected and aligned.

The 5 best PoS pattern classes, their alignment with the original PoS pattern class and the scores are shown in Table 5.7.

Alignments for VERB ADJ			**ScrAl**	**ScrSim**
VERB	ADJ	——		
VERB	ADJ	PREP	0.75	$5{,}602\,797\,3 \cdot 10^{-5}$
VERB	ADJ	——		
VERB	ADJ	SUBST	0.75	$5{,}401\,988 \cdot 10^{-5}$
VERB	ADJ	——		
VERB	ADJ	CLB	0.75	$5{,}376\,750\,8 \cdot 10^{-5}$
VERB	——	ADJ		
VERB	ADJ	ADJ	0.75	$4{,}668\,838 \cdot 10^{-5}$
VERB	ADJ			
VERB	UKJ		0.67	$2{,}758\,744\,2 \cdot 10^{-4}$

Table 5.7: 5-best PoS pattern alignments for the PoS pattern class VERB ADJ , along with alignment scores (**ScrAl**) and bag-of-words similarity scores (**ScrSim**) for each alignment.

Inference of New Regexes After aligning all sequences, matrices are constructed based on the results. For each position in the aligned PoS pattern sequences, a column is made, and all variations of input and output symbols for that position are collected. Table 5.8 shows the columns constructed from the previous 5-best alignment step.

Expansion Columns for VERB PREP **)**								
VERB	VERB	OPER	PREP	PREP	OPER	——	SUBST	——
VERB	VERB	OPER	PREP	PREP	OPER	——	ADJ	——
VERB	VERB	OPER	PREP	PREP	OPER	——	PRON	——
VERB	VERB	OPER	——	PREP	——	PREP	PREP	OPER
VERB	VERB	OPER	PREP	PREP	OPER	——	ADV	——
Expansion Columns for VERB SUBST **)**								
VERB	VERB	OPER	SUBST	SUBST	ARG	——	CLB	——
VERB	VERB	OPER	SUBST	SUBST	ARG	——	PREP	——
VERB	VERB	OPER	SUBST	SUBST	ARG	——	PAREN	——
VERB	VERB	OPER	SUBST	SUBST	ARG	——	ADV	——
VERB	VERB	OPER	SUBST	UKJ	ARG			
Expansion Columns for VERB ADJ **)**								
VERB	VERB	OPER	ADJ	ADJ	ARG	——	PREP	——
VERB	VERB	OPER	ADJ	ADJ	ARG	——	SUBST	——
VERB	VERB	OPER	ADJ	ADJ	ARG	——	CLB	——
VERB	VERB	OPER	——	ADJ	——	ADJ	ADJ	ARG
VERB	VERB	OPER	ADJ	UKJ	ARG			

Table 5.8: Expansion columns generated from the 5-best transducer expansion of transducer 2

The columns are subsequently collapsed into sub-regexes using some simple heuristics:

1. All identical input-output pairs in a column are condensed into one input-output pair.

2. If there are input pairs with matching inputs and one or more of these pairs have NIL as the output, the pairs with NIL as output are discarded, hence prioritizing occurences of input matches that actually produce some output.

3. All unique input-output pairs (if they are more than one) are interpreted as disjunctions.

4. If a column contains an input-output pair where both the input and output is NIL, a zero-or-one operator "?" is added.

Collapsed Columns for VERB PREP		
VERB OPER	PREP NIL	ADV NIL
	NIL NIL	PREP OPER
	PREP OPER	PRON NIL
		ADJ NIL
		SUBST NIL
		NIL NIL
New input/output regexes inferred from the collapsed columns		
VERB	PREP ?	(ADV | PREP | PRON | ADJ | SUBST) ?
OPER	OPER ?	(NIL | OPER | NIL | NIL | NIL) ?
Collapsed Columns for VERB SUBST		
VERB OPER	UKJ ARG	ADV NIL
	SUBST ARG	PARENTES-SLUTT NIL
		PREP NIL
		CLB NIL
		NIL NIL
New input/output regexes inferred from the collapsed columns		
VERB	(UKJ | SUBST)	(ADV | PAREN | PREP | CLB) ?
OPER	(ARG | ARG)	(NIL | NIL | NIL | NIL) ?
Collapsed Columns for VERB ADJ		
VERB OPER	UKJ ARG	ADJ ARG
	ADJ NIL	CLB NIL
	NIL NIL	SUBST NIL
	ADJ ARG	PREP NIL
New input/output regexes inferred from the collapsed columns		
VERB	(UKJ | ADJ) ?	(ADJ | CLB | SUBST | PREP)
OPER	(ARG | ARG) ?	(ARG | NIL | NIL | NIL)

Table 5.9: Collapsed columns and new input-output regex pairs generated from the results from Table 5.8.

Finally, the sub-regexes are concatenated, resulting in new input-output strings for a new set of transducers. Table 5.9 shows how the columns from Table 5.8 are collapsed and regexes for three new transducers are made.These new transducers will match a higher number of PoS pattern classes. The amount of matched PoS pattern classes has gone from three PoS pattern classes in the initial transducer to 25 PoS pattern classes in the three inferred transducers. Due to the nature of the alignment algorithm and the operator word functionality, the new transducers generally add relations only to similar structures within the definientia while ignoring previously unseen combinations.

The expansion procedure decreases the amount of work required to discover patterns that occur throughout the dictionary. The scrutinization of lists over PoS patterns and glosses are mostly transferred from the user to the program, enabling the user to concentrate on the general properties of operator words. The process can be repeated on the generated transducers, enabling an iterative pseudo-bootstrapping over the source material.

After the first expansion of transducer 2, we perform a second iteration of transducer expansion, meaning that the inferred transducers are expanded in the same way that the original manually created transducer was expanded.

Rel	**2.1**	**2.1.1**	**2.2**	**2.2.1**	**2.3**	**2.3.1**
OPER	18087	29043	0	554	0	554
ARG	0	0	2263	5800	7972	14872
ENTAILS	501	694	370	441	133	416
INVOLVED	47	85	777	900	0	751
HAS_HYPERONYM	189	1684	1315	1792	1564	1787
CAUSES	0	0	942	1050	997	1050

Table 5.10: Relation frequency for all expansions of transducer 2. The relation type is shown in the leftmost column.

The third and final transducer generated manually was devised by looking at the largest PoS pattern classes not captured by any transducer. We remember from Section 1.1 that some PoS pattern classes contain only non-explanatory glosses. Taking these out of the consideration, the choice fell upon the pattern `VERB KOMMA VERB ADJ`. The transducer was specified match only this PoS pattern class and subsequently put through a 10-best expansion.

Rel	**3**	**3.1**
ARG	283	1141
ENTAILS	108	232
INVOLVED	0	259
HAS_HYPERONYM	721	11248
CAUSES	115	145

Table 5.11: Relation frequency for transducer 3 and its expansion 3.1. The relation type is shown in the leftmost column.

7 Summary

All transducers, both manually created and the expansions, are listed in Table D.1. A summary of the measurements done in the previous sections is shown in Table 5.12.

	General Stats					1-to-n Ambiguity			
Id	**PoS**	**Def**	**Lem**	**Sen**	**Rel**	**Min**	**Max**	**Mean**	**StdDev**
1	4	1866	2850	8381	17683	3	31	5.7788	3.8326
2	3	701	1168	3656	4970	3	36	6.8837	5.2342
2.1	28	2147	1760	4633	16054	3	33	10.6530	7.0458
2.1.1	65	2480	2178	5700	26195	3	58	11.8315	7.7909
2.2	24	1764	1166	3298	4653	3	36	6.3740	4.6827
2.2.1	67	2463	1506	4309	7253	3	36	7.3411	5.3609
2.3	29	2115	919	2099	8197	3	36	10.8858	5.8813
2.3.1	63	2432	1827	4565	19430	3	36	9.8185	5.9721
3	1	71	188	856	1065	3	36	7.5532	5.8083
3.1	19	1360	2090	6364	10711	3	36	6.7577	5.2073

Table 5.12: Summary of initial measures for all transducers. The columns show the transducer id (**Id**), the number of captured PoS patterns, definitions, lemmas and senses (**PoS**, **Def**, **Lem** and **Sen**, respectively). **Sen** gives the number of semantic relations generated (OPER and ARG relations omitted). The last four columns give the 1-to-n ambiguity measures. **Min** and **Max** gives the minimum and maximum 1-to-n ambiguity encountered, while **Mean** and **StdDev** gives the mean and standard deviation.

Chapter 6

Graph Generation

So far, we have devised three transducers, two of which have been subject to transducer expansion. We now move on to the task of processing the resulting set of relations generated by these transducers. We start by defining three types of graphs in Section 1 in order to establish the formal components we need for this stage. We then proceed to the removal of non-semantic relations and the merging of the relation data generated by all transducers in Section 2. After this step, some graph manipulation steps done for the purpose of reducing the 1-to-n target ambiguity. This is described in Section 3, **??** and Section 3.2. Finally, an attempt at automatic synset inferrence is described in Section 3.3.

1 Graph Types

We define three types of graphs that we need: the *sense graph*, *lemma graph* and *synset graph*

There are three types of graphs with slightly different characteristics available. Formal descriptions of the graphs as well as examples follow:

1.1 Sense Graph

The sense graph is the initial graph created by the transducers. Each node is identified by an integer corresponding to a definition ID in BOB. The graph corresponds to the 1-to-n ambiguity in the form of one or more edges from one source lemma to the possible target lemmas. If there is only one edge, the relation is unambiguous. This is the most detailed of the three graphs, and it serves as the foundation for the other two graphs.

> **Definition:** A sense graph is a directed graph represented as a 4-tuple (N, L, λ, R) where:
>
> N is a finite set of nodes identified by a sense ID
>
> L is a finite set of strings that represent the lemmas occurring in the graph
>
> λ is function $N \rightarrow L$, mapping a sense ID to a lemma
>
> R is a set of labeled edges (relations) $\{r_1, r_2, ...r_n\}$ where $r = \{(x, y)|r(x, y)\}$ and $x, y \in N \times N$.

A visual representation of a small sense graph is shown in Figure 6.1.

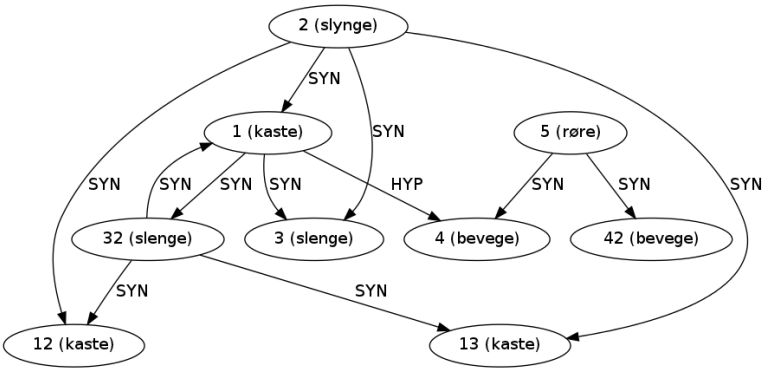


Figure 6.1: An example of a sense graph.

1.2 Lemma Graph

The lemma graph gives an overview over the concepts and is used for N things. It is used for generating lists showing the general relation between two lemmas, omitting information about sense. Each node is represented by a string. Every node also contains a list of integers, each integer representing a possible sense for the lemma in question.

> **Definition:** A lemma graph is a directed graph represented as a 4-tuple (N, I, λ, R) where:
>
> N is a finite set of nodes identified by a string (i.e. the lemma)
>
> I is a finite set of integers that represent sense IDs
>
> λ is function $N \rightarrow I$, mapping a lemma to a sense ID.
>
> R is a set of labeled edges (relations) $\{r_1, r_2, ...r_n\}$ where $r = \{(x, y)|r(x, y)\}$ and $x, y \in N \times N$.

A visual representation of a lemma graph is shown in Figure 6.2. Observe the difference between the unique identificator for the nodes in this graph as compared to the unique identificator in the sense graph.

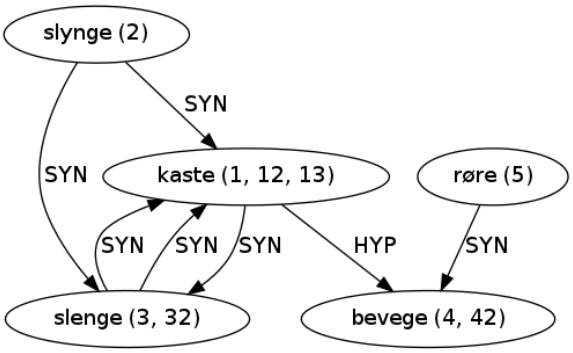


Figure 6.2: An example of a lemma graph

1.3 Synset Graph

The synset graph represents the final graph, ready to be translated into a wordnet. It is an abstraction of the sense graph. Every node contains a list of integers representing senses that are considered to be synonyms. As in the sense graph, each integer is mapped to a lemma, corresponding to the definition id's in BOB.

> **Definition:** A synset graph is a directed graph represented as a 6-tuple $(S, L, I, \lambda, \gamma, R)$ where:
>
> S is a finite set of nodes identified by an integer (synset ID)
>
> L is a finite set of strings that represent lemmas
>
> I is a finite set of integers that represent sense IDs.
>
> λ is function $I \rightarrow S$, mapping a sense ID to a synset.
>
> γ is function $I \rightarrow L$, mapping a sense ID to a lemma.
>
> R is a set of labeled edges (relations) $\{r_1, r_2, ...r_n\}$ where $r = \{(x, y)|xry\}$ and $x, y \in S \times S$.

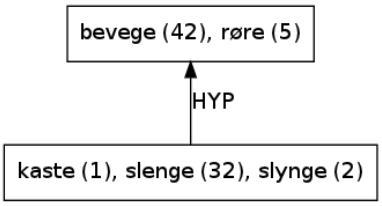


Figure 6.3: An example of a synset graph

2 Cleanup and Merge

We start by removing the non-semantic relations OPER and ARG in order to focus our measurements on the relations that are actual candidates for the final wordnet data. As we remember from the observations done in Chapter 9, a large number of relations were not transformed due to an incomplete set of operator words and arguments. As can be seen in Table 6.1, the reduction is substantial but we still have a number of relations to work with.

After this clean-up, the relation data from all transducers are merged into one single sense graph by taking the union of all sets of relations. This will reduce the number of relations further because of overlapping transducer results. The resulting sense graph statistics can be seen in **??**

A summary is in order to recap what we have done so far. We have created three transducers manually. Transducer 1 has been assigned HAS_SYNONYM relations directly and has not been subject to transducer expansion. Transducer 2 used an input/output regex that originally matched three PoS pattern classes. This transducer was subject to two iterations of transducer expansion, the first iteration performing a 5-best expansion while the second iteration performed a 10-best expansion. The third transducer targeted one specific PoS pattern

Id	Before	After
1	20864	20864
2	5969	2207
2.1	18824	737
2.1.1	31506	2463
2.2	5667	3404
2.2.1	10537	4183
2.3	10666	2694
2.3.1	19430	4004
3	1227	944
3.1	13025	11884

Table 6.1: Relation count after removing non-semantic relations.

Transducer merged: The sense graph resulting from the merging of all transducers

Regex:

Statistics

Hits		1-to-n Ambiguity	
PoS	103	**Min**	3
Def	4033	**Max**	64
Lem	5414	**Mean**	9.0227
Sen	16829	**StdDev**	7.5911

Rel	Freq
HAS_SYNONYM	20864
HAS_HYPERONYM	13063
INVOLVED	1244
ENTAILS	1430
CAUSES	1195

Figure 6.4: General data for transducer merged

class and was subject to a single 10-best expansion. This has resulted in the targeting of 103 PoS pattern classes in total. The total sum of members for these PoS classes amounts to 4033 definientia. 5414 unique lemmas have been encountered in the targeted definienda and definientia, pointing to a total of 16829 possible senses. We will now attempt to reduce the 1-to-n ambiguity.

3 Graph Manipulation

Statistics on the resulting sense graph are compiled to give an overview of the structure and quality, as well as to provide a foundation for graph manipulation in order to improve its quality before being subject to manual post-processing. A number of properties are interesting in this respect, and are presented in the following section.

3.1 Disambiguation by PoS Tags

The first disambiguation step is to reduce the number of potential senses for target words according to their PoS tag. As described in Chapter 4, each definition is part of an article with a defined grammatical code which is converted to an equivalent PoS symbol compatible with OBT+Stat. Each word in the definiens is tagged with a PoS tag. We now assume that a target word with a certain PoS tag should point to articles describing definienda with identical lemmas, *and* an identical grammatical class. For example, a target word like *finne '(to) find'* tagged as a verb is much more likely to point to definitions for *finne* that describe a verb, rather than a noun - it makes no sense that the verb *finne* would have a candidate sense belonging to the noun *finne*, describing a person from Finland.

By finding the translated PoS tag for articles containing the senses pointed to by a target word, we can exclude the ones that are unlikely to be sense candidates for that particular target word.

3.2 Disambiguation by Cycles

Cycles that occur in HAS_HYPERONYM and HAS_SYNONYM relations are interesting phenomena. They do not occur very often, but when they do, they should be investigated. To discover such cycles, a depth-first search was executed on the sense graph. When a previously visited node is revisited, we know that a cycle has been discovered.

Finding cycles in HAS_SYNONYM chains is not suprising when the source material is taken into consideration. When making lists of synonyms for a definition, the closest synonyms will spring to mind. Given two closely synonymous words, the probability of them appearing in each others definiens should be significant.

Cycles occuring in HAS_HYPERONYM chains are more interesting. This can be interpreted in two ways. Either, the dictionary definition is inconsistently formed; it should rather be formed according to the way synonyms are represented. Or, we have encountered two verbs whose relation is hard to decide upon. Given that the wordnet hierarchy for verbs tends to be shallow and bushy rather than treelike, we assume that in both cases a HAS_SYNONYM relation should be preferred.

The final graph contains 218 HAS_SYNONYM cycles and 31 HAS_HYPERONYM cycles. This does not amount to much, but as we are working on an incomplete sense graph we can assume that these numbers will increase with a more complete set of operator words and a larger transducer coverage of the dictionary.

Cycles discovered in HAS_SYNONYM chains contribute to the disambiguation of the 1-to-n target ambiguities for the involved nodes. In a HAS_SYNONYM cycle, the relation that closes the loop from the last node in the cycle to the first one gives rise to an unambigious path, as can be seen in **??**. This is due to the 1-to-n ambiguity and the symmetric property of HAS_SYNONYM relations. The relations that make up such a path are kept while the others are discarded.

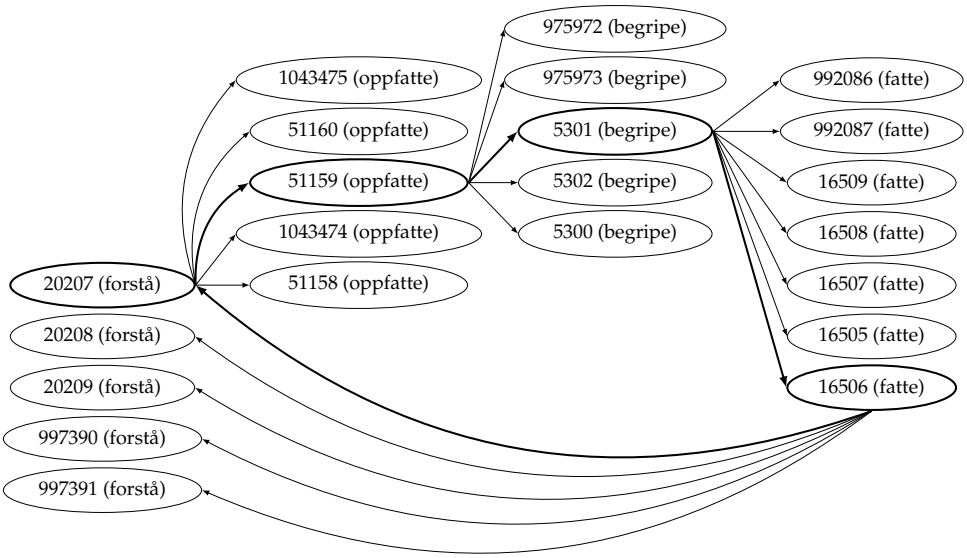


Figure 6.5: Example of disambiguation by synonym cycle

Cycles discovered in HAS_HYPERONYM chains indicate inconsistencies in the dictionary and/or concepts that are difficult to define hierarchically. In most of these cases, it is relatively safe to assume that a HAS_SYNONYM relation should replace the HAS_HYPERONYM relation. We thus change the HAS_HYPERONYM relations making up the cycle into HAS_SYNONYM relations and disambiguate these paths in the same way as for HAS_SYNONYM paths, reducing the 1-to-n target disambiguation. Table 6.2

	Cleanup	**DPoS**	**DHypCyc**	**DSynCyc**
Rel	37796	32135	31817	29965
AMin	3	3	3	1
AMax	64	62	62	58
AAvg	9.0227	7.7081	7.6318	7.2379
AStd	7.5911	6.3011	6.1934	6.0339

Table 6.2: Results from cleanup and partial disambiguation the final sense graph.

3.3 Inferring Synsets from HAS_SYNONYM Relations

Transducer 1 was created for the purpose of generating a large sum of HAS_SYNONYM relations. The assumption was that this would provide a solid ground for synset creation. Since the HAS_SYNONYM relation is both symmetric and reflexive, the assumption was that the inference of synsets would consist of iterating through each HAS_SYNONYM relation, adding

both the LHS and RHS to a synset if one of them was found to be a member of some synset, or to create a new synset if no such membership was discovered.

It turns out that this strategy is too agressive. When applied to the resulting sense graph where the majority of the HAS_SYNONYM relations have a high degree of 1-to-n target ambiguity, the inferred synsets start out with an acceptable size. But as the synsets grow bigger, so does the probability of merging two separate synsets by a single invalid relation. After processing all relations, most senses have been merged into one enormous synset. To make sure that the conversion of HAS_HYPERONYM cycles to HAS_SYNONYM cycles was not the cause, the synset inference was done for both versions. The same phenomenon occurred in both graphs.

The inevitable conclusion thus seems to be that the proposed method of synset inference must be applied to the resulting data from a graph that has a much lower 1-to-n target ambiguity and a low rate of invalid relations. The automatic reduction of 1-to-n target ambiguity done in was not sufficient to prevent this from happening.

Chapter 7

Manual Post-Processing

Evaluating a wordnet when there is no previous project to compare it with is not a trivial task. In the event that an existing wordnet is available, it may be used as a near-gold standard. This will however not give much information about any new relations gained from the new wordnet as it will be difficult to determine whether the new relation is a false positive or a missing relation in the near-gold standard. Besides, if there is an already existing wordnet available, the motivation for creating a new one comparable to the existing one is obviously not as big.

The best evaluation approach will in most cases be to establish an upper-bound measure based on a set of human annotators, evaluating either all the generated relations or a randomly selected subset. In the latter case, an approximate measure of the quality of the wordnet can be given with a confidence level depending on the size of the subset. Furthermore, by combining the results from the annotators and using some agreement measure, one can attempt to identify difficult areas for both the method and the source material. Finally, the manual word sense disambiguation can also be incorporated in this step, reducing the total amount of post-processing work somewhat. Section 1 gives a general description of the post-processing and annotation task. Section 2 presents the application developed for the annotators. The post-processing done by the annotators for this study is investigated in Section 3.

One important requirement for such an evaluation approach is that it should be efficient to the point where it is clearly beneficial in comparison to building a wordnet manually. Measurements of the time spent on post-processing relations are presented in Section 3.4.

1 Description of the Post-processing Step

The goal of the post-processing step in this study is to mark relations as either correct or incorrect the final output of DICT2WN, and to disambiguate relations considered to be correct. A lemma graph created from the final sense graph is used. Each relation between an unambiguous definiendum and a possibly ambiguous target word in its corresponding definiens is defined either to be valid or invalid. If it is valid, a disambiguation is done where one or more of the senses contained in the node representing the target word is chosen to be correct.

In addition, a timestamp for every operation on the evaluated relation set is registered in order to get an approximate measure for the time spent on the manual labor.

2 The Post-Processing Application: DICT2WNPP

DICT2WN contains functionality for exporting its generated graphs to a separate SQLite database for further processing. An application was developed to make the manual post-processing as efficient and easy as possible. The application is written in Java, presenting the user with a graphical user interface showing one relation at a time, along with a list of one or more possible nodes that the relation can point to. Since each relation has a 1-to-n ambiguity, the possible choices of nodes presented to the user is restricted to the right-hand side of the relation. The user thus has the possibility of either disambiguation (thereby accepting the relation) or invalidation. A set of possible actions that the user can perform are explicitly defined, and triggered based on the users actions on the interface.

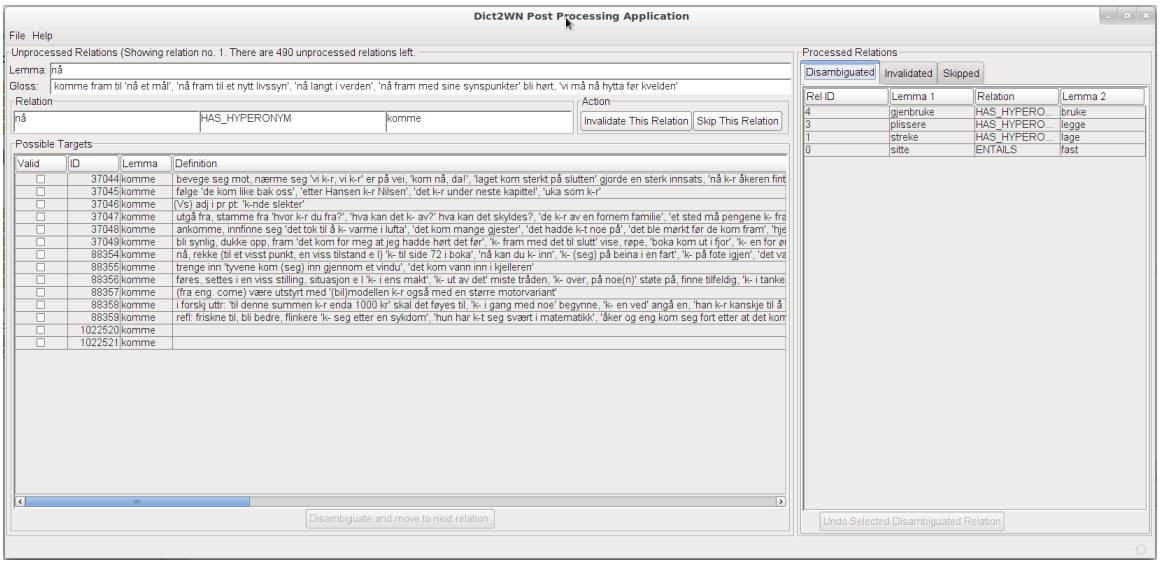


Figure 7.1: Screenshot of the post-processing application for Dict2WN

Whenever the user performs one of the defined actions, the action type and a timestamp corresponding to the point in time when the user performed the action is stored. This list of actions and corresponding timestamps serves as the foundation for the workload evaluation. By examining the amount of time that has passed between each event, one can get an average measure of the time spent post-processing a relation. The actions defined for the post-processing is defined in Table 7.1.

To prevent coffee breaks etc. from affecting the workload statistics, intervals at 5 minutes and higher were ignored. This was considered a safe threshold based on tests during the creation of Dict2WNPP, measuring the amount of time it takes to select between the possible actions, including a search for the lemmas in question on the web and in BOB itself.

It was deemed unrealistic to expect the participants to completely post-process all the relations. Because of this, the list of relations is ordered in a way that corresponds to the frequency distribution of PoS pattern classes captured by the method. The same list of relations is sent to each participant. The post-processing result with the smallest amount of relations defines the number of relations to use for the main agreement statistics. Higher numbers of relations are treated the same way, but with fewer annotators. Due to the ordering of the relations, this will nonetheless ensure that the agreement rate is based on a set of random trials that roughly correspond to the overall distribution of the result as a whole.

Name	Description
DISAMBIGUATE	Registered when a user is done selecting one or more possible relations for an ambiguous relation.
INVALIDATE	Registered when a user requests the next relation without selecting any of the possible relations for the current relation.
UNDO_DISAMBIGUATION	Registered when a user selects a relation from the list of disambiguated relations.
UNDO_INVALIDITATION	Registered when a user selects a relation from the list of invalid relations.
SKIP	Registered when a user skips a relation without invalidating or disambiguating it.

Table 7.1: Overview of the possible actions that can be performed by the user of the post-processing application made for Dict2WN.

3 Annotation Study

A total of three participants took part in the post-processing step, deciding upon correct and incorrect relations and annotating correct relations with the appropriate target senses. All three annotators have experience with lexical semantics, and are given guidelines as specified in the manual for DICT2WNPP (see Appendix B). Table 7.2 gives an overview over the post-processing done by the three annotators. The total number of post-processed relations can be seen in the *Total* column, while the columns *Disambiguate*, *Invalidate* and *Skip* show the individual distribution of actions for each annotator.

Annotator	Disambiguate		Invalidate		Skip		Total
1	402	(0.703%)	168	(0.294%)	2	(0.003%)	572
2	176	(0.680%)	65	(0.251%)	18	(0.069%)	259
3	460	(0.755%)	149	(0.245%)	0	(0.000%)	609

Table 7.2: Individual results for each annotator participating in the post-processing. The last column (Total) represents the total number of relations post-processed by each individual annotator.

Initially, the results seem to be somewhat positive, although the disambiguation ratio could of course have been even higher. It should be noted that the one annotator with the highest disambiguation ratio happens to be the author of this thesis, something that shows the importance of having more than one annotator when trying to evaluate the outcome of such a project. In addition, the two annotators with the highest disambiguation ratio have both worked together on lexicographic projects before, and have both been working on NorNet specifically. The annotator with the lowest disambiguation ratio is the one that could said to be the least biased, having worked on entirely different projects than the two other annotators. The importance of a diverse set of annotators is thus shown quite clearly here.

To investigate how often the annotators agree upon the possible actions to perform per relation, we need to dive into the details of the results. The individual results attained from the three annotators were combined into two data sets, referred to as the *3-annotator* and *2-annotator*

data set from now on. The 3-annotator data set contains 259 relations that were post-processed by all three judges. The 2-annotator data set contains the 259 relations from the 3-annotator set, plus 313 relations that were post-processed by two annotators. The results from the third annotator contributing to the 3-annotator data set is only considered when analyzing the 3-annotator data set.

3.1 Using Fleiss' Kappa for Agreement Measures

There are a number of ways to measure agreement between a set of annotators (e.g. annotators, psychologists, etc.), the more popular ones making use of the Kappa coefficient as defined in Equation (7.1) (Carletta, 1996):

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{7.1}$$

where $P(A)$ is the relative observed agreement among a set of annotators and $P(E)$ is the proportion of times one would expected the annotators to agree by chance.

In many cases, Cohen's Kappa measure is used for measuring pairwise agreement between two annotators. In our case however, we are dealing with more than two annotators. I thus have the choice of either making some comparison matrix of Cohen's Kappa measures for all three annotators, finding an adequate average measure for the matrix as a whole, or using an agreement measure generalized to two or more judges. I have decided to make use of the latter.

Fleiss' kappa is such a generalization, giving a measure for the agreement between a fixed number of annotators for some number of categorical ratings. It is defined through a number of steps (Fleiss et al., 1971), all of which are described in the context of the annotation process explained earlier. Measurements from both the three-annotator dataset and the two-annotator dataset are also provided at each step.

Using the same notation as Fleiss, we let the subscript $i = 1, \ldots, N$ represent the relations (N represents the total number of relations in the dataset), and the subscript $j = 1, \ldots k$ represent the categories. In our case, $k = 3$ for the ordered set of categories {**Disambiguate**, **Invalidate**, **Skip**}.

We first find the proportions p_j of all assignments that were assigned to the jth category .

$$p_j = \frac{1}{Nn} \sum_{i=1}^{N} n_{ij} \tag{7.2}$$

where n is the number of actions taken per relation (which in our case is equal to the number of annotators) and n_{ij} is the number of raters who assigned the ith subject to the jth action. The resulting distribution of category assignments for our two datasets are shown in Table 7.3.

Dataset	Disambiguate	Invalidate	Skip
3-Annotator	0.667	0.310	0.023
2-Annotator	0.726	0.272	0.000

Table 7.3: Per-action agreement ratios (p_j) for the two post-processing sets, calculated with Equation (7.2).

The number of skipped relations turned out to be quite low and is therefore not considered to be very usable for any general remarks, other than as a part of the general Kappa measure. They are however included for the sake of completeness wherever neccessary. We can see that

these numbers seem to stand in proportion to the ones given in the individual annotator results shown in Table 7.2.

After finding proportions of the assignments, we move on to measuring the extent of agreement for each relation, denoted by P_i where i is the relation number:

$$P_i = \frac{1}{n(n-1)} \left(\sum_{j=1}^{k} n_{ij}^2 - n \right) \tag{7.3}$$

As before, n is the number of actions taken per relation. i iterates through each relation, while n_{ij}^2 represents the number of annotators deciding upon the j'th action for the i'th relation.

An excerpt of the agreement list for all relations (P_i's) is shown in Table 7.4. The complete lists of P_i agreements are not included as they are not very informative.

3-ANNOTATOR							
Id	**Relation**			**Dis**	**Inv**	**Skip**	P_i
54	oversende	HAS_HYPERONYM	sende	3	0	0	1.00
55	undres	ENTAILS	over	0	3	0	1.00
56	klappe	HAS_HYPERONYM	legge	3	0	0	1.00
57	rokere	HAS_HYPERONYM	utføre	2	1	0	0.33
58	urbanisere	CAUSES	bymessig	3	0	0	1.00
2-ANNOTATOR							
Id	**Relation**			**Dis**	**Inv**	**Skip**	P_i
114	tolke	ENTAILS	uttrykk	1	1	0	0.0
115	telefonere	HAS_HYPERONYM	ringe	0	2	0	1.0
116	skeivle	HAS_HYPERONYM	bringe	2	0	0	1.0
117	trimme	HAS_HYPERONYM	mosjonere	1	1	0	0.0
118	vringle	ENTAILS	vrang	2	0	0	1.0

Table 7.4: An excerpt of the per-relation agreement table P_i generated using Equation (7.3). Each row represents one relation, each identified by a number (the **Id** column). The **Relation** column shows the actual relation, presented at lemma level. The **Dis**, **Inv** and **Skip** columns show the number of annotators that performed a disambiguate, invalidate or skip action respectively, for the relation. The P_i shows the agreement score, with 0 n̄o agreement and 1 f̄ull agreement.

A summary of the information from the P_i agreement tables is presented in Table 7.5, listing the ratio of fully agreed, partially agreed and completely disagreed decisions for the post-processed relations. Partially agreed decicions are not applicable to the two-annotator data set since two annotators will always wholly agree or disagree.

Dataset	**Full Agreement**		**Partial Agreement**		**No Agreement**	
	Count	**Perc**	**Count**	**Perc**	**Count**	**Perc**
Three-Annotator	127	49.03%	126	48.65%	6	2.32%
Two-Annotator	384	67.13%	N/A	N/A	188	32.87%

Table 7.5: Agreement ratio summary created from the list of per-relation agreements (calculated using Equation (7.3) and subsequently summed up).

The results show that out of 259 post-processed relations, only about half are fully agreed upon by all annotators in the three-annotator set. The ratio for full agreement in the two-

annotator set is higher, but since there is no possibility for partial agreement between two annotators, these two measures should probably not be trivially compared. We will thus focus mostly on the three-annotator data set.
We find the overall agreement measure $\bar{P}$ by taking the mean of the P_is:

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i \tag{7.4}$$

This gives us the probability of some relation being given the same category by two different annotators. The results for our two datasets can be found in the $\bar{P}$ column in Table 7.6. The values for the two datasets are close together indicating that one can apply the conclusions drawn on the smaller three-judge dataset to the bigger two-judge dataset.
Going further, we form a null hypothesis H_0 that the annotators select categories for each subject randomly:

$$\bar{P}_e = \sum_{j=1}^{k} p_j^2 \tag{7.5}$$

where once again, j iterates over the categories $1 \ldots k$ and p_j^2 is the squared proportion of all assignments that were assigned to the j'th category. The results of this is given in the $\hat{P}_e$ column in Table 7.6.
By now we have enough measures to arrive at our definition of the general Kappa statistic:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{7.6}$$

The resulting κ measurements for our two datasets along with their variance and standard error are shown in the κ, $Var(\kappa)$ and $SE(\kappa)$ columns respectively in Table 7.6.

Dataset	$\bar{P}$	$\bar{P}_e$	κ	$Var(\kappa)$	$SE(\kappa)$	$\kappa/SE(\kappa)$	**Confidence Interval**
3-Annotator	0.6525	0.5412	0.2426	0.0019	0.0439	5.5304	(0.15664437 0.32862565)
2-Annotator	0.6713	0.6016	0.1751	0.0026	0.0514	3.4082	(0.07440292 0.27579758)

Table 7.6: Overview of the general measures for the Fleiss' Kappa Statistics. The columns show the overall agreement measure ($\bar{P}$), the overall probability for a random decision ($\bar{P}_e$), the Kappa measure for the dataset as a whole (κ), the variance ($Var(\kappa)$), standard error ($SE(\kappa)$) and final z score ($\kappa/SE(\kappa)$) for the Kappa measure.

The overall Kappa measures for the two datasets are not very encouraging, the 3-annotator dataset being the only one falling within the bottom end of what is generally considered to be fair agreement (0.20 - 0.39). However, the confidence intervals ($\kappa \pm 1.96SE(\kappa)$) do not cross 0, which indicates that there is in fact at least a small statistic significance in the agreement rates. Still, the overall impression so far is that there is a general problem deciding upon which relations are correct and which are not.

Kappa Measures for Individual Actions

Fleiss describes a way to generate kappa values for each individual category. In our case, the categories are the possible actions to take for some relation, i.e. *Disambiguate*, *Invalidate* and *Skip*. The results for these measurements for both data sets are shown in Table 7.7.

3-ANNOTATOR							
Action	$\sum_i n_{ij}^2$	p_j	$\bar{P}_j$	κ_j	$Var(\kappa_j)$	$SE(\kappa_j)$	$\kappa_j/SE(\kappa_j)$
Disambiguate	1292	0.6667	0.7471	0.2413	0.0208	0.1441	1.6751
Invalidate	481	0.3102	0.4979	0.2722	0.0088	0.0940	2.8954
Skip	18	0.0232	0.0000	-0.0237	0.0183	0.1351	-0.1755
2-ANNOTATOR							
Action	$\sum_i n_{ij}^2$	p_j	$\bar{P}_j$	κ_j	$Var(\kappa_j)$	$SE(\kappa_j)$	$\kappa_j/SE(\kappa_j)$
Disambiguate	1475	0.7264	0.7750	0.1775	0.0282	0.1680	1.0570
Invalidate	435	0.2719	0.3987	0.1742	0.0123	0.1108	1.5727
Skip	2	0.0017	0.0000	-0.0018	0.5061	0.7114	-0.0025

Table 7.7: Overview of agreement measures for each individual action. The columns show the squared sum of decisions for each action ($\sum_i n_{ij}^2$), the per-action agreement ratio as (p_j), the overall agreement measure for the action, ($\bar{P}_j$), the Kappa measure for the action (κ_j) along with the variance ($Var(\kappa_j)$), standard error ($SE(\kappa_j)$) and final z score $\kappa_j/SE(\kappa_j)$.

It seems that *Invalidate* is the action that is the most agreed upon. The overall ratio of invalidations in the three-annotator data set are at approximately 31% with a Kappa measure of 0.2722 and a low standard error. This is perhaps the best estimate we can do for the results of DICT2WN.

3.2 Relation Frequency Distributions

Frequency distributions over relations found in the annotator datasets were compiled in order to gain some insight into why the agreement ratios turn out the way they do. First, statistics for the relations found in the sets of full, partial and no agreement were calculated; disregarding the type of action being done on each. The results are shown in Table 7.8.

3-ANNOTATOR								
Full Agreement			**Partial Agreement**			**No Agreement**		
Relation Type	**Freq**	**Perc**	**Relation Type**	**Freq**	**Perc**	**Relation Type**	**Freq**	**Perc**
HYP	54	42.52%	HYP	78	61.90%	HYP	4	66.67%
SYN	28	22.05%	ENTAILS	18	14.29%	INVOLVED	1	16.67%
INVOLVED	16	12.60%	SYN	18	14.29%	SYN	1	16.67%
ENTAILS	16	12.60%	INVOLVED	9	7.14%			
CAUSES	13	10.24%	CAUSES	3	2.38%			
2-ANNOTATOR								
Full Agreement			**Partial Agreement**			**No Agreement**		
Relation Type	**Freq**	**Perc**	**Relation Type**	**Freq**	**Perc**	**Relation Type**	**Freq**	**Perc**
HYP	160	41.67%	N/A			HYP	128	68.09%
SYN	109	28.39%	N/A			ENTAILS	24	12.77%
INVOLVED	50	13.02%	N/A			SYN	24	12.77%
ENTAILS	36	9.38%	N/A			INVOLVED	10	5.32%
CAUSES	29	7.55%	N/A			CAUSES	2	1.06%

Table 7.8: Frequency lists over relations found in the 3-annotator data set, according to agreement.

Decisions for hyperonym relations seem to be difficult to agree upon. For the 3-annotator data set, the ratio of partially agreed decisions on HAS_HYPERONYM relations surpass the ratio of fully agreed ones by almost 20%. The ENTAILS relations also seem to have a high probability of disagreement. The results for the other relations imply that the annotators tend to agree more often than they disagree. Similar observations can be made for the 2-judge dataset. Although there is a lower number for no agreement regarding decisions for hyperonym relations, it still occupies the vast majority of relations with no agreement.

3.3 Disambiguation Agreement Measure

The second use of Fleiss' Kappa measure is for the agreement on the disambiguation done for all the relations marked as disambiguated by all participants, i.e. fully agreed validated relations.

Using the measure much in the same way as in the previous section, N is the total number of possible disambiguations (i.e. each possible sense-pair for a pair of lemmas), n is the number of decisions made per sense-pair, and k is the number of categories. For our purpose, the categories are *valid* (a disambiguated relation) and *invalid*. The number of fully agreed relations are 127 and 384, for the three-annotator dataset and two-annotator dataset respectively.

The number of possible sense-pairs are quite high (971 and 2778 for the three-annotator and two-annotator datasets respectively) and do not add much relevant information at this point, thus tables for each individual result are not included. A summary for the p_j measure is presented in Table 7.9.

Dataset	Valid	Invalid
Three-Annotator	0.478	0.522
Two-Annotator	0.387	0.613

Table 7.9: p_j measurements for the two post-processing sets (disambiguation).

It is important to note that the high ratio for the **Invalid** category is to be expected. Since most relations have 1-to-n target ambiguity (except for the ones disambiguated automatically), one would actually prefer a high **Invalid** ratio and a low **Valid** ratio as this means that the relations have been disambiguated as much as possible.

Moving on as before onto the extent of agreement P_i for the ith subject, we find the amount of fully agreed disambiguations, partially agreed disambiguations and disambiguations with no agreement (presented in Table 7.10).

Dataset	Full		Partial		None	
	Freq	Perc	Freq	Perc	Freq	Perc
Three-Annotator	900	92.69%	71	7.31%	0	0.00%
Two-Annotator	2646	95.25%	N/A	N/A	132	4.75%

Table 7.10: P_i measurements for the two post-processing sets (disambiguation).

The overall agreement measures $\bar{P}$ for the two datasets are shown in the $\bar{P}$ column in Table 7.11, and turn out to be almost identical. The final kappa statistic is also very close. The only noticeable difference between the two datasets in this respect is the probability for for

random assignments of categories (the column $\hat{P}_e$), which is 0.50 for the three-annotator dataset and 0.53 for the two-annotator dataset.

Dataset	$\bar{P}$	$\hat{P}_e$	κ	$Var(\kappa)$	$SE(\kappa)$	$\frac{\kappa}{SE(\kappa)}$
3-Annotator	0.951	0.501	0.902	0.001	0.036	25.076
2-Annotator	0.953	0.526	0.900	0.00009	0.001	94.978

Table 7.11: Overview of the general measures for the Fleiss' Kappa Statistics for the disambiguation stage.

This shows that in the event that annotators agree fully on a valid relation, they will in almost all cases agree on the choice of target sense(s). One reason for this might be that the dictionary data will give the annotators some bias, pointing them towards the right sense by the choices they are given for the possible target senses. This could be interpreted as a proof that the quality of the post-processing will rely just as much on the quality of the dictionary as the rest of the method. If there is no suitable target sense based on the definitions presented to the annotator, there is a danger of losing an otherwise acceptable relation. On the other hand, this will ensure an even closer relationship between the dictionary and the resulting wordnet, which in turn ensures that the wordnet should reflect the represented language to a high extent - given that the dictionary does so.

3.4 Measuring Average Annotation Intervals

This part of the evaluation gives an approximate measure of the time spent post-processing the relations generated from Dict2WN. A timestamp representing the time of which an action was performed was stored for every action performed by a participant. When analyzing the results of the post-processing, a list of intervals (the amount of milliseconds between every timestamp) was compiled for every participant. Intervals larger than 5 minutes were omitted in order to prevent lunch/coffee breaks and similar events from affecting the statistics. The minimum interval, the maximum interval, the mean and the standard deviation was calculated for every participant as well as the combined lists of all participants. The results are shown in Table 7.12.

Annotator	Min	Max	Mean	StdDev
1	4.65	215.63	16.62	19.52
2	0.11	151.75	15.85	15.95
3	1.50	204.90	20.29	19.32

Table 7.12: Annotation intervals for the post-processing step.

The results for each annotator indicate that the time spent on each relation varies highly. Even so, given the mean and the standard deviation for each annotator, one can make a general assumption that the average time spent making a decision for a relation will be 30-90 seconds, increasing to 2-4 minutes when a difficult relation is encountered.

3.5 Concluding Remarks

It is difficult to make any strong claims about the output of DICT2WN based on the results of the annotation study. We could perhaps make some general conclusion based on the annotators individual results, but then we would ignore the low agreement rate. As seen in the kappa

measures for individual actions, the most reliable measure is probably the **Invalidate** actions with a 31% ratio, which gives us a very rough impression of the error rate of DICT2WN.

The agreement increases in the 2-annotator set; this could either be interpreted as an indication of bias, or as an indication of a tendency for semantic relations to mean different things for different scientific fields. This should be investigated further in a larger study.

The annotation intervals seem to indicate that the post-processing job is feasible in terms of the amount of manual labor it requires.

Chapter 8

Evaluation of Dict2WN

A final evaluation of the process is presented in this chapter, making use of the evaluated results from the manual post-processing step. A detailed study of the annotation itself is presented and gives some insights into the difficulties encountered. Section 3.4 investigates the time spent on the post-processing work and gives a rough prediction for a large-scale post-processing step.

1 Finding the recall of a semantic network

Recall is defined as $\frac{TP}{TP+FN}$, but finding the true number of false negatives in these types of graphs depend on a number of things. The central problem of figuring out the false negatives in semantic networks must be considered to rely on an open world assumption. The open world assumption is the assumption that the truth-value of a statement is independent of whether or not it is known by any single observer or agent to be true. In other words, for any statement about a concept that is not explicitly represented in the graph, it is either a possibility that the statement is false, or that the missing relation is a false negative; but the missing relation is not a sufficient condition to make such conclusions. A wordnet is supposed to model knowledge about concepts in a given language, and given the complexity and size of any natural language, any claim of a wordnet being complete with regards to both concept and relation coverage should be met with suspicion.

Because of this, it is difficult to give a definitive measure of the completeness of a wordnet or to get an exact number of false negatives; trying to decide whether a missing relation should be labeled as a false negative or simply as unknown is not always trivial.

2 Transducer Evaluation

To attempt an evaluation of the individual transducers, frequency lists showing how often the different transducers were responsible for a certain action with a certain agreement were compiled. By interpreting disambiguations as true positives and invalidations as false positives, we can attain measures of precision using the formula $P = \frac{TP}{TP+FP}$. Furthermore, for the 3-annotator data set, we can get two sets of precision rates by either considering only the fully agreed disambiguations and invalidations, or the combined number of fully and partially agreed disambiguations and invalidations.

The results are shown in Table 8.1. The **PAgr** column shows the precision rates based on fully agreed decisions. The **PMaj** shows the precision rates based on both fully and partially agreed decisions.

3-ANNOTATOR								
Id	**DAgr**	**DMaj**	**TP**	**IAgr**	**IMaj**	**FP**	**PAgr**	**PMaj**
1	23	15	38	2	1	3	0.92	0.927
2	5	5	10	1	2	3	0.833	0.769
2.1	1	2	3	4	4	8	0.2	0.273
2.1.1	8	7	15	0	14	14	1.0	0.517
2.2	21	15	36	3	7	10	0.875	0.783
2.2.1	39	29	68	11	14	25	0.78	0.731
2.3	17	12	29	4	10	14	0.81	0.674
2.3.1	27	27	54	10	0	10	0.73	0.844
3	5	1	6	0	0	0	1.0	1.0
3.1	36	41	77	7	16	23	0.837	0.77
2-ANNOTATOR								
Id	**DAgr**	**DMaj**	**TP**	**IAgr**	**IMaj**	**FP**	**Pagr**	**PMaj**
1	95	N/A	95	19	N/A	19	0.833	N/A
2	26	N/A	26	5	N/A	5	0.839	N/A
2.1	3	N/A	3	1	N/A	1	0.75	N/A
2.1.1	26	N/A	26	6	N/A	6	0.813	N/A
2.2	60	N/A	60	12	N/A	12	0.833	N/A
2.2.1	97	N/A	97	20	N/A	20	0.829	N/A
2.3	44	N/A	44	6	N/A	6	0.88	N/A
2.3.1	77	N/A	77	19	N/A	19	0.802	N/A
3	18	N/A	18	3	N/A	3	0.857	N/A
3.1	125	N/A	125	27	N/A	27	0.822	N/A

Table 8.1: Transducer precision rates. The columns show fully agreed disambiguations (**DAgr**), partially agreed disambiguations (**DMaj**) along with fully and partially agreed invalidations (**IAgr** and **IMaj**, respectively). **PAgr** gives the precision rate for the fully agreed decisions, treating **DAgr** as true positives and **IAgr** as false positives. **PMaj** shows the precision rate when the combined number of fully agreed and partially agreed decisions are interpreted as true positives and false positives (shown in **TP** and **FP** respectively). Both precision rate columns are calculated according to the formula $P = \frac{TP}{TP+FP}$.

There are a number of interesting observations to be done at this point. With a few exceptions, the majority of the transducers seem to perform fairly well.

Transducer 1 performs very well, with a precision at 0.927 in the 3-annotator data set and 0.833 in the 2-annotator dataset. This transducer is the first one created manually, with a trivial assignment of HAS_SYNONYM relations for all definientia consisting of single verbs or comma-separated verbs. From this we can infer that this transducer is mostly correct in its assigments, and that the dictionary is consistently defining verbs in terms of sets of synonyms whenever a comma-separated/single-verb definition is used.

Transducer 2 drops in precision in the 3-annotator data set, but keeps up with transducer 1 in the 2-annotator data set. This transducer was the second one that was created manually, being targeted towards definitions consisting of one verb followed by either a noun, adjective or a preposition. Instead of having hard-coded relations for each PoS tag encountered, it made use of operator words to infer the relations. Even though the precision drops, it is still high enough for the transducer to be considered successful. However, since no expansion has been done on this transducer, there is a small possibility that either some of the operator words should be investigated, or the consistency of the definentia catched by this transducer should be investigated.

Transducer 2.1 is the first 5-best expansion of transducer 2. The amount of post-processed relations are however so low that it is difficult to make any conclusions on this one.

Transducer 2.1.1 is the 5-best expansion of transducer 2.1, in other words it is part of the second iteration of the 5-best expansion of transducer 2. One very interesting phenomenon occurs in the invalidation data for this transducer: there are 14 false positives but none are fully agreed upon. One interpretation of this is that this transducer has hit upon a set of definitions that are inconsistently formed. Another interpretation is that it has found a set of definitions within which the words are difficult to decide the right relation upon.

Transducer 2.3.1 shows another interesting phenomenon: There are 10 false positives, all of which have full agreement. This indicates either that we should revise this transducer, or that we should revise some of the operator words. Since this is one of the transducers resulting from the second 5-best expansion of transducer 2, it might also imply some threshold for how deep one should go when expanding.

Transducer 3 was the last one to be manually created, targeting a single large PoS pattern class (VERB KOMMA VERB ADJ) not catched by any of the other transducers. It has perfect precision in the 3-annotator set and is the highest scoring transducer for the 2-annotator set, indicating both a high level of consistency in the dictionary definitions and good operator words. It is however also one of the transducers that catch the smallest number of definitions. Were it not for the additional data in the 2-annotator set, it would be difficult to make any valid conclusion since only 6 relations were post-processed by all three annotators.

Transducer 3.1 is the most liberal expansion experiment, performing a 10-best expansion on transducer 3. This transducer has a high coverage, but still an adequate precision.

One final observation to be made is the comparison of the results attained from the 3-annotator and 2-annotator data sets. The 2-annontator data set is approximately twice the size of the 3-annotator data set. There seems to be a convergence of the precision rates. The precision rates gained from the 3-annotator data set has a mean of 0.7288, with a standard deviation of 0.6229. The precision rates found in the 2-annotator set gives a mean of 0.8258, with a standard deviance of 0.3237. I am tempted to infer from this that the true precision rate lies closer to the values found in the 2-annotator data set, but some care has to be taken in this respect. Since there is no possibility of a majority decision for two annotators, one could assume that the precision will approach the mean of the 3-judge dataset if there are more judges.

2.1 Transducer Coverage

Coverage ratios list the amount of the dictionary data that is covered by the transducer when looking at either the definition id's, PoS pattern classes or lemmas. The definition id's are considered to be the most relevant in this case as it shows how many definientia are catched.

Id	CDef	ORel	Precision
1	0.0226	0.0306	0.927
2	0.0085	0.1810	0.769
2.1	0.0260	0.1284	0.273
2.1.1	0.0521	0.1398	0.517
2.2	0.0214	0.2139	0.783
2.2.1	0.0397	0.2177	0.731
2.3	0.0256	0.1548	0.674
2.3.1	0.0477	0.2150	0.844
3	0.0009	0.0088	1.0
3.1	0.0165	0.0235	0.77

Table 8.2: List over the initial transducers. asdf

A high coverage ratio for the definition id is the most general measurement and tells us directly how much of the dictionary data is covered. The coverage ratio for PoS patterns show us how much the regular expression for each transducer catches.

2.2 Transducer Overlap

Unless one is very restrictive when defining transducers and their expansion, they will overlap on a number of occasions in terms of which PoS pattern classes, definitions, relations and lemmas they capture. This can be measured by measuring the size of the intersection of the relations and captured PoS patterns. Table 8.4 gives an overview over the overlap ratios for PoS patterns, definition ID's and the actually generated relations.

The measure for relation overlap (seen in the column named **ORel**) tells us something about the level of redundancy of each transducer. A low score tells us that the transducer generates a high number of unique relations that the other transducers fail to generate.

2.3 Transducer Score Summary

A summary of the scores for all generated transducers is presented in Table 8.4

Id	OPoS	ODef	OLem	ORel
1	2.1066	21.0113	25.7387	3.0594
2	4.8349	20.0000	24.1529	18.1027
2.1	27.0160	53.2545	56.0606	12.8355
2.1.1	31.6819	52.7554	55.5978	13.9824
2.2	24.2982	46.8529	49.8911	21.3903
2.2.1	31.2151	52.8839	55.9195	21.7690
2.3	29.4247	52.9994	55.7480	15.4834
2.3.1	32.7908	53.4562	56.2291	21.4971
3	0.5195	0.5818	1.2743	0.8814
3.1	2.4550	6.7575	13.9937	2.3462

Table 8.3: Average overlap measures for all transducers. The columns show the transducer id (**Id**), PoS pattern overlap (**OPoS**, definition overlap (**Def**)), lemma overlap (**OLem**) and relations (**ORel**).

Id	CDef	CPoS	CLem	Overlap	Precision
1	0.0226	0.0001	0.2827	0.0306	0.927
2	0.0085	0.0001	0.1062	0.1810	0.769
2.1	0.0260	0.0009	0.3253	0.1284	0.273
2.1.1	0.0521	0.0021	0.6517	0.1398	0.517
2.2	0.0214	0.0008	0.2673	0.2139	0.783
2.2.1	0.0397	0.0021	0.4965	0.2177	0.731
2.3	0.0256	0.0009	0.3205	0.1548	0.674
2.3.1	0.0477	0.0020	0.5970	0.2150	0.844
3	0.0009	0.00003	0.0108	0.0088	1.0
3.1	0.0165	0.0006	0.2061	0.0235	0.77

Table 8.4: List over the initial transducers. asdf

These are used for the final evaluation measure. The **Precision** score is retrieved from the **PMaj** column for the 3-annotator dataset in Table 8.1. The reasoning behind choosing this precision rate set out of the three possible was made as follows:

There are three different sets of precision scores to choose from in Table 8.1 - **PAgr** and **PMaj** in the 3-annotator data set and **PAgr** in the 2-annotator data set. The choice of annotator data set fell upon the 3-annotator one because this is the data set that contains annotators with the most diverse background. Since the 2-annotator data set consists solely of people who have worked together previously on both NorNet and other lexicographical work, there is a higher danger of bias for this set. This reduces the choice of precision rates down to **PAgr** and **PMaj** in the 3-annotator set. **PMaj** was finally chosen as it was considered to be the set of precision rates that are the most descriptive.

The **CDef** column gives the definition Id coverage, which is taken from Table 8.4. The **ORel** gives the overlap ratio of generated relations, which is taken from Table 8.3.

Chapter 9

Conclusion and Further Work

A general method for semi-automatic generation of semantic networks based on information in a dictionary has been proposed in this thesis. A detailed study has been presented, investigating potential components for a framework that covers much of the process of extracting explanatory information from dictionary definitions in order to create a semantic network. Every step, from aquiring the source material to the final post processing stage have been investigated.

An implementation of the method has been developed and tested on Bokmålsordboka, a dictionary for Norwegian Bokmål. The resulting data has been the subject of a detailed analysis, both through automatic means and manual post-processing. A number of observations have been made during this development merits further investigation, and that could serve as the basis for further analysis and improvement of the method.

A thorough evaluation of the annotation process has been documented through the development of the post-processing application and an analysis of the results from three annotators. The results of this evaluation indicate some points of concern regarding agreement on verb relations, some of which might be generalized to remarks on annotation processes for semantic networks, or maybe annotation processes in general.

The use of PoS pattern classes and transducers cover much of what is needed for the application of Nygaard's method. By adding operator word functionality, we get hold of the lexical information that we need in order to handle changes in semantic content that is not accounted for by PoS tags alone. The transducer expansion lets us spend less time on discovering patterns in the source material manualle, thus we can to a certain extent claim to remedy some of the problems encountered in the DanNet pilot study.

A number of definientia belong to a class which we have called non-explanatory definentia. These were not considered in this study. Due to the high degree of consistency in such definientia, extracting information about domain and syntactic relations

The 1-to-n target ambiguity combined with the problems encountered in the process of automatic synset inference implies that a fully automatic method will be extremely difficult to develop. However, as we can see from the measurements done on the annotation intervals, the post-processing step is considered to be affordable in terms of the time spent on it.

The amount of generated relations is not close to covering all relations for all verb definitions extracted from BOB. Considering the number of post-processed relations done by all three annotators, and that this seems to be the best way we have of measuring the quality of the method as of now, creating/expanding transducers further would probably not add much information. The randomly selected subset constituting the 259 relations post-processed by all three annotators is enough to make some general remarks about the small set

of transducers. A larger set of relations generated by a larger amount of transducers would increase the possibility of the post-processed relations being spread out too thinly across the different transducers, making it difficult to say something about each one.

The lack of a well-defined gold standard makes the evaluation of the method difficult. In addition, the lack of agreement seen in the annotation study implies that a well-defined gold standard might be difficult to attain altogether. We have seen in Chapter 8 that in the cases when human annotators agree fully upon a relation being valid, the precision rates of the transducers seem to be quite high. It is however difficult to generalize this because of the annotation study.

Once a large enough graph of semantic relations is post-processed and fully disambiguated, more sophisticated analyses can be done for the various components of the proposed method. There are many possible configurations for the different parts of DICT2WN, but a full exploration of these is difficult to do unless the generated semantic networks can be compared to one that is deemed correct. Since we are dealing with a kind of source material that can be considered to be fairly static, we can assume that claims made based on a comparison of generated semantic networks with fully post-processed material will increase in strengh as the size of the post-processed material increases.

The set of relations that can be extracted by an analysis of explanatory parts of definitions has been extended with the proposed method. There are however a number of relations that are not considered. Antonomy for verbs would probably be a likely candidate, as well as the various subtypes of the INVOLVED relation.

A number of INVOLVED relations were extracted from the targeted verb definentia. No further specification of this relation was attempted. One could either assign the labor of specification to the manual post-processing stage, or try to find more sophisticated ways of distinguishing between the various possible types of INVOLVED relations by employing increased operator word functionality. It is also possible that more nuances could be discovered by making use of the fine-grained PoS tags.

The choice of transducers fell upon sequential finite-state transducers, as explained in . There was no ambiguity to deal with when applying these to the patterns devised for any transducer in this study, but no stronger claim can be made until a larger study is done on more complex patterns.

The operator word functionality seems to work adequately, but there are still a lot of things to investigate. First of all, the frequency list made during this master's thesis is probably not of the best. One should be able to get a better overview by applying techniques for generating collocation statistics and n-gram frequency lists.

Operator words and arguments that are unchanged by the transformation result in a set of relations of the forms LHS OPER RHS and LHS ARG RHS . These are discarded in the graph generation process. To improve the set of operator words, these should be compiled into lists and investigated.

As of now, there is no explicit way of modeling operator word precedence, apart from the fact the the operator words to the right implicitly have a precedence over operator words to their left because of the right-to-left iteration over relation sequences. By augmenting the operator words with some more sophisicated precedence assignment, they might become even more useful.

Another important thing to consider is that the amount of time spent deciding upon what kinds of relation sequence transformations should be done for each operator word is justifiable only if the rules apply on a general basis. If this work has to be redone for every dictionary, the efficiency of this method will not be as impressive. My intuition is that much of what is done for the operator words can be reused for other dictionaries, but I cannot make a strong claim

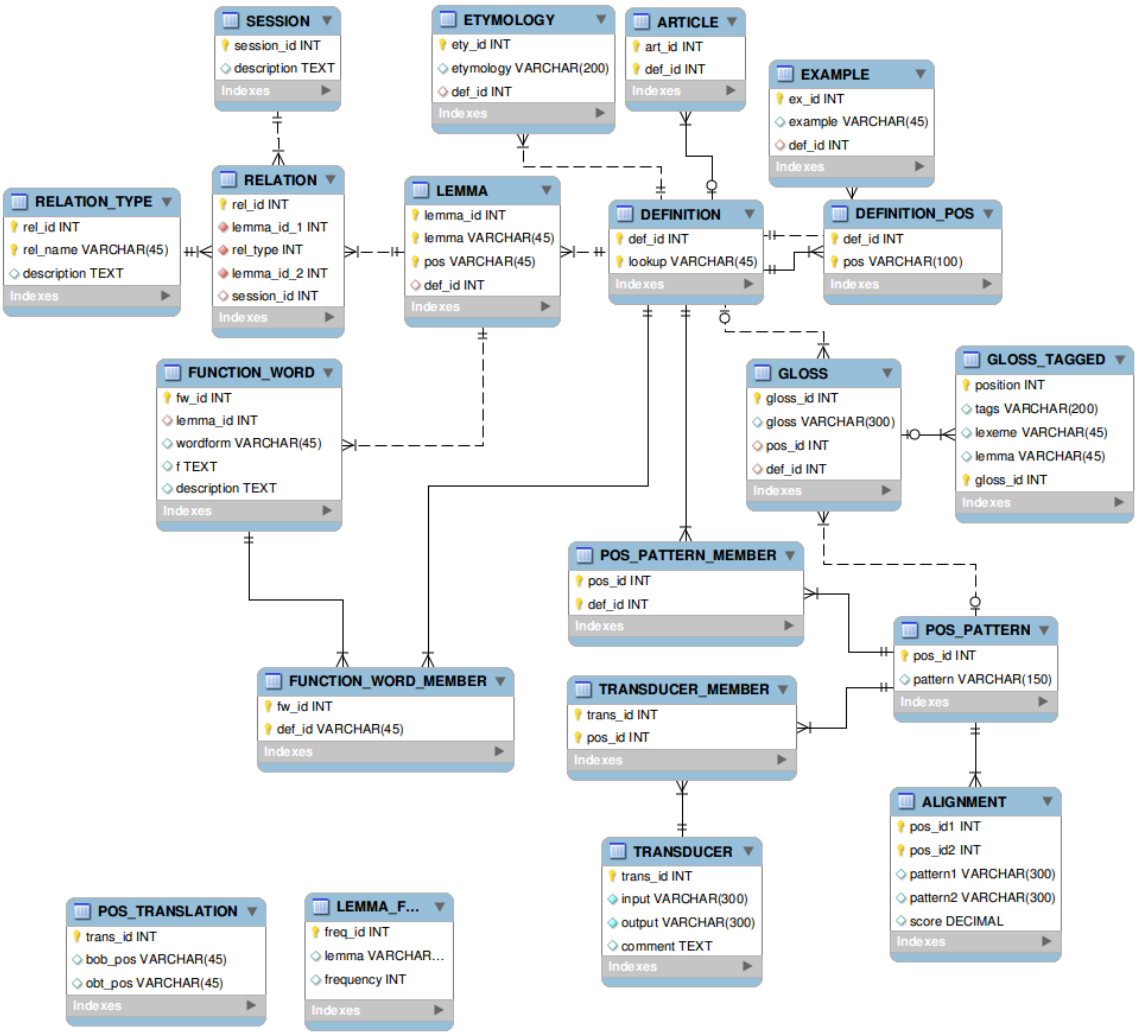
for this before more experiments are done.

There are probably a number of things that can be done to improve the alignment algorithm used for DICT2WN. Experiments with different parameters for the scoring function should be performed. The Smith-Waterman algorithm as defined and implemented in this thesis is also one of the simpler ways to perform a local sequence alignment. It would be very interesting to see if more advanced methods will improve the dynamicity of DICT2WN. Improving the scoring system by adding functionality for gap scores and using substitution matrices for PoS tags are examples of augmentations of the alignment method that could lead to a more versatile fuzzy matching of PoS pattern classes.

Appendix A

Dict2WN Program Description and Database

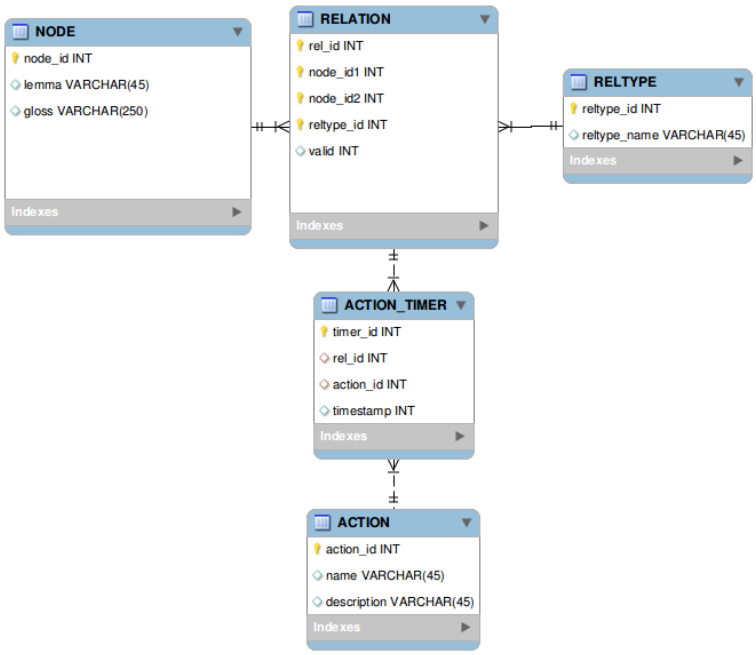
0.4 Database EER Schema



Appendix B

Dict2WNPP Program Description and Database

0.5 Database EER Schema



0.6 Manual for Dict2WNPP

User Manual for Dict2WNPP

1.7 Objectives, Motivation and General Remarks

This program is part of the post-processing/evaulation step for the method I am working on in my master thesis. It is designed with two objectives in mind: to identify invalid relations (i.e. relations that are outright wrong) and to disambiguate correct relations (i.e. finding one or more correct senses for a valid relation). This is a very specific process and does not include the possibility to add non-existing relations or to change relation types. For the purpose of

evaluating the generated data in my thesis, I consider the functionality in this program to be sufficient.

Every participant will work on the same sequence of generated relations, disambiguating, invalidating or skipping them as one sees fit (detailed instructions for this is in subsection [subsubappendix][9][21474836472,1]1.9). The results from this will be analyzed in my thesis. It will serve as the foundation for the evaluation of my method, both to find ways to refine it and to get an approximate measure of its quality. The data from the results will not be tied to any specific participant, but every one who choose to help me with this will have my eternal gratitude and an honorable mention in my thesis.

The user interface might seem a little stringent - it does not give you much choice as to which relations you can work with. This is done on purpose in order to try to get all participants to process as many of the same relations as possible. By doing it this way, I get the best possible data for an agreement measure. You do have the option of skipping a relation. I encourage everyone to try to make a decision for as many relations as possible. However, if many participants skip many of the same relations some interesting remarks might be made about a general uncertainty for certain types of verbs. In other words, even skipped relations is useful data for me.

> **Example:** Say we have four participants, and each participant has processed 300, 500, 800 and 1200 relations respectively, By "forcing" everyone to go through the same sequence of relations, I have the possibility to do a thorough comparison of the participants choices for the first 300 relations. The next 200 relations (301 - 500) will be a comparison of three participants, while the next 300 (501 - 800) will be a comparison of two paricipants. The remaining results (801 - 1200) will not be used for any agreement analysis but will still be useful.

There is no lower limit on the amount of relations to process, although I of course hope that this work will be interesting enough to get a nice chunk of results from each participant. There are quite a lot of relations and I don't expect anyone to go through all of them. I will go through as many relations as I can, but unbiased results from people not directly involved in my thesis is of incredible value to me. I have to assume that I personally have some bias, and that this bias might be subconsciously reflected in the choices I make for the relations (I of course want this method to be a good one).

I have tried to make the user interface as streamlined and straightforward as possible, but I appreciate any kind of feedback on the user experience as this program might serve as the foundation for a more advanced tool for NorNet in the future.

1.8 Download and Installation

Dict2WNPP can be found at the webpage `http://folk.uio.no/runelk/dict2wnpp`. The program is self-contained and does not rely on anything except Java 6. If you're in doubt as to which java version you have you can go to `http://www.java.com/en/download/installed.jsp` and click *Verify Java Version* to see what version you have on your computer. As long as *Your Java version* is shown as version 6 or more you should be ok, otherwise follow the instructions on the webpage to install a newer version. Given an adequate Java version, the installation process should be fairly simple:

1. Go to the webpage mentioned above.

2. Click on the URL on the webpage. You will be asked to open or save a file called *dict2wnpp.zip*. This is an archive containing the program. Most computers should know how to handle this type of files automatically. Either choose to open it right away, or download it to some appropriate place and open the file there. Either way, a program will open the zip file and display a folder *dict2wnpp*. This folder contains the program.

3. Put this folder in an appropriate location (your desktop or another place you can find easily). Inside of this folder you will find a couple of files that will run the program, depending on the operation system you are using:

   Windows Double-click **dict2wnpp.bat**.

   Linux Either double-click **dict2wnpp.sh** and choose *run*, or run **dict2wnpp.sh** from the command-line with the command '**sh dict2wnpp.sh**'.

   Mac OSX Same as Linux.

1.9 The User Interface

A window looking like the one below should appear shortly after starting the program. If nothing happens, or if you get some kind of error message, contact me at runelk@ifi.uio.no.

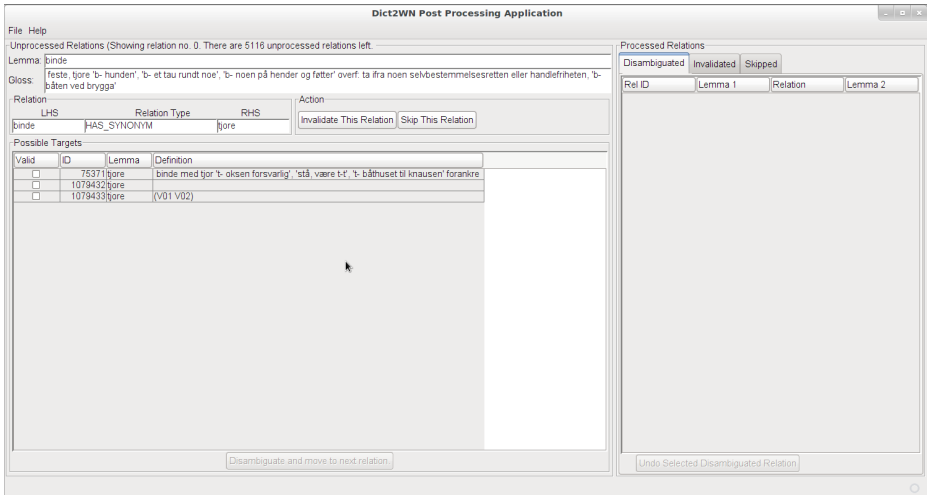


Figure B.1: Full screenshot of the Dict2WNPP user interface

This is the Dict2WN Post Processing application. The user interface is divided into two main areas: *Unprocessed Relations* and *Processed Relations*. A description of these two areas follow.

> **Note:** For the remainder of this manual, a relation will be defined as a lemma on the left hand side of a relation type (written as LHS), the relation type (e.g. HAS_SYNONYM), and a lemma on the right hand side of the relation (written as RHS).

Unprocessed Relations

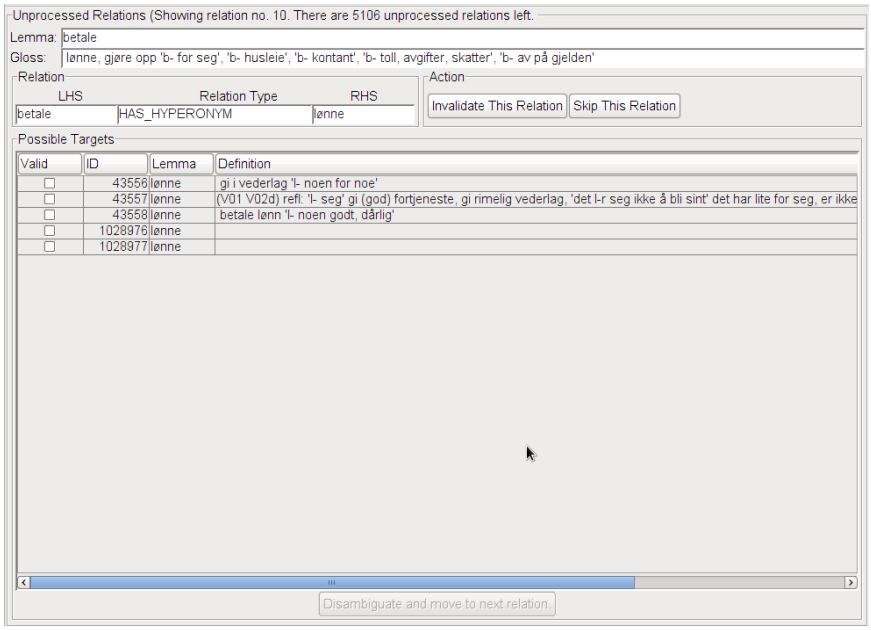


Figure B.2: Screenshot of the area for unprocessed relations

This area of the user interface presents the relations one by one. The upper part shows the lemma that occupies the LHS of the relation, along with its gloss. Below this is the actual relation (marked as *LHS*, *Relation Type* and *RHS*) along with two buttons for either invalidating or skipping the current relation. Below this is a list of possible targets for the RHS of the relation. The targets are equivalent with senses, corresponding to the various definitions from articles in Bokmålsordboka. You can also check the definitions at `http://www.nob-ordbok.uio.no/perl/ordbok.cgi?OPP=&bokmaal=+&ordbok=bokmaal`. You will see that the definitions are mostly similar to the definitions that show up in the user interface. Some senses have empty definition fields. These are included for the sake of completeness elsewhere in my method and you can disregard them when you decide what to do with a relation.

Each possible target is presented with its ID, lemma and definition. The leftmost column (*Valid*) contains one checkbox for each possible target. If you decide that the current relation is a valid one, you can select one or more targets by clicking the corresponding checkboxes. This will activate the *Disambiguate and move to next relation* button at the bottom, enabling you to store the choices you have made.

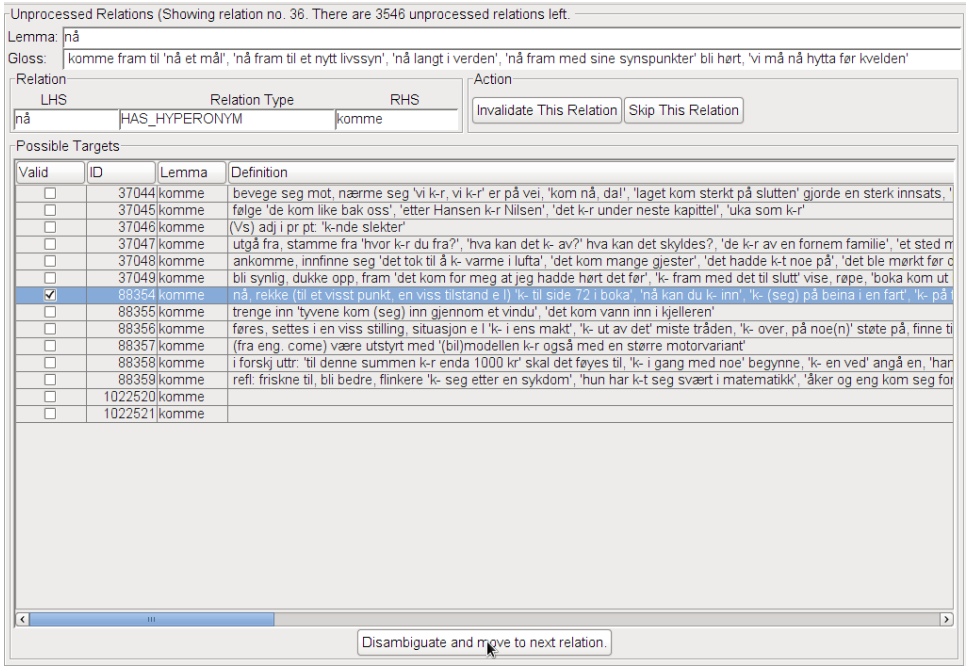


Figure B.3: Screenshot of a relation about to be disambiguated.

In other words, you have three choices for every relation: Disambiguate, Invalidate or Skip. Upon doing one of these three actions, the relation will be processed accordingly, the *Processed Relation* section will be updated reflecting the choice you just made, and the next relation will be presented to you. You are encouraged to either disambiguate or invalidate the relations. You *can* choose to skip a relation (at a later point, you can go to the list of processed relations as described in the next section and retrieve it), but this should be a last resort. If the relation is wrong, invalidate it. If there is absolutely **no** way to decide if the relation in question is a valid one or not, skip it and preferably review it later.

One important thing to remember here is that there are multiple definitions (senses) for most lemmas. This means that one must take care to decide if the relation is valid *given the gloss shown for the LHS*. You will find that the same LHS will appear many times, but its gloss will change.

Processed Relations

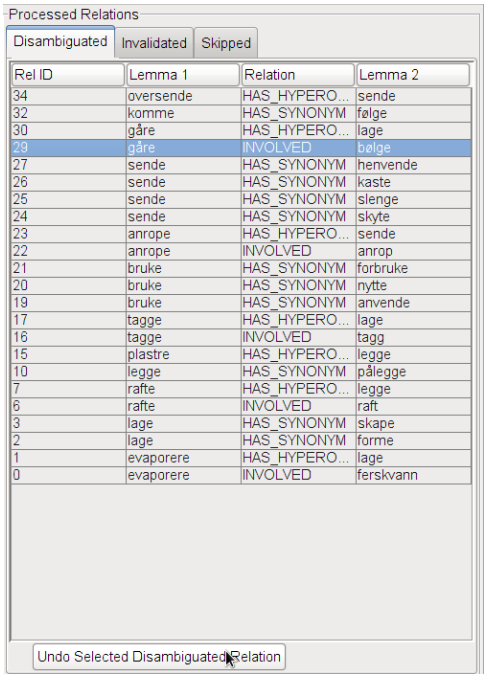


Figure B.4: Screenshot of a processed relation about to be undone

This part of the user interface presents an overview of the relations you have gone through so far. There are three different lists that correspond to the three types of actions you can perform in the *Unprocessed Relations* section - *Disambiguated*, *Invalidated* and *Skipped*. Every relation you process will be added to the appropriate list, ordered in such a way that the last processed relation will be at the top. You can switch between the lists by clicking on the appropriate flag near the top.

If you want to undo a previously processed relation, you can select the desired relation by clicking on it. This will activate a button near the bottom; by clicking this button the selected relation will be added back to the list of unprocessed relations. The *Unprocessed Relations* area will then update, enabling you to make a new choice for the relation you chose to undo.

1.10 Relation Overview

The following list is a description of the various relations you will encounter. The conditions for determining if a relation is valid is presented next to the name of the relation type. This list, along with the gloss/definitions for the lemmas are the guidelines for the decisions you make for each relation.

HAS_HYPERONYM True if the lemma on the left-hand side can be seen as a specialized form of the lemma on the right-hand side (e.g. *produsere* HAS_HYPERONYM *skape*).

HAS_SYNONYM True if the lemma on the left-hand side can be substituted for the lemma on the right-hand side without changing the meaning of a sentence in some context (e.g. *spise* HAS_SYNONYM *ete*).

CAUSES True if the lemma on the left-hand side is a sufficient cause for the lemma on the right-hand side (e.g. *drepe* CAUSES *dø*).

INVOLVED This is a supertype of a set of relations, and it is true if the RHS is that which is directly involved in the LHS. More specific descriptions for each subtype of this relation follow below. Note that none of the subtypes will occur in the material you are given. At the moment only INVOLVED occurs as I need to isolate the valid relations from the invalid relations and look for reoccurring patterns for the subtypes at a later point. Basically, if one or more of the criteria for the subtypes hold, the supertype should be considered valid.

- **INVOLVED_AGENT** True if the RHS is the cause or initiator (i.e. grammatical agent) of the left hand side. (e.g. *undervise* INVOLVED_AGENT *lærer*)
- **INVOLVED_PATIENT** True if the RHS is not an agent but directly involved with or affected by the left hand side (e.g. *undervise* INVOLVED_PATIENT *student*)
- **INVOLVED_INSTRUMENT** True if the RHS is an object used in some way when the LHS is performed (e.g. *male* INVOLVED_INSTRUMENT *pensel*)
- **INVOLVED_LOCATION** (e.g. *undervise* INVOLVED_LOCATION *skole*)
- **INVOLVED_RESULT** True if the RHS can be seen as something that is the result of the LHS (e.g. *fryse* INVOLVED_RESULT *is*)
- **INVOLVED_MANNER** Is true if the RHS says something about the manner of which the LHS is performed (e.g. *skrike* INVOLVED_MANNER *høy*)

ENTAILS A relaxed form of logical entailment. True if the RHS is a likely criteria for doing what is described by the LHS (e.g. *snorke* ENTAILS *sove*).

Some Remarks About Difficult Decisions

It is not always easy to determine if a relation is valid or not. Some things worth remembering are presented in this section. In general, try not to think about other possible relations between the lemmas presented to you. As stated in the beginning of this manual, this program does not allow for editing of missing relations.

The Synonymy/Hyperonymy Dilemma The distinction between synonymy and hyperonymy for verbs are somewhat fuzzy. The method I am developing tends to prefer synonym relations over hyperonym relations as synonym relations usually seem to conform more to the intuition for many verbs.

You will however encounter many relations where the only difference is the relation type (e.g. *slå* HAS_SYNONYM *banke* and *slå* HAS_HYPERONYM *banke*). This is because my method combines several rounds of generating relations where some rules lead to one relation type, and other rules lead to another. This reflects the synonymy/hyperonymy dilemma to a certain extent. I do some automatic decisions that reduce the number of these dilemmas, but there are still a lot of them left.

I don't want to say much about how you should decide upon which relation type is the most appropriate one. Just remember to treat every relation on its own. The results are valuable no matter what; either for the assessment the quality of the method (if participants tend to agree), or to the discussion on the synonymy/hyperonymy dilemma (if participants tend to disagree). In other words: make the choice that feels right at the time. Don't worry if you feel

that you might have been inconsistent from one processing session to another - this might also say something about the difficulty of assigning such relations consistently to verbs.

Involved As a supertype for a set of relations, this relation needs to be treated by me in a slightly different way than the other relations. The identification of valid INVOLVED relations will contribute to the discussion of whether or not my method can be improved to the point where it classifies the various relation subtypes to a satisfactory degree of accuracy. Because of this, there are no subtypes of INVOLVED in this dataset - as described in the list of relations, consider INVOLVED relations valid if one of their subtypes are valid.

1.11 Exporting The Data

When you are fed up with processing relations, it is time to send the results to me. To do this, open the *File* menu and choose *Export Data*. A file chooser dialog will pop up, prompting you to select a file name and location for the exported data. A default filename (*d2wnpp-export*) is supplied, but you can use whatever you feel like. Upon pressing *OK*, a .zip file (e.g. *d2wnpp-export.zip*) will be saved to the location you specified, containing all the data I need from you. All you have to do now is to send me a mail with this zip file attached and I'll respond happily.

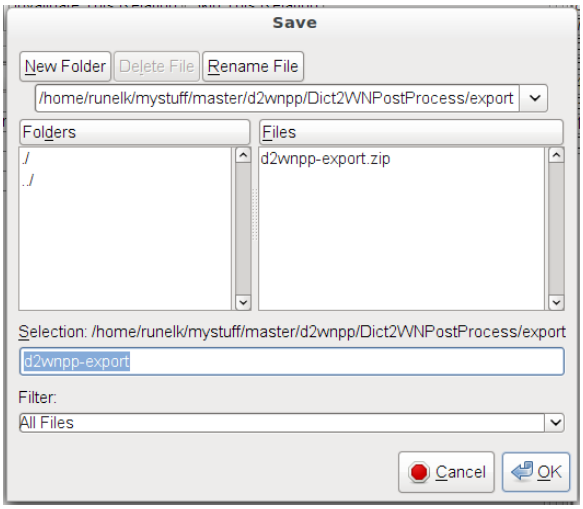


Figure B.5: Screenshot of the export dialog popping up when you choose *Export Data* from the *File* menu.

There is no need to save anything before you send the final result to me. All decisions you make with the user interface is stored as you go along. This means that you can start and stop the program without worrying if your work has been registered or not.

However, should you at any point download the program from the webpage and put it in the same destination as the last time, you **will overwrite all data**, losing any work you have done. You probably won't get into this situation, but if you're in doubt or have to reinstall the program for some reason, don't hesitate to send me a mail.

Thank you, and have fun!

Appendix C

Detailed Operator Word Data

Table C.1: Complete list of operator words defined for Dict2WN. Relation types abbreviated for the sake of readability.

[*I*	[$_{SELF}$ SUBST → NIL , SYMB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*at*	[$_{SELF}$ ADJ → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*av*	[$_{SELF}$ PREP → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ SUBST → INV]]
[*bevege*	[$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*bli*	[$_{SELF}$ VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → CAU , SUBST → NIL , VERB → NIL]]
[*bort*	[$_{SELF}$ ADJ → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*bringe*	[$_{SELF}$ SUBST → NIL , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ SUBST → INV]]
[*bruke*	[$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ SUBST → INV]]
[*danne*	[$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*drive*	[$_{SELF}$ SUBST → NIL , VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*e.l.*	[$_{SELF}$ FORK → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*el.*	[$_{SELF}$ FORK → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*en*	[$_{SELF}$ TALL → NIL , ADJ → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*etter*	[$_{SELF}$ PREP → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*falle*	[$_{SELF}$ SUBST → NIL , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*fast*	[$_{SELF}$ ADJ → ENT] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*for*	[$_{SELF}$ PREP → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*fra*	[$_{SELF}$ PREP → NIL] [$_{LEFT}$ SUBST → INV , VERB → HYP] [$_{RIGHT}$NIL]]
[*fram*	[$_{SELF}$ ADJ → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*få*	[$_{SELF}$ ADJ → ENT , VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*føre*	[$_{SELF}$ SUBST → NIL , ADJ → ENT , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*gi*	[$_{SELF}$ VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ SUBST → ENT]]
[*gjøre*	[$_{SELF}$ VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → CAU , SUBST → INV]]
[*gå*	[$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*ha*	[$_{SELF}$ SYMB → NIL , VERB → NIL , INTERJ → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → INV]]
[*holde*	[$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → INV]]
[*i*	[$_{SELF}$ PREP → NIL] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ ADJ → ENT , SUBST → INV]]
[*ikke*	[$_{SELF}$ SUBST → NIL , ADJ → NIL] [$_{LEFT}$ SUBST → INV] [$_{RIGHT}$ ADJ → ANT , VERB → ANT]]
[*inn*	[$_{SELF}$ ADJ → NIL] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ SUBST → INV]]

[*jf.* [$_{SELF}$ FORK → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → INV , VERB → SYN]]
[*komme* [$_{SELF}$ SUBST → NIL , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → ENT]]
[*l* [$_{SELF}$ SYMB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*la* [$_{SELF}$ SUBST → NIL , VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → INV , VERB → HYP]]
[*lage* [$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , ADV → CAU , SUBST → INV]]
[*legge* [$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → INV]]
[*liten* [$_{SELF}$ ADJ → ENT] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*lyd* [$_{SELF}$ SUBST → INV] [$_{LEFT}$ ADJ → ENT] [$_{RIGHT}$ ADJ → ENT , SUBJ → INV]]
[*med* [$_{SELF}$ SUBST → NIL , PREP → NIL] [$_{LEFT}$ ADJ → ENT , VERB → HYP] [$_{RIGHT}$ ADJ → ENT , SUBST → INV]]
[*mot* [$_{SELF}$ SUBST → NIL , PREP → NIL] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ SUBST → INV]]
[*mye* [$_{SELF}$ ADJ → ENT] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*ned* [$_{SELF}$ ADJ → NIL] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$NIL]]
[*ny* [$_{SELF}$ SUBST → NIL , ADJ → ENT] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*o* [$_{SELF}$] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*om* [$_{SELF}$ SUBST → NIL , PREP → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → DOMAIN]]
[*opp* [$_{SELF}$ ADJ → NIL] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ SUBST → INV]]
[*ordne* [$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*over* [$_{SELF}$ PREP → ENT] [$_{LEFT}$ SUBST → INV] [$_{RIGHT}$ SUBST → INV]]
[*pr.* [$_{SELF}$ FORK → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*på* [$_{SELF}$ PREP → NIL] [$_{LEFT}$ SUBST → INV , VERB → HYP] [$_{RIGHT}$ SUBST → INV , ADJ → ENT]]
[*sammen* [$_{SELF}$ ADJ → ENT] [$_{LEFT}$ SUBST → INV , VERB → HYP] [$_{RIGHT}$ ADJ → ENT , SUBST → INV]]
[*se* [$_{SELF}$ VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*sende* [$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ SUBST → INV]]
[*sette* [$_{SELF}$ VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ SUBST → INV]]
[*si* [$_{SELF}$ SUBST → NIL , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*sin* [$_{SELF}$ FORK → NIL] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ SUBST → INV]]
[*skille* [$_{SELF}$ SUBST → INV , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*skjære* [$_{SELF}$ SUBST → NIL , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*slå* [$_{SELF}$ SUBST → NIL , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*snakke* [$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT]]
[*sterk* [$_{SELF}$ ADJ → ENT] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ SUBST → INV]]
[*stille* [$_{SELF}$ SUBST → NIL , VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]

[*stoff* [$_{SELF}$ SUBST → INV] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*stor* [$_{SELF}$ ADJ → ENT] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*støte* [$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*særlig* [$_{SELF}$ ADJ → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*ta* [$_{SELF}$ VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ SUBST → INV]]
[*til* [$_{SELF}$ PREP → NIL] [$_{LEFT}$ SUBST → INV , VERB → HYP] [$_{RIGHT}$ ADJ → ENT , SUBST → CAU]]
[*tilbake* [$_{SELF}$ ADJ → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*under* [$_{SELF}$ SUBST → NIL , PREP → ENT] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ SUBST → INV]]
[*ut* [$_{SELF}$ ADJ → NIL] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ SUBST → INV]]
[*utføre* [$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → INV]]
[*vann* [$_{SELF}$ ADJ → NIL , SUBST → INV] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ SUBST → INV]]
[*ved* [$_{SELF}$ SUBST → NIL , PREP → NIL] [$_{LEFT}$ VERB → HYP] [$_{RIGHT}$ ADJ → ENT , VERB → HYP , SUBST → ENT]]
[*vise* [$_{SELF}$ SUBST → INV , VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → ENT]]
[*viss* [$_{SELF}$ ADJ → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ SUBST → INV]]
[*være* [$_{SELF}$ SUBST → NIL , VERB → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$ ADJ → ENT , SUBST → INV]]
[*å* [$_{SELF}$ SUBST → NIL , INTERJ → NIL , PREFIX → NIL] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]
[*ødelegge* [$_{SELF}$ VERB → HYP] [$_{LEFT}$NIL] [$_{RIGHT}$NIL]]

Appendix D

Detailed Transducer Data

PoS Patterns Captured by the Initial Transducers

Transducer 1							
Input	VERB	KOMMA	VERB	KOMMA	VERB	KOMMA	VERB
Output	SYN	NIL	SYN	NIL	SYN	NIL	SYN
Input	VERB	KOMMA	VERB	KOMMA	VERB		
Output	SYN	NIL	SYN	NIL	SYN		
Input	VERB	KOMMA	VERB				
Output	SYN	NIL	SYN				
Input	VERB						
Output	SYN						
Transducer 2							
Input	VERB	PREP					
Output	OPER	OPER					
Input	VERB	SUBST					
Output	OPER	ARG					
Input	VERB	ADJ					
Output	OPER	ARG					
Transducer 4							
Input	VERB	KOMMA	VERB	ADJ			
Output	HYP	NIL	OPER	ARG			

Table D.1: List of PoS Pattern classes captured by initial transducers. Some relation types are abbreviated for the sake of readability. The abbreviations are HAS_SYNONYM $\rightarrow$ SYN ; HAS_HYPERONYM $\rightarrow$ HYP

PoS Patterns Captured by Transducer Expansions

The PoS pattern classes captured by the expanded transducers are listed in Table D.2, Table D.3, Table D.4 and Table D.5.

Table D.2: List of PoS Pattern classes captured by initial transducers. Some relation types are abbreviated for the sake of readability. The abbreviations are HAS_SYNONYM → SYN ; HAS_HYPERONYM → HYP

Transducer 2.1			
Input	VERB	SUBST	ADJ
Output	OPER	OPER	NIL
Input	VERB	UKJENT	PREP
Output	OPER	OPER	OPER
Input	VERB	UKJENT	SUBST
Output	OPER	OPER	NIL
Input	VERB	ADV	ADJ
Output	OPER	NIL	NIL
Input	VERB	ADV	PREP
Output	OPER	NIL	OPER
Input	VERB	SUBST	SUBST
Output	OPER	OPER	NIL
Input	VERB	ADV	SUBST
Output	OPER	NIL	NIL
Input	VERB	PREP	PRON
Output	OPER	OPER	NIL
Input	VERB	PREP	ADV
Output	OPER	OPER	NIL
Input	VERB	PREP	PREP
Output	OPER	OPER	OPER
Input	VERB	VERB	SUBST
Output	OPER	OPER	NIL
Input	VERB	SUBST	ADV
Output	OPER	OPER	NIL
Input	VERB	ADJ	ADV
Output	OPER	OPER	NIL
Input	VERB	PREP	ADJ
Output	OPER	OPER	NIL
Input	VERB	VERB	PREP
Output	OPER	OPER	OPER
Input	VERB	ADJ	PREP
Output	OPER	OPER	OPER
Input	VERB	PRON	
Output	OPER	NIL	
Input	VERB	ADJ	ADJ
Output	OPER	OPER	NIL

Continued on next page.

Table D.2: List of PoS Pattern classes captured by transducer 2.1. (Continued from last page)

Transducer 2.1			
Input	VERB	ADV	
Output	OPER	NIL	
Input	VERB	SUBST	PREP
Output	OPER	OPER	OPER
Input	VERB	ADJ	SUBST
Output	OPER	OPER	NIL
Input	VERB	PREP	
Output	OPER	OPER	
Input	VERB	PREP	SUBST
Output	OPER	OPER	NIL
Input	VERB	UKJENT	
Output	OPER	OPER	
Input	VERB	SUBST	
Output	OPER	OPER	
Input	VERB	VERB	
Output	OPER	OPER	
Input	VERB	ADJ	
Output	OPER	OPER	
Input	VERB		
Output	OPER		

Table D.3: List of PoS Pattern classes captured by transducer 2.2. Some relation types are abbreviated for the sake of readability. The abbreviations are HAS_SYNONYM $\rightarrow$ SYN ; HAS_HYPERONYM $\rightarrow$ HYP

Transducer 2.2			
Input	VERB	UKJENT	PREP
Output	OPER	ARG	NIL
Input	VERB	ADV	PREP
Output	OPER	ARG	NIL
Input	VERB	UKJENT	CLB
Output	OPER	ARG	NIL
Input	VERB	UKJENT	PARENTES-SLUTT
Output	OPER	ARG	NIL
Input	VERB	PREP	ADV
Output	OPER	NIL	NIL
Input	VERB	PREP	PREP
Output	OPER	NIL	NIL

Continued on next page.

Table D.3: List of PoS Pattern classes captured by transducer 2.2. (Continued from last page)

Transducer 2.2			
Input	VERB	PRON	PREP
Output	OPER	ARG	NIL
Input	VERB	CLB	
Output	OPER	NIL	
Input	VERB	SUBST	ADV
Output	OPER	ARG	NIL
Input	VERB	ADJ	ADV
Output	OPER	ARG	NIL
Input	VERB	PRON	ADV
Output	OPER	ARG	NIL
Input	VERB	PREP	CLB
Output	OPER	NIL	NIL
Input	VERB	ADJ	PREP
Output	OPER	ARG	NIL
Input	VERB	ADJ	CLB
Output	OPER	ARG	NIL
Input	VERB	PRON	
Output	OPER	ARG	
Input	VERB	SUBST	PARENTES-SLUTT
Output	OPER	ARG	NIL
Input	VERB	SUBST	CLB
Output	OPER	ARG	NIL
Input	VERB	ADV	
Output	OPER	ARG	
Input	VERB	SUBST	PREP
Output	OPER	ARG	NIL
Input	VERB	PREP	
Output	OPER	NIL	
Input	VERB	UKJENT	
Output	OPER	ARG	
Input	VERB	SUBST	
Output	OPER	ARG	
Input	VERB	ADJ	
Output	OPER	ARG	
Input	VERB		
Output	OPER		

Table D.4: List of PoS Pattern classes captured by transducer 2.3. Some relation types are abbreviated for the sake of readability. The abbreviations are HAS_SYNONYM → SYN ; HAS_HYPERONYM → HYP

Transducer 2.3			
Input	VERB	SUBST	ADJ
Output	OPER	NIL	ARG
Input	VERB	UKJENT	PREP
Output	OPER	ARG	NIL
Input	VERB	UKJENT	SUBST
Output	OPER	ARG	NIL
Input	VERB	ADV	ADJ
Output	OPER	ARG	ARG
Input	VERB	ADV	PREP
Output	OPER	ARG	NIL
Input	VERB	UKJENT	CLB
Output	OPER	ARG	NIL
Input	VERB	SUBST	SUBST
Output	OPER	NIL	NIL
Input	VERB	ADV	SUBST
Output	OPER	ARG	NIL
Input	VERB	PRON	ADJ
Output	OPER	ARG	ARG
Input	VERB	PREP	PREP
Output	OPER	ARG	NIL
Input	VERB	PRON	PREP
Output	OPER	ARG	NIL
Input	VERB	CLB	
Output	OPER	NIL	
Input	VERB	PRON	SUBST
Output	OPER	ARG	NIL
Input	VERB	PREP	ADJ
Output	OPER	ARG	ARG
Input	VERB	PREP	CLB
Output	OPER	ARG	NIL
Input	VERB	ADJ	PREP
Output	OPER	ARG	NIL
Input	VERB	ADJ	CLB
Output	OPER	ARG	NIL
Input	VERB	PRON	
Output	OPER	ARG	

Continued on next page.

Table D.4: List of PoS Pattern classes captured by transducer 2.3. (Continued from last page)

Transducer 2.3			
Input	VERB	ADJ	ADJ
Output	OPER	ARG	ARG
Input	VERB	SUBST	CLB
Output	OPER	NIL	NIL
Input	VERB	ADV	
Output	OPER	ARG	
Input	VERB	SUBST	PREP
Output	OPER	NIL	NIL
Input	VERB	ADJ	SUBST
Output	OPER	ARG	NIL
Input	VERB	PREP	
Output	OPER	NIL	
Input	VERB	PREP	SUBST
Output	OPER	ARG	NIL
Input	VERB	UKJENT	
Output	OPER	ARG	
Input	VERB	SUBST	
Output	OPER	NIL	
Input	VERB	ADJ	
Output	OPER	ARG	
Input	VERB		
Output	OPER		

Table D.5: List of PoS Pattern classes captured by transducer 3.1. Some relation types are abbreviated for the sake of readability. The abbreviations are HAS_SYNONYM $\rightarrow$ SYN ; HAS_HYPERONYM $\rightarrow$ HYP

Transducer 3.1					
Input	VERB	KOMMA	VERB	ADJ	ADJ
Output	HYP	NIL	OPER	ARG	ARG
Input	VERB	KOMMA	VERB	ADV	PREP
Output	HYP	NIL	OPER	ARG	NIL
Input	VERB	KOMMA	VERB	PREP	PREP
Output	HYP	NIL	OPER	NIL	NIL
Input	VERB	KOMMA	VERB	VERB	PREP
Output	HYP	NIL	OPER	ARG	NIL
Input	VERB	KOMMA	VERB	PREP	ADJ
Output	HYP	NIL	OPER	NIL	ARG

Continued on next page.

Table D.5: List of PoS Pattern classes captured by transducer 3.1. (Continued from last page)

Transducer 3.1					
Input	VERB	KOMMA	VERB	ADV	
Output	HYP	NIL	OPER	ARG	
Input	VERB	KOMMA	VERB	PRON	SUBST
Output	HYP	NIL	OPER	ARG	NIL
Input	VERB	KOMMA	VERB	ADJ	SUBST
Output	HYP	NIL	OPER	ARG	NIL
Input	VERB	KOMMA	VERB	PRON	
Output	HYP	NIL	OPER	ARG	
Input	VERB	KOMMA	VERB	VERB	
Output	HYP	NIL	OPER	ARG	
Input	VERB	KOMMA	VERB	PRON	ADJ
Output	HYP	NIL	OPER	ARG	ARG
Input	VERB	KOMMA	VERB	SUBST	
Output	HYP	NIL	OPER	ARG	
Input	VERB	KOMMA	VERB	SUBST	PREP
Output	HYP	NIL	OPER	ARG	NIL
Input	VERB	KOMMA	VERB	PRON	PREP
Output	HYP	NIL	OPER	ARG	NIL
Input	VERB	KOMMA	VERB	ADJ	PREP
Output	HYP	NIL	OPER	ARG	NIL
Input	VERB	KOMMA	VERB	ADJ	
Output	HYP	NIL	OPER	ARG	
Input	VERB	KOMMA	VERB	PREP	SUBST
Output	HYP	NIL	OPER	NIL	NIL
Input	VERB	KOMMA	VERB	PREP	
Output	HYP	NIL	OPER	NIL	
Input	VERB	KOMMA	VERB		
Output	HYP	NIL	OPER		

Appendix E

Detailed Data for the Post-Processing Evaluation

0.12 List of Post-processed Relations Grouped by Agreement

3 Judge Dataset

Table E.1: 3-Judge Dataset: List of relations accepted by all three judges, lemma level.

Lemma 1	Relation	Lemma 2
adherere	HAS_HYPERONYM	holde
anføre	HAS_HYPERONYM	nevne
anrope	INVOLVED	anrop
anta	HAS_SYNONYM	formode
antaste	ENTAILS	påtrengende
anvise	ENTAILS	ordre
artikulere	HAS_HYPERONYM	forme
avta	INVOLVED	kjøper
avtjene	HAS_HYPERONYM	utføre
bale	HAS_HYPERONYM	streve
behandle	HAS_SYNONYM	bearbeide
beta	HAS_SYNONYM	frata
bløtlegge	HAS_HYPERONYM	legge
eksponere	INVOLVED	lys
ekstemporere	HAS_HYPERONYM	lese
enes	CAUSES	enig
etikettere	INVOLVED	etikett
fare	HAS_SYNONYM	ferdes
fare	HAS_SYNONYM	reise
fastholde	HAS_HYPERONYM	holde
fatte	HAS_HYPERONYM	gripe
fikle	HAS_SYNONYM	plukke
fikle	HAS_SYNONYM	pusle
fikle	HAS_SYNONYM	tukle
filtrere	HAS_HYPERONYM	sile

Continued on next page

Table E.1: 3-Judge Dataset: List of relations accepted by all three judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
finne	HAS_SYNONYM	vurdere
fly	HAS_SYNONYM	flakse
fløte	INVOLVED	vann
forevise	HAS_HYPERONYM	vise
forta	HAS_HYPERONYM	forsvinne
fratre	HAS_HYPERONYM	slutte
fuske	CAUSES	dårlig
gjelde	HAS_HYPERONYM	angå
gjenbruke	HAS_HYPERONYM	bruke
helligholde	ENTAILS	hellig
henge	HAS_HYPERONYM	holde
henvise	HAS_HYPERONYM	vise
himle	INVOLVED	himling
hvitne	CAUSES	hvit
innbringe	HAS_HYPERONYM	bringe
inngi	HAS_HYPERONYM	sende
innskipe	HAS_HYPERONYM	bringe
instituere	HAS_HYPERONYM	stifte
instruere	ENTAILS	opplæring
instruere	HAS_HYPERONYM	veilede
intensivere	HAS_HYPERONYM	forsterke
kaue	HAS_SYNONYM	hauke
kaue	HAS_SYNONYM	lokke
klappe	HAS_HYPERONYM	legge
klargjøre	CAUSES	klar
klinge	HAS_SYNONYM	høres
klinge	HAS_SYNONYM	lyde
knuge	HAS_HYPERONYM	klemme
komprimere	HAS_HYPERONYM	presse
krikle	HAS_HYPERONYM	lage
krinse	HAS_SYNONYM	kretse
krysse	HAS_HYPERONYM	gå
kultivere	HAS_HYPERONYM	dyrke
kåle	HAS_HYPERONYM	ødelegge
landsforvise	HAS_HYPERONYM	vise
mane	HAS_SYNONYM	egge
modne	CAUSES	moden
more	HAS_SYNONYM	muntre
nedbygge	HAS_HYPERONYM	redusere
nyte	HAS_HYPERONYM	fortære
oppirre	HAS_HYPERONYM	ergre
oppta	HAS_SYNONYM	besette
organisere	HAS_HYPERONYM	ordne
orientere	HAS_HYPERONYM	opplyse

Continued on next page

Table E.1: 3-Judge Dataset: List of relations accepted by all three judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
oversende	HAS_HYPERONYM	sende
parfymere	CAUSES	velluktende
pigge	INVOLVED	pigg
plaske	HAS_SYNONYM	skvette
plaske	HAS_SYNONYM	skvalpe
plire	HAS_SYNONYM	myse
plissere	HAS_HYPERONYM	legge
puffe	HAS_SYNONYM	dampe
rafte	INVOLVED	raft
reduplisere	HAS_SYNONYM	fordoble
respektere	INVOLVED	aktelse
romme	HAS_SYNONYM	inneholde
røpe	CAUSES	kjent
sikle	INVOLVED	lyst
skjefte	INVOLVED	skaft
snike	HAS_SYNONYM	liste
snike	HAS_SYNONYM	lure
sone	INVOLVED	bot
sortere	HAS_HYPERONYM	inndele
strippe	INVOLVED	striptease
summe	HAS_HYPERONYM	surre
tagge	INVOLVED	tagg
tilsnakke	HAS_HYPERONYM	irettesette
trasse	ENTAILS	trassig
ugge	ENTAILS	redd
urbanisere	CAUSES	bymessig
utlyse	HAS_HYPERONYM	kunngjøre
vite	ENTAILS	sikker
vringle	ENTAILS	vrang
yte	HAS_HYPERONYM	gi
åpne	CAUSES	synlig

Accepted Relations by All 3 Judges (Full Agreement)

Table E.2: 3-Judge Dataset: List over relations accepted by two of three judges, lemma level.

Lemma 1	Relation	Lemma 2
adherere	ENTAILS	fast
adherere	HAS_HYPERONYM	henge
aksentuere	HAS_HYPERONYM	framheve
aksentuere	INVOLVED	vekt
anrope	HAS_HYPERONYM	sende
anta	HAS_HYPERONYM	formode

Continued on next page

Table E.2: 3-Judge Dataset: List over relations accepted by two of three judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
antaste	HAS_HYPERONYM	tilsnakke
avfette	CAUSES	fri
besette	INVOLVED	beslag
besette	HAS_SYNONYM	oppta
besette	HAS_HYPERONYM	oppta
dimittere	HAS_HYPERONYM	sende
divertere	HAS_SYNONYM	more
divertere	HAS_HYPERONYM	more
fare	HAS_SYNONYM	gå
fastholde	ENTAILS	fast
figurere	HAS_HYPERONYM	stå
finne	HAS_SYNONYM	få
finne	HAS_SYNONYM	oppnå
finne	HAS_SYNONYM	synes
finne	HAS_SYNONYM	vinne
flagge	ENTAILS	tegn
flanere	HAS_HYPERONYM	slentre
fly	HAS_SYNONYM	farte
fly	HAS_SYNONYM	springe
forakte	HAS_HYPERONYM	ringeakte
forfjamse	HAS_HYPERONYM	bringe
fuske	HAS_HYPERONYM	slurve
gåre	INVOLVED	bølge
gåre	HAS_HYPERONYM	lage
halse	HAS_HYPERONYM	fare
hevde	HAS_HYPERONYM	vinne
hoppe	ENTAILS	urolig
inngi	HAS_HYPERONYM	gi
inngi	HAS_HYPERONYM	levere
innløpe	HAS_HYPERONYM	komme
insistere	HAS_HYPERONYM	holde
kannelere	INVOLVED	kannelyre
kannelere	HAS_HYPERONYM	lage
kase	HAS_HYPERONYM	legge
kaue	HAS_SYNONYM	rope
klinge	HAS_HYPERONYM	lyde
klore	HAS_HYPERONYM	hogge
kontrollere	INVOLVED	herredømme
kopulere	HAS_HYPERONYM	føye
krølle	HAS_HYPERONYM	legge
kursivere	HAS_HYPERONYM	sette
lugne	HAS_HYPERONYM	roe
løse	HAS_HYPERONYM	ordne

Continued on next page

Table E.2: 3-Judge Dataset: List over relations accepted by two of three judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
more	HAS_SYNONYM	glede
more	HAS_SYNONYM	oppmuntre
more	HAS_SYNONYM	underholde
nå	HAS_HYPERONYM	komme
nå	HAS_HYPERONYM	komme
omgås	ENTAILS	selskapelig
oppnorske	HAS_HYPERONYM	fornorske
oppta	INVOLVED	beslag
permittere	ENTAILS	permisjon
plastre	HAS_HYPERONYM	legge
rafte	HAS_HYPERONYM	legge
reduplisere	HAS_SYNONYM	gjenta
renne	HAS_HYPERONYM	gli
rokere	HAS_HYPERONYM	utføre
romme	HAS_HYPERONYM	inneholde
ryke	HAS_HYPERONYM	tape
røpe	HAS_HYPERONYM	avsløre
sifrere	ENTAILS	siffer
skade	ENTAILS	uheldig
skeivle	HAS_HYPERONYM	bringe
skje	HAS_HYPERONYM	hende
skjene	HAS_HYPERONYM	fare
smashe	HAS_HYPERONYM	utføre
sprike	HAS_HYPERONYM	spile
sprike	HAS_HYPERONYM	vise
streke	HAS_HYPERONYM	lage
strippe	HAS_HYPERONYM	utføre
stryke	HAS_HYPERONYM	sløyfe
stryke	HAS_HYPERONYM	utgå
synkverve	HAS_HYPERONYM	forvirre
tagge	HAS_HYPERONYM	lage
terrassere	ENTAILS	form
tolke	ENTAILS	uttrykk
trimme	HAS_HYPERONYM	mosjonere
trinse	HAS_SYNONYM	rulle
trinse	HAS_SYNONYM	trille
vake	HAS_HYPERONYM	flyte
virke	INVOLVED	virkning

Accepted Relations for 2 of 3 Judges (Partial Agreement)

Table E.3: 3-Judge Dataset: List over relations invalidated by all three judges, lemma level.

Lemma 1	Relation	Lemma 2
aksentuere	HAS_HYPERONYM	legge
avdramatisere	CAUSES	liten
avsky	ENTAILS	sterk
beundre	ENTAILS	over
effektivisere	CAUSES	mye
finne	HAS_SYNONYM	kjenne
forurense	INVOLVED	ur
forutskikke	HAS_HYPERONYM	komme
intensivere	CAUSES	mye
kontrollere	ENTAILS	over
korse	HAS_HYPERONYM	legge
motorisere	HAS_HYPERONYM	legge
oppnorske	CAUSES	mye
oversende	ENTAILS	over
plaske	HAS_SYNONYM	slå
plastre	INVOLVED	plast
prioritere	ENTAILS	fortrinn
renne	HAS_HYPERONYM	helle
renne	HAS_HYPERONYM	helle
seigpine	ENTAILS	over
sjenere	HAS_HYPERONYM	være
sluke	HAS_HYPERONYM	legge
sluke	ENTAILS	under
spare	HAS_HYPERONYM	bruke
spraye	HAS_HYPERONYM	bruke
undres	ENTAILS	over
virke	HAS_HYPERONYM	ha

Invalidated Relations by All 3 Judges (Full Agreement)

Table E.4: 3-Judge Dataset: List over relations invalidated by the majority of three judges, lemma level.

Lemma 1	Relation	Lemma 2
besette	HAS_HYPERONYM	legge
besette	HAS_HYPERONYM	oppta
etse	HAS_SYNONYM	gravere
formalisere	HAS_HYPERONYM	bringe
framlegge	HAS_HYPERONYM	legge
glemme	HAS_HYPERONYM	legge
helligholde	HAS_HYPERONYM	holde
heroisere	HAS_HYPERONYM	betrakte
himle	HAS_HYPERONYM	legge

Continued on next page

Table E.4: 3-Judge Dataset: List over relations invalidated by the majority of three judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
kave	CAUSES	ubehjelpelig
knuge	ENTAILS	over
koke	ENTAILS	sterk
kunne	ENTAILS	trygg
lugne	HAS_HYPERONYM	bringe
nå	HAS_HYPERONYM	komme
nå	HAS_HYPERONYM	oppnå
oppmyke	CAUSES	liten
oppta	HAS_HYPERONYM	besette
oppta	HAS_HYPERONYM	besette
oppta	HAS_HYPERONYM	legge
overanstrenge	ENTAILS	sterk
overbelaste	ENTAILS	sterk
reduplisere	HAS_HYPERONYM	fordoble
renne	HAS_SYNONYM	helle
ryke	HAS_HYPERONYM	sende
samstave	HAS_HYPERONYM	komme
se	HAS_HYPERONYM	oppdage
servere	HAS_HYPERONYM	framføre
sjokkåpne	ENTAILS	sterk
skrinlegge	HAS_HYPERONYM	legge
snike	HAS_HYPERONYM	lure
snyte	HAS_HYPERONYM	pusse
telefonere	HAS_HYPERONYM	ringe
tilkjennegi	ENTAILS	uttrykk
underkaste	INVOLVED	gjenstand
underlegge	INVOLVED	gjenstand
unnsleppe	HAS_HYPERONYM	komme
vake	HAS_HYPERONYM	holde
velsigne	ENTAILS	over

Invalidated Relations by 2 of 3 Judges (Partial Agreement)

Table E.5: 3-Judge Dataset: List over relations fully disagreed by all three judges, lemma level.

Lemma 1	Relation	Lemma 2
kjørne	INVOLVED	merke
krusle	HAS_HYPERONYM	kreke
sælde	HAS_HYPERONYM	sikte
trinse	HAS_SYNONYM	tumle
ulke	HAS_HYPERONYM	gulpe
uteske	HAS_HYPERONYM	utfordre

Relations With No Agreement Between 3 Judges

2 Judge Dataset

Table E.6: 2-Judge Dataset: List of relations accepted by both judges, lemma level.

Lemma 1	Relation	Lemma 2
adherere	ENTAILS	fast
adherere	HAS_HYPERONYM	holde
anfalle	HAS_HYPERONYM	angripe
anfekte	HAS_HYPERONYM	bringe
anføre	HAS_HYPERONYM	nevne
anke	HAS_HYPERONYM	klage
anrope	INVOLVED	anrop
anta	HAS_SYNONYM	formode
antaste	ENTAILS	påtrengende
antenne	INVOLVED	fyr
anvise	ENTAILS	ordre
applisere	HAS_SYNONYM	anvende
applisere	HAS_HYPERONYM	anvende
artikulere	HAS_HYPERONYM	forme
avfette	CAUSES	fri
avgrense	INVOLVED	grense
avskipe	HAS_HYPERONYM	sende
avta	INVOLVED	kjøper
avtjene	HAS_HYPERONYM	utføre
avtrappe	HAS_HYPERONYM	minske
bale	HAS_HYPERONYM	streve
barbere	INVOLVED	skjegg
behandle	HAS_SYNONYM	bearbeide
bekrefte	HAS_SYNONYM	befeste
bekrefte	HAS_SYNONYM	styrke
bekrefte	HAS_SYNONYM	stadfeste
besette	HAS_HYPERONYM	oppta
bestille	HAS_SYNONYM	tinge
bestri	HAS_SYNONYM	betale
bestri	HAS_HYPERONYM	betale
bestri	HAS_SYNONYM	utrede
beta	HAS_SYNONYM	frata
betale	HAS_HYPERONYM	lønne
bevokte	INVOLVED	vakt
bite	INVOLVED	tann
bløtlegge	HAS_HYPERONYM	legge
bolte	HAS_HYPERONYM	feste
brodde	HAS_SYNONYM	brydde
brodde	HAS_SYNONYM	spire
brotsje	INVOLVED	brotsj

Continued on next page

Table E.6: 2-Judge Dataset: List of relations accepted by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
deise	HAS_SYNONYM	dumpe
deise	HAS_HYPERONYM	falle
dimittere	HAS_HYPERONYM	sende
drikke	HAS_HYPERONYM	nyte
drømme	INVOLVED	drøm
dyrke	HAS_HYPERONYM	ære
dølge	ENTAILS	hemmelig
dølge	HAS_SYNONYM	skjule
dølge	HAS_HYPERONYM	skjule
effektuere	HAS_HYPERONYM	utføre
egge	HAS_SYNONYM	inspirere
egge	HAS_SYNONYM	oppildne
egge	HAS_SYNONYM	utfordre
eksponere	INVOLVED	lys
ekstemporere	HAS_HYPERONYM	lese
enes	CAUSES	enig
eskapere	HAS_HYPERONYM	flykte
estimere	HAS_HYPERONYM	verdsette
etikettere	INVOLVED	etikett
evaporere	INVOLVED	ferskvann
fare	HAS_SYNONYM	ferdes
fare	HAS_SYNONYM	reise
fastholde	HAS_HYPERONYM	holde
fatte	HAS_HYPERONYM	gripe
fikle	HAS_SYNONYM	plukke
fikle	HAS_SYNONYM	pusle
fikle	HAS_SYNONYM	tukle
filtrere	HAS_HYPERONYM	sile
fininnstille	HAS_HYPERONYM	justere
finne	HAS_SYNONYM	få
finne	HAS_SYNONYM	oppnå
finne	HAS_SYNONYM	vurdere
finne	HAS_SYNONYM	vinne
flagge	ENTAILS	tegn
floragrafere	INVOLVED	floragram
floragrafere	HAS_HYPERONYM	sende
fly	HAS_SYNONYM	flakse
flæ	INVOLVED	flytholt
fløte	INVOLVED	vann
forevise	HAS_HYPERONYM	vise
forfølge	HAS_HYPERONYM	bringe
forgripe	HAS_HYPERONYM	stjele
forkjetre	INVOLVED	avstand
forkjetre	HAS_HYPERONYM	forvrenge

Continued on next page

Table E.6: 2-Judge Dataset: List of relations accepted by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
forkludre	INVOLVED	uorden
forkorte	CAUSES	kort
formode	HAS_SYNONYM	anta
forspille	HAS_HYPERONYM	ødelegge
forta	HAS_HYPERONYM	forsvinne
fortjene	ENTAILS	verdig
forutsette	HAS_HYPERONYM	anta
fratre	HAS_HYPERONYM	slutte
friliste	INVOLVED	vare
fukte	CAUSES	fuktig
fukte	HAS_HYPERONYM	væte
fuske	CAUSES	dårlig
generere	HAS_SYNONYM	danne
generere	HAS_SYNONYM	frambringe
gildre	HAS_HYPERONYM	stille
gjelde	HAS_HYPERONYM	angå
gjenbruke	HAS_HYPERONYM	bruke
gjete	HAS_HYPERONYM	vokte
gjøne	HAS_SYNONYM	ape
gjøne	HAS_SYNONYM	fjase
gjøne	HAS_SYNONYM	narre
gjøne	HAS_SYNONYM	skjemte
glime	HAS_HYPERONYM	lyse
gramse	HAS_SYNONYM	famle
gramse	HAS_SYNONYM	grafse
gåre	INVOLVED	bølge
gåre	HAS_HYPERONYM	lage
halle	HAS_SYNONYM	helle
halle	HAS_SYNONYM	skråne
hele	HAS_SYNONYM	lege
hele	HAS_HYPERONYM	utbedre
helligholde	ENTAILS	hellig
henge	HAS_HYPERONYM	holde
hensette	HAS_SYNONYM	anbringe
hensette	HAS_SYNONYM	flytte
hensette	HAS_SYNONYM	sette
henvise	HAS_HYPERONYM	vise
hevde	HAS_HYPERONYM	vinne
himle	INVOLVED	himling
hogge	HAS_SYNONYM	bite
hogge	HAS_SYNONYM	gripe
hogge	HAS_SYNONYM	slå
hvitne	CAUSES	hvit
innbringe	HAS_HYPERONYM	bringe

Continued on next page

Table E.6: 2-Judge Dataset: List of relations accepted by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
inngi	HAS_HYPERONYM	sende
innkomme	HAS_HYPERONYM	komme
innskipe	HAS_HYPERONYM	bringe
instituere	HAS_HYPERONYM	stifte
instruere	ENTAILS	opplæring
instruere	HAS_HYPERONYM	veilede
intensivere	HAS_HYPERONYM	forsterke
isne	CAUSES	iskald
jodisere	HAS_SYNONYM	jodere
justere	HAS_SYNONYM	innstille
justere	HAS_SYNONYM	regulere
kalle	ENTAILS	navn
kannelere	INVOLVED	kannelyre
kannelere	HAS_HYPERONYM	lage
kante	HAS_HYPERONYM	velte
kase	HAS_HYPERONYM	legge
kaue	HAS_SYNONYM	hauke
kaue	HAS_SYNONYM	lokke
kitte	HAS_HYPERONYM	feste
kjøve	HAS_HYPERONYM	kvele
kjøve	INVOLVED	pust
klappe	HAS_HYPERONYM	legge
klargjøre	CAUSES	klar
klinge	HAS_SYNONYM	høres
klinge	HAS_SYNONYM	lyde
klusse	HAS_SYNONYM	fikle
klusse	HAS_HYPERONYM	fikle
klusse	HAS_SYNONYM	tukle
knuge	HAS_HYPERONYM	klemme
kommandere	INVOLVED	kommando
kompensere	HAS_SYNONYM	erstatte
kompensere	HAS_SYNONYM	godtgjøre
komplettere	CAUSES	fullstendig
komplettere	HAS_HYPERONYM	utfylle
komprimere	HAS_HYPERONYM	presse
konvoluttere	HAS_HYPERONYM	legge
krikle	HAS_HYPERONYM	lage
krinse	HAS_SYNONYM	kretse
krysse	HAS_HYPERONYM	gå
krølle	INVOLVED	krølle
kultivere	HAS_HYPERONYM	dyrke
kvare	HAS_HYPERONYM	bringe
kvittere	INVOLVED	gjengjeld
kåle	HAS_HYPERONYM	ødelegge

Continued on next page

Table E.6: 2-Judge Dataset: List of relations accepted by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
landsforvise	HAS_HYPERONYM	vise
lure	HAS_HYPERONYM	spekulere
lysne	CAUSES	lys
mane	HAS_SYNONYM	egge
mane	HAS_SYNONYM	formane
mane	HAS_SYNONYM	oppfordre
modne	CAUSES	moden
more	HAS_SYNONYM	muntre
mørke	CAUSES	mørk
mørne	CAUSES	mør
nedbygge	HAS_HYPERONYM	redusere
nitte	HAS_SYNONYM	klinke
nyte	HAS_HYPERONYM	fortære
okkupere	INVOLVED	beslag
omgjøre	HAS_HYPERONYM	lage
omtale	HAS_SYNONYM	anmelde
oppirre	HAS_HYPERONYM	ergre
oppskrive	HAS_SYNONYM	revaluere
oppta	INVOLVED	beslag
oppta	HAS_SYNONYM	besette
organisere	HAS_HYPERONYM	ordne
orientere	HAS_HYPERONYM	opplyse
overlate	HAS_HYPERONYM	utlevere
oversende	HAS_HYPERONYM	sende
overstige	ENTAILS	stor
parfymere	CAUSES	velluktende
perfeksjonere	CAUSES	fullkommen
perse	HAS_HYPERONYM	trykke
pigge	INVOLVED	pigg
plante	INVOLVED	plante
plaske	HAS_SYNONYM	skvette
plaske	HAS_SYNONYM	skvalpe
plire	HAS_SYNONYM	myse
plissere	HAS_HYPERONYM	legge
poppe	INVOLVED	popkorn
presentere	HAS_SYNONYM	framføre
prøve	HAS_SYNONYM	granske
prøve	HAS_SYNONYM	kontrollere
prøve	HAS_SYNONYM	teste
prøve	HAS_SYNONYM	undersøke
pudre	HAS_HYPERONYM	drysse
pudre	INVOLVED	pudder
puffe	HAS_SYNONYM	dampe
purre	HAS_HYPERONYM	varsle

Continued on next page

Table E.6: 2-Judge Dataset: List of relations accepted by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
rafte	HAS_HYPERONYM	legge
rafte	INVOLVED	raft
rakle	HAS_SYNONYM	drive
rakle	HAS_SYNONYM	flakke
rakle	HAS_SYNONYM	farte
rakle	HAS_SYNONYM	slenge
rape	HAS_HYPERONYM	gli
rape	ENTAILS	hørbar
ratte	HAS_HYPERONYM	styre
reduplisere	HAS_SYNONYM	fordoble
regjere	INVOLVED	makt
regjere	HAS_HYPERONYM	styre
reke	HAS_SYNONYM	drive
reke	HAS_SYNONYM	farte
reke	HAS_SYNONYM	slenge
renne	HAS_SYNONYM	støte
renne	HAS_SYNONYM	stikke
reprodusere	HAS_HYPERONYM	frambringe
respektere	INVOLVED	aktelse
respektere	INVOLVED	hensyn
returnere	HAS_HYPERONYM	sende
rive	HAS_HYPERONYM	rasere
rokere	HAS_HYPERONYM	utføre
romme	HAS_SYNONYM	inneholde
runge	ENTAILS	gjenlyd
ryke	HAS_HYPERONYM	tape
røpe	CAUSES	kjent
saktne	CAUSES	langsom
saluttere	HAS_HYPERONYM	skyte
sannsynliggjøre	CAUSES	sannsynlig
separere	HAS_SYNONYM	skille
sikle	INVOLVED	lyst
sive	HAS_SYNONYM	trenge
sjaue	INVOLVED	spetakkel
skeivle	HAS_HYPERONYM	bringe
skjefte	INVOLVED	skaft
skjule	HAS_SYNONYM	dølge
skjule	ENTAILS	hemmelig
skrike	HAS_SYNONYM	rope
skvette	HAS_SYNONYM	drive
skvette	HAS_SYNONYM	sprøyte
skvette	HAS_SYNONYM	sprute
skåle	HAS_HYPERONYM	drikke
slepe	HAS_SYNONYM	drasse

Continued on next page

Table E.6: 2-Judge Dataset: List of relations accepted by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
slepe	HAS_SYNONYM	hale
slodde	HAS_HYPERONYM	harve
smergle	HAS_HYPERONYM	slipe
smerte	CAUSES	vond
snike	HAS_SYNONYM	liste
snike	HAS_SYNONYM	lure
snuppe	HAS_HYPERONYM	skjære
sone	INVOLVED	bot
sortere	HAS_HYPERONYM	inndele
spante	INVOLVED	spant
spore	ENTAILS	hest
stable	HAS_HYPERONYM	legge
stanse	HAS_HYPERONYM	bringe
stille	HAS_HYPERONYM	møte
stjele	HAS_HYPERONYM	ta
strekke	HAS_SYNONYM	forstrekke
strenge	INVOLVED	streng
strigle	HAS_HYPERONYM	kjemme
strippe	INVOLVED	striptease
stryke	HAS_HYPERONYM	utgå
subordinere	HAS_HYPERONYM	underordne
subsidiere	HAS_HYPERONYM	hjelpe
summe	HAS_HYPERONYM	surre
sutenere	HAS_HYPERONYM	underholde
synkverve	HAS_HYPERONYM	forvirre
særmerke	HAS_SYNONYM	kjennetegne
særmerke	HAS_SYNONYM	særprege
tagge	INVOLVED	tagg
tette	CAUSES	tett
tilføye	HAS_HYPERONYM	føye
tilsløre	HAS_HYPERONYM	tåkelegge
tilsnakke	HAS_HYPERONYM	irettesette
tore	HAS_SYNONYM	våge
trasse	ENTAILS	trassig
tretne	CAUSES	trett
trinse	HAS_SYNONYM	rulle
trinse	HAS_SYNONYM	trille
tryte	INVOLVED	slutt
tåte	HAS_SYNONYM	patte
tåte	HAS_SYNONYM	sutte
ugge	ENTAILS	redd
urbanisere	CAUSES	bymessig
utbringe	HAS_HYPERONYM	bringe
utbygge	HAS_HYPERONYM	utforme

Continued on next page

Table E.6: 2-Judge Dataset: List of relations accepted by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
utlyse	HAS_HYPERONYM	kunngjøre
utstå	HAS_HYPERONYM	vente
vafle	HAS_HYPERONYM	slå
vanne	HAS_HYPERONYM	helle
vanne	INVOLVED	vann
vanne	INVOLVED	vann
vanne	ENTAILS	vann
varsle	ENTAILS	beskjed
vinne	ENTAILS	god
virke	INVOLVED	innvirkning
vite	ENTAILS	sikker
votere	INVOLVED	avstemning
votere	HAS_HYPERONYM	stemme
vringle	ENTAILS	vrang
ynde	ENTAILS	glad
ynde	HAS_HYPERONYM	like
yte	HAS_HYPERONYM	gi
åpne	CAUSES	synlig

Accepted Relations by Both Judges

Table E.7: 2-judge Dataset: List over relations invalidated by both judges, lemma level.

Lemma 1	Relation	Lemma 2
aksentuere	HAS_HYPERONYM	legge
applisere	HAS_SYNONYM	tilpasse
avdramatisere	CAUSES	liten
avsky	ENTAILS	sterk
besette	HAS_HYPERONYM	oppta
bestille	HAS_HYPERONYM	tinge
bestryke	HAS_HYPERONYM	holde
beundre	ENTAILS	over
dovne	CAUSES	midlertidig
dulle	HAS_HYPERONYM	vise
dølge	HAS_HYPERONYM	skjule
effektivisere	CAUSES	mye
etse	HAS_SYNONYM	gravere
finne	HAS_SYNONYM	kjenne
formode	HAS_HYPERONYM	anta
forurense	INVOLVED	ur
forutskikke	HAS_HYPERONYM	komme
framlegge	HAS_HYPERONYM	legge
halle	HAS_HYPERONYM	helle

Continued on next page

Table E.7: 2-judge Dataset: List over relations invalidated by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
harve	ENTAILS	over
henge	HAS_HYPERONYM	holde
henge	HAS_HYPERONYM	være
heroisere	HAS_HYPERONYM	betrakte
hoppe	HAS_HYPERONYM	sprette
innlosjere	ENTAILS	få
innrømme	ENTAILS	få
intensivere	CAUSES	mye
kontrollere	ENTAILS	over
korse	HAS_HYPERONYM	legge
kunne	ENTAILS	trygg
motorisere	HAS_HYPERONYM	legge
oppnorske	CAUSES	mye
oppta	HAS_HYPERONYM	besette
oversende	ENTAILS	over
overstige	HAS_HYPERONYM	overskride
plaske	HAS_SYNONYM	slå
plastre	INVOLVED	plast
presentere	HAS_SYNONYM	spille
prioritere	ENTAILS	fortrinn
reduplisere	HAS_HYPERONYM	fordoble
renne	HAS_HYPERONYM	helle
renne	HAS_HYPERONYM	helle
renne	HAS_HYPERONYM	stikke
samstave	HAS_HYPERONYM	komme
seigpine	ENTAILS	over
sjenere	HAS_HYPERONYM	være
skjule	HAS_HYPERONYM	dølge
sluke	HAS_HYPERONYM	legge
sluke	ENTAILS	under
slutte	HAS_HYPERONYM	holde
snike	HAS_HYPERONYM	lure
snyte	HAS_HYPERONYM	pusse
spare	HAS_HYPERONYM	bruke
spraye	HAS_HYPERONYM	bruke
stjele	HAS_HYPERONYM	legge
særmerke	HAS_HYPERONYM	kjennetegne
tabloidisere	ENTAILS	over
telefonere	HAS_HYPERONYM	ringe
undres	ENTAILS	over
utbasunere	CAUSES	alminnelig
velsigne	ENTAILS	over
virke	HAS_HYPERONYM	ha

Invalidated Relations by Both Judges

Table E.8: 2-Judge Dataset: List over relations disagreed by both judges, lemma level.

Lemma 1	Relation	Lemma 2
adherere	HAS_HYPERONYM	henge
aksentuere	HAS_HYPERONYM	framheve
aksentuere	INVOLVED	vekt
anrope	HAS_HYPERONYM	sende
anta	HAS_HYPERONYM	formode
antaste	HAS_HYPERONYM	tilsnakke
barbere	HAS_HYPERONYM	rake
besette	INVOLVED	beslag
besette	HAS_HYPERONYM	legge
besette	HAS_SYNONYM	oppta
bestille	INVOLVED	avtale
bestryke	ENTAILS	under
bevokte	HAS_HYPERONYM	holde
bevokte	ENTAILS	over
bilegge	HAS_HYPERONYM	legge
bo	HAS_HYPERONYM	finnes
borge	HAS_HYPERONYM	garantere
bre	ENTAILS	over
brodde	HAS_HYPERONYM	brydde
brotsje	HAS_HYPERONYM	bruke
deise	HAS_SYNONYM	falle
dimittere	HAS_HYPERONYM	sende
divertere	HAS_HYPERONYM	more
divertere	HAS_SYNONYM	more
dukke	HAS_HYPERONYM	komme
dølge	HAS_HYPERONYM	holde
eskapere	HAS_HYPERONYM	komme
evaporere	HAS_HYPERONYM	lage
fare	HAS_SYNONYM	gå
fastholde	ENTAILS	fast
figurere	HAS_HYPERONYM	stå
file	HAS_HYPERONYM	lage
finne	HAS_SYNONYM	synes
flanere	HAS_HYPERONYM	slentre
fly	HAS_SYNONYM	farte
fly	HAS_SYNONYM	springe
forakte	HAS_HYPERONYM	ringeakte
forfjamse	HAS_HYPERONYM	bringe
forkludre	HAS_HYPERONYM	bringe
forkludre	HAS_HYPERONYM	forkvakle
formalisere	HAS_HYPERONYM	bringe
formode	HAS_HYPERONYM	holde
formode	ENTAILS	sannsynlig
forutsette	ENTAILS	avhengig

Continued on next page

Table E.8: 2-Judge Dataset: List over relations disagreed by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
fri	ENTAILS	tilbud
fuske	HAS_HYPERONYM	slurve
generere	HAS_SYNONYM	avle
glemme	HAS_HYPERONYM	legge
gli	HAS_HYPERONYM	skli
gramse	HAS_SYNONYM	beføle
halse	HAS_HYPERONYM	fare
hele	HAS_SYNONYM	utbedre
helligholde	HAS_HYPERONYM	holde
himle	HAS_HYPERONYM	legge
hogge	HAS_SYNONYM	sette
holde	HAS_HYPERONYM	anse
hoppe	ENTAILS	urolig
hospitalisere	HAS_HYPERONYM	legge
implisere	HAS_HYPERONYM	medføre
inngi	HAS_HYPERONYM	gi
inngi	HAS_HYPERONYM	levere
innløpe	HAS_HYPERONYM	komme
innrømme	HAS_HYPERONYM	gi
insistere	HAS_HYPERONYM	holde
ise	HAS_HYPERONYM	legge
justere	HAS_SYNONYM	tilpasse
kapsle	HAS_HYPERONYM	omgi
kaue	HAS_SYNONYM	rope
kave	CAUSES	ubehjelpelig
kjørne	INVOLVED	merke
klinge	HAS_HYPERONYM	lyde
klore	HAS_HYPERONYM	hogge
knepre	HAS_HYPERONYM	lage
knuge	ENTAILS	over
koke	ENTAILS	sterk
kommandere	ENTAILS	over
komme	HAS_HYPERONYM	ankomme
kontrollere	INVOLVED	herredømme
kopulere	HAS_HYPERONYM	føye
krusle	HAS_HYPERONYM	kreke
krølle	HAS_HYPERONYM	lage
krølle	HAS_HYPERONYM	legge
kursivere	HAS_HYPERONYM	sette
kveike	HAS_HYPERONYM	styrke
lake	HAS_HYPERONYM	legge
leve	HAS_HYPERONYM	livnære
lugge	HAS_HYPERONYM	rykke
lugne	HAS_HYPERONYM	bringe

Continued on next page

Table E.8: 2-Judge Dataset: List over relations disagreed by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
lugne	HAS_HYPERONYM	roe
lære	HAS_HYPERONYM	forkynne
løpe	HAS_HYPERONYM	springe
løse	HAS_HYPERONYM	ordne
mane	HAS_SYNONYM	påvirke
more	HAS_SYNONYM	glede
more	HAS_SYNONYM	oppmuntre
more	HAS_SYNONYM	underholde
notere	HAS_HYPERONYM	opptegne
nå	HAS_HYPERONYM	komme
nå	HAS_HYPERONYM	komme
nå	HAS_HYPERONYM	komme
nå	HAS_HYPERONYM	oppnå
nøytralisere	HAS_HYPERONYM	motvirke
okkupere	HAS_HYPERONYM	legge
okkupere	HAS_HYPERONYM	oppta
omgås	ENTAILS	selskapelig
oppmyke	CAUSES	liten
oppnorske	HAS_HYPERONYM	fornorske
opprettholde	ENTAILS	fast
opprettholde	HAS_HYPERONYM	holde
oppta	HAS_HYPERONYM	besette
oppta	HAS_HYPERONYM	legge
overanstrenge	ENTAILS	sterk
overbelaste	ENTAILS	sterk
perfeksjonere	HAS_HYPERONYM	utdanne
permittere	ENTAILS	permisjon
plastre	HAS_HYPERONYM	legge
poppe	HAS_HYPERONYM	lage
presentere	HAS_HYPERONYM	framføre
prille	HAS_HYPERONYM	fingre
prøve	HAS_HYPERONYM	innøve
pudre	HAS_HYPERONYM	legge
pulse	HAS_HYPERONYM	pulsere
reduplisere	HAS_SYNONYM	gjenta
regne	HAS_HYPERONYM	utføre
renne	HAS_HYPERONYM	gli
renne	HAS_SYNONYM	helle
romme	HAS_HYPERONYM	inneholde
runge	HAS_HYPERONYM	dundre
ryke	HAS_HYPERONYM	sende
rå	HAS_HYPERONYM	herske
røpe	HAS_HYPERONYM	avsløre
saktne	HAS_HYPERONYM	bli

Continued on next page

Table E.8: 2-Judge Dataset: List over relations disagreed by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
se	HAS_HYPERONYM	oppdage
servere	HAS_HYPERONYM	framføre
sifrere	ENTAILS	siffer
sjaue	HAS_HYPERONYM	holde
sjaue	HAS_HYPERONYM	larme
sjokkåpne	ENTAILS	sterk
skade	ENTAILS	uheldig
skille	HAS_HYPERONYM	sprekke
skje	HAS_HYPERONYM	hende
skjene	HAS_HYPERONYM	fare
skjule	HAS_HYPERONYM	holde
skjære	HAS_HYPERONYM	gå
skrinlegge	HAS_HYPERONYM	legge
slepe	HAS_SYNONYM	dra
sluke	HAS_SYNONYM	kreve
slå	HAS_SYNONYM	gå
smashe	HAS_HYPERONYM	utføre
sneise	INVOLVED	korn
spenne	HAS_SYNONYM	sparke
spjære	HAS_HYPERONYM	flenge
sprike	HAS_HYPERONYM	spile
sprike	HAS_HYPERONYM	vise
sprute	HAS_HYPERONYM	sende
stjele	INVOLVED	beslag
streke	HAS_HYPERONYM	lage
strippe	HAS_HYPERONYM	utføre
stryke	HAS_HYPERONYM	sløyfe
sulle	HAS_HYPERONYM	nynne
svare	ENTAILS	gjenlyd
sveipe	ENTAILS	over
sælde	HAS_HYPERONYM	sikte
ta	HAS_HYPERONYM	skaffe
tagge	HAS_HYPERONYM	lage
terrassere	ENTAILS	form
tilføye	HAS_HYPERONYM	legge
tilkjennegi	ENTAILS	uttrykk
tilsløre	HAS_HYPERONYM	holde
tolke	ENTAILS	uttrykk
trimme	HAS_HYPERONYM	mosjonere
trinse	HAS_SYNONYM	tumle
ulke	HAS_HYPERONYM	gulpe
underkaste	INVOLVED	gjenstand
underlegge	INVOLVED	gjenstand
undertrykke	HAS_HYPERONYM	hemme

Continued on next page

Table E.8: 2-Judge Dataset: List over relations disagreed by both judges, lemma level. (Continued from last page)

Lemma 1	Relation	Lemma 2
undertrykke	HAS_HYPERONYM	holde
unnsleppe	HAS_HYPERONYM	komme
uteske	HAS_HYPERONYM	utfordre
vake	HAS_HYPERONYM	flyte
vake	HAS_HYPERONYM	holde
vanne	HAS_HYPERONYM	legge
vedlegge	HAS_HYPERONYM	legge
vinne	HAS_HYPERONYM	greie
virke	INVOLVED	virkning
votere	HAS_HYPERONYM	holde
være	HAS_HYPERONYM	snuse
vørde	HAS_HYPERONYM	ense

Relations With No Agreement Between the 2 Judges

0.13 Frequency List of PoS Patterns Grouped by Agreement

3 Judge Dataset

Table E.9: 3-Judge Dataset: Frequency list of PoS patterns found in fully agreed disambiguations.

Freq	PoS Pattern						
5	VERB	KOMMA	VERB	ADJ			
4	VERB						
4	VERB	KOMMA	VERB	KOMMA	VERB	KOMMA	VERB
4	VERB	KOMMA	VERB	PRON	PREP		
4	VERB	KOMMA	VERB	KOMMA	VERB		
4	VERB	ADJ					
4	VERB	KOMMA	VERB	PREP			
4	VERB	KOMMA	VERB	ADV			
4	VERB	KOMMA	VERB				
3	VERB	KOMMA	VERB	PRON			
3	VERB	ADJ	PREP				
3	VERB	SUBST	PREP				
3	VERB	ADJ	PREP	SUBST			
2	VERB	KOMMA	VERB	ADJ	PREP		
2	VERB	SUBST	CLB	VERB	PREP		
2	VERB	KOMMA	VERB	PREP	PREP		
2	VERB	SUBST	PREP	SUBST			
2	VERB	KOMMA	VERB	PRON	SUBST		
2	VERB	ADJ	SUBST				
2	VERB	KOMMA	VERB	ADJ	SUBST		
2	VERB	PREP					

Continued on next page

Table E.9: 3-Judge Dataset: Frequency list of PoS patterns found in fully agreed disambiguations. (Continued from last page)

Freq	PoS Pattern				
2	VERB	PARENTES-BEG	PREP	PARENTES-SLUTT	
2	VERB	PREP	PREP	SUBST	
2	VERB	SUBST	PREP	DET	
2	VERB	SUBST	ADV		
2	VERB	ADJ	CLB		
1	VERB	ADV			
1	VERB	SUBST	VERB	PREP	
1	VERB	KOMMA	VERB	PREP	SUBST
1	VERB	PREP	SUBST		
1	VERB	PREP	ADJ		
1	VERB	PRON	ADJ		
1	VERB	SUBST			
1	VERB	KOMMA	VERB	SUBST	
1	VERB	KOMMA	VERB	ADJ	ADJ
1	VERB	KOMMA	VERB	ADV	PREP
1	VERB	KOMMA	VERB	SUBST	PREP
1	VERB	PREP	PREP		
1	VERB	PREP	ADJ	SUBST	
1	VERB	KOMMA	VERB	VERB	PREP
1	VERB	KOMMA	VERB	PREP	ADJ
1	VERB	SUBST	CLB		
1	VERB	ADJ	ADV		

Table E.10: 3-Judge Dataset: Frequency list of PoS patterns found in majority agreed disambiguations.

Freq	PoS Pattern						
5	VERB	KOMMA	VERB	ADJ	PREP		
4	VERB	KOMMA	VERB	KOMMA	VERB	KOMMA	VERB
4	VERB	KOMMA	VERB	VERB			
3	VERB	KOMMA	VERB	SUBST	PREP		
3	VERB	KOMMA	VERB	PREP	PREP		
3	VERB	KOMMA	VERB				
3	VERB	KOMMA	VERB	KOMMA	VERB		
3	VERB	SUBST	PREP	SUBST			
3	VERB	KOMMA	VERB	ADJ	SUBST		
3	VERB	SUBST					
3	VERB	KOMMA	VERB	SUBST			
3	VERB	KOMMA	VERB	PREP	SUBST		
2	VERB	SUBST	CLB	VERB	PREP		
2	VERB	PREP	PREP	SUBST			
2	VERB	KOMMA	VERB	PRON	ADJ		
2	VERB	KOMMA	VERB	PREP			

Continued on next page

Table E.10: 3-Judge Dataset: Frequency list of PoS patterns found in majority agreed disambiguations. (Continued from last page)

Freq	PoS Pattern						
2	VERB	PREP	PREP				
2	VERB	ADJ	SUBST	PREP			
2	VERB	SUBST	PREP	DET			
2	VERB	PRON	SUBST				
1	VERB	ADJ	PREP	SUBST			
1	VERB	ADJ	PREP				
1	VERB	ADJ					
1	VERB	KOMMA	VERB	PRON	PREP		
1	VERB	PREP					
1	VERB	PREP	SUBST				
1	VERB	SUBST	PREP				
1	VERB	KOMMA	VERB	ADV			
1	VERB	KOMMA	VERB	VERB	PREP		
1	VERB	KOMMA	VERB	ADJ			
1	VERB	ADV					
1	VERB	KOMMA	VERB	PREP	ADJ		
1	VERB	PRON					
1	VERB	UKJENT	PARENTES-SLUTT				
1	VERB	SUBST	CLB				
1	VERB	KOMMA	VERB	PRON	SUBST		
1	VERB	SUBST	ADV				

Table E.11: 3-Judge Dataset: Frequency list of PoS patterns found in fully agreed invalidations .

Freq	PoS Pattern						
3	VERB	ADJ	ADJ				
2	VERB	VERB	PREP				
2	VERB	SUBST	PREP	DET			
1	VERB	KOMMA	VERB	SUBST	PREP		
1	VERB	ADJ	SUBST	PREP			
1	VERB	KOMMA	VERB	KOMMA	VERB	KOMMA	VERB
1	VERB	SUBST					
1	VERB	PREP	ADV				
1	VERB	KOMMA	VERB	ADJ	ADJ		
1	VERB	SUBST	PREP				
1	VERB	PREP	SUBST				
1	VERB	PREP	PREP	SUBST			
1	VERB	KOMMA	VERB	ADJ	SUBST		
1	VERB	PREP	PREP				
1	VERB	KOMMA	VERB	KOMMA	VERB		
1	VERB	KOMMA	VERB	VERB			
1	VERB	PREP	ADJ	SUBST			

Continued on next page

Table E.11: 3-Judge Dataset: Frequency list of PoS patterns found in fully agreed invalidations. (Continued from last page)

Freq	PoS Pattern				
1	VERB	KOMMA	VERB	PRON	ADJ
1	VERB	PREP	PRON		
1	VERB	SUBST	PARENTES-SLUTT		
1	VERB	KOMMA	VERB	SUBST	

Table E.12: 3-Judge Dataset: Frequency list of PoS patterns found in majority agreed invalidations .

Freq	PoS Pattern				
3	VERB	KOMMA	VERB	SUBST	PREP
2	VERB	PREP	ADJ	SUBST	
2	VERB	PREP			
2	VERB	ADV			
2	VERB	KOMMA	VERB	PREP	ADJ
2	VERB	ADJ	SUBST		
2	VERB	PREP	ADJ		
2	VERB	KOMMA	VERB		
2	VERB	PREP	SUBST		
2	VERB	PREP	SUBST	ADV	
1	VERB				
1	VERB	ADJ			
1	VERB	SUBST			
1	VERB	ADJ	PREP		
1	VERB	ADJ	SUBST	PREP	
1	VERB	KOMMA	VERB	PREP	SUBST
1	VERB	PREP	SUBST	PREP	
1	VERB	KOMMA	VERB	PRON	
1	VERB	KOMMA	VERB	VERB	
1	VERB	ADJ	ADJ		
1	VERB	KOMMA	VERB	SUBST	
1	VERB	KOMMA	VERB	PRON	SUBST
1	VERB	SUBST	ADV		
1	VERB	KOMMA	VERB	PRON	ADJ
1	VERB	SUBST	PREP		

Table E.13: 3-Judge Dataset: Frequency list of PoS patterns found for no agreement.

Freq	PoS Pattern				
1	VERB	SUBST	PREP	SUBST	
1	VERB	KOMMA	VERB	PRON	ADJ
1	VERB	KOMMA	VERB	PREP	SUBST

Continued on next page

Table E.13: 3-Judge Dataset: Frequency list of PoS patterns found for no agreement. (Continued from last page)

Freq	PoS Pattern				
1	VERB	KOMMA	VERB	KOMMA	VERB
1	VERB	KOMMA	VERB	PRON	
1	VERB	KOMMA	VERB	SUBST	PREP

2 Judge Dataset

Table E.14: 2-Judge Dataset: Frequency list of PoS patterns found in agreed disambiguations.

Freq	PoS Pattern						
16	VERB	KOMMA	VERB	KOMMA	VERB		
15	VERB	KOMMA	VERB				
14	VERB	ADJ					
13	VERB	SUBST	PREP				
13	VERB	KOMMA	VERB	ADJ			
12	VERB						
11	VERB	KOMMA	VERB	PREP			
10	VERB	KOMMA	VERB	SUBST	PREP		
10	VERB	SUBST					
9	VERB	PREP	SUBST				
9	VERB	KOMMA	VERB	ADV			
9	VERB	KOMMA	VERB	SUBST			
9	VERB	KOMMA	VERB	PREP	SUBST		
9	VERB	KOMMA	VERB	KOMMA	VERB	KOMMA	VERB
7	VERB	KOMMA	VERB	ADJ	PREP		
7	VERB	SUBST	PREP	SUBST			
7	VERB	PREP					
7	VERB	KOMMA	VERB	PRON	PREP		
6	VERB	KOMMA	VERB	PRON			
5	VERB	KOMMA	VERB	VERB			
4	VERB	KOMMA	VERB	PREP	PREP		
4	VERB	KOMMA	VERB	PRON	SUBST		
4	VERB	ADJ	PREP	SUBST			
4	VERB	SUBST	ADV				
4	VERB	KOMMA	VERB	PREP	ADJ		
3	VERB	PREP	PREP	SUBST			
3	VERB	SUBST	CLB	VERB	PREP		
3	VERB	ADJ	SUBST				
3	VERB	KOMMA	VERB	ADJ	SUBST		
3	VERB	ADJ	PREP				
2	VERB	ADV					
2	VERB	PARENTES-BEG	PREP	PARENTES-SLUTT			
2	VERB	SUBST	PREP	DET			
2	VERB	ADJ	CLB				

Continued on next page

Table E.14: 2-Judge Dataset: Frequency list of PoS patterns found in agreed disambiguations. (Continued from last page)

Freq	PoS Pattern				
1	VERB	SUBST	VERB	PREP	
1	VERB	KOMMA	VERB	PRON	ADJ
1	VERB	PREP	ADJ		
1	VERB	PRON	ADJ		
1	VERB	KOMMA	VERB	ADJ	ADJ
1	VERB	KOMMA	VERB	ADV	PREP
1	VERB	PREP	PREP		
1	VERB	PREP	ADJ	SUBST	
1	VERB	PRON	SUBST		
1	VERB	KOMMA	VERB	VERB	PREP
1	VERB	SUBST	CLB		
1	VERB	ADJ	ADV		

Table E.15: 2-Judge Dataset: Frequency list of PoS patterns found in agreed invalidations.

Freq	PoS Pattern						
7	VERB	KOMMA	VERB				
5	VERB	KOMMA	VERB	SUBST	PREP		
5	VERB	ADJ	ADJ				
3	VERB	KOMMA	VERB	ADJ			
3	VERB	PREP	PREP	SUBST			
3	VERB	KOMMA	VERB	SUBST			
2	VERB	ADJ	SUBST	PREP			
2	VERB	PREP	SUBST				
2	VERB	VERB	PREP				
2	VERB	ADJ	SUBST				
2	VERB	KOMMA	VERB	PREP	ADJ		
2	VERB	PREP					
2	VERB	KOMMA	VERB	PRON	ADJ		
2	VERB	SUBST	PREP				
2	VERB	SUBST	PREP	DET			
1	VERB						
1	VERB	KOMMA	VERB	KOMMA	VERB	KOMMA	VERB
1	VERB	SUBST					
1	VERB	PREP	ADV				
1	VERB	KOMMA	VERB	ADJ	ADJ		
1	VERB	KOMMA	VERB	ADJ	SUBST		
1	VERB	PREP	PREP				
1	VERB	KOMMA	VERB	ADJ	PREP		
1	VERB	KOMMA	VERB	KOMMA	VERB		
1	VERB	KOMMA	VERB	VERB			
1	VERB	ADV					

Continued on next page

Table E.15: 2-Judge Dataset: Frequency list of PoS patterns found in agreed invalidations. (Continued from last page)

Freq	PoS Pattern				
1	VERB	PREP	ADJ	SUBST	
1	VERB	PREP	PRON		
1	VERB	SUBST	PARENTES-SLUTT		
1	VERB	KOMMA	VERB	PRON	SUBST

Table E.16: 2-Judge Dataset: Frequency list of PoS patterns found for no agreement.

Freq	PoS Pattern						
12	VERB	KOMMA	VERB	SUBST	PREP		
8	VERB	KOMMA	VERB	SUBST			
8	VERB	KOMMA	VERB	PRON			
8	VERB	SUBST					
7	VERB	KOMMA	VERB				
7	VERB	KOMMA	VERB	ADJ			
7	VERB	KOMMA	VERB	KOMMA	VERB		
6	VERB	PREP	SUBST				
6	VERB	PREP					
6	VERB	KOMMA	VERB	ADV			
6	VERB	KOMMA	VERB	VERB			
6	VERB	KOMMA	VERB	PREP			
6	VERB	KOMMA	VERB	PREP	SUBST		
5	VERB	KOMMA	VERB	ADJ	PREP		
5	VERB	SUBST	PREP	SUBST			
5	VERB	KOMMA	VERB	KOMMA	VERB	KOMMA	VERB
5	VERB	ADJ	PREP				
4	VERB	KOMMA	VERB	PREP	PREP		
4	VERB	PREP	PREP	SUBST			
4	VERB	KOMMA	VERB	PREP	ADJ		
4	VERB	KOMMA	VERB	PRON	ADJ		
3	VERB	SUBST	PREP				
3	VERB	KOMMA	VERB	PRON	PREP		
3	VERB	KOMMA	VERB	ADJ	SUBST		
3	VERB	ADJ	SUBST				
3	VERB						
2	VERB	ADV					
2	VERB	PREP	ADJ	SUBST			
2	VERB	ADJ					
2	VERB	PREP	PREP				
2	VERB	ADJ	SUBST	PREP			
2	VERB	PREP	ADJ				
2	VERB	SUBST	PREP	DET			
2	VERB	SUBST	ADV				

Continued on next page

Table E.16: 2-Judge Dataset: Frequency list of PoS patterns found for no agreement. (Continued from last page)

Freq	PoS Pattern				
2	VERB	PREP	SUBST	ADV	
1	VERB	SUBST	CLB	VERB	PREP
1	VERB	PREP	SUBST	PREP	
1	VERB	KOMMA	VERB	VERB	PREP
1	VERB	ADJ	ADJ		
1	VERB	PRON	SUBST		
1	VERB	PRON			
1	VERB	UKJENT	PARENTES-SLUTT		
1	VERB	SUBST	CLB		

Bibliography

Den Danske Ordbog. http://ordnet.dk/ddo.

The Global Wordnet Association. http://www.globalwordnet.org/.

Eneko Agirre, Olatz Ansa, David Martinez, and Hovy E. Enriching wordnet concepts with topic signatures. In *Procceedings of the SIGLEX workshop on "WordNet and Other Lexical Resources: Applications, Extensions and Customizations". In conjunction with NAACL."*, 2001.

Antonietta Alonge, Nicoletta Calzolari, Piek Vossen, Laura Bloksma, Irene Castellón, Maria Antònia Martí, and Wim Peters. The linguistic design of the eurowordnet database. *Computers and the Humanities*, 32(2-3):91–115, 1998. URL http://dx.doi.org/10.1023/A:1001117508293.

Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 805–810, 2003.

Henning Bergenholtz, Ilse Cantell, Ruth Vatvedt Fjeld, Dag Gundersen, Jon Hilmar Jónsson, and Bo Svensén. *Nordisk Leksikografisk Ordbok*. Universitetsforlaget, 1997.

Ted Briscoe. *Computational lexicography for natural language processing*. Longman Publishing Group, White Plains, NY, USA, 1989.

Miriam Butt. The Light Verb Jungle. In *Harvard Working Papers in Linguistics*, volume 9, pages 1–49. Harvard, 2003.

Jean Carletta. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22:249–254, 1996.

Christiane Fellbaum. Semantic network of english verbs. In *WordNet: An Electronic Lexical Database*, pages 69–104. MIT Press, 1998a.

Christiane Fellbaum. A semantic network of english: The mother of all WordNets. In *Computers and the Humanities*, volume 32, pages 209–220. Kluwer Academic Publishers, 1998b.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998c.

Christiane Fellbaum and George A. Miller. Folk psychology or semantic entailment? a reply to rips and conrad (1989). In *Psychological Review 97*, page 565–570, 1990.

Ruth Vatvedt Fjeld, Julie Matilde Torjusen, and Rune Lain Knudsen. Fra alfabet til begrep: Bokmålsordboka og NorNet. In *Nordiska Studier i Lexikografi 11. Rapport från Konferensen om leksikografi i Norden, Lund 24 - 27 mai 2011.*, 2012. Submitted for publication.

J.L. Fleiss et al. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

Péter Halácsy, András Kornai, and Csaba Oravecz. HunPos – an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 209–212, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Sanda M. Harabagiu and Dan I. Moldovan. Knowledge processing on an extended wordnet. In *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

Franz Josef Hausmann and Herbert Ernst Wiegand. Component parts and structures of general monolingual dictionaries: A survey. In Franz Josef Hausmann, Oskar Reichmann, Herbert Ernst Wiegand, and Ladislav Zgusta, editors, *Wörterbücher: ein internationales Handbuch zur Lexicographie*, chapter 36, pages 328–360. Walter de Gruyter, 1989.

Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

Janne Bondi Johannessen, Kristin Hagen, Anders Nøklestad, and André Lynum. OBT+Stat: Evaluation of a Combined CG and Statistical Tagger. In Eckhard Bick, Kristin Hagen, Kaili Müürisep, and Trond Trosterud, editors, *NEALT Proceedings Series*, volume 14, pages 26–34. NEALT, 2011.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008.

Sophia Katrenko, Pieter W. Adriaans, and Maarten van Someren. Using local alignments for relation recognition. *J. Artif. Intell. Res. (JAIR)*, 38:1–48, 2010.

Dekang Lin. Automatic retrieval and clustering of similar words. 1998.

J. Lin. Divergence measures based on the shannon entropy. In *IEEE Transactions on Information Theory*, pages 145–151. IEEE Information Theory Society, January 1991.

Krister Lindén and Lauri Carlson. FinnWordNet - WordNet på finska via översättning. In *LexicoNordica*, volume 17, pages 119–140. Nordisk Forening for Leksikografi, 2010.

George A. Miller. Nouns in wordnet. In *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311, 1997.

Roberto Navigli and Simone Paolo Ponzetto. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Uppsala, Sweden, 11–16 July 2010, pages 216–225, 2010.

Lars Nygaard. Frå ordbok til ordnett. UiO, 2006. Cand. Philol. Thesis.

Patrick Pantel. Inducing ontological co-occurrence vectors. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 125–132, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219856. URL `http://dx.doi.org/10.3115/1219840.1219856`.

Bolette Sandford Pedersen, Sanni Nimb, Jørg Asmussen, Nicolai Hartvig Sørensen, Lars Trap-Jensen, and Henrik Lorentzen. Dannet: the challenge of compiling a wordnet for danish by reusing a monolingual dictionary. *Language Resources and Evaluation*, 2009.

Bollette S. Pedersen, Anna Braasch, Sanni Nimb, Jørg Asmussen, Nicolai Sørensen, Henrik Lorentzen, and Lars Trap-Jensen. Lingvistiske specifikationer for dannet version 2. Technical report, Center for Sprogteknologi, Københavns Universitet and Det Danske Sprog- og Litteraturselskab, 2011.

John I. Saeed. *Semantics*. Blackwell Publishers Ltd, 1997.

T. F Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147, 1981.

Dan Tufiş, Dan Cristea, and Sofia Stamou. Balkanet: Aims, Methods, Results and Perspectives. A General Overview. *Romanian Joural of Information Science and Technology*, 7(1-2):9–43, 2004.

Ellen M. Voorhees. Using wordnet for text retrieval. In *Wordnet: An Electronic Lexical Database*. MIT Press, 1998.

Piek Vossen. EuroWordNet general document. Technical report, University of Amsterdam, 2002.

Åke Viberg, Kerstin Lindmark, Ann Lindvall, and Ingmarie Mellenius. The Swedish WordNet Project. In *Proceedings of Euralex*, pages 407–412. Copenhagen University, 2002.