Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Maren Helene Orvedal

Generating Novel Compounds in Technical Translations

Master's thesis in Computer Science
Supervisor: Björn Gambäck
January 2020

NTNU
Norwegian University of Science and Technology

Maren Helene Orvedal

Generating Novel Compounds in Technical Translations

Master's Thesis in Computer Science, Autumn 2019

Department of Computer Science
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Abstract

Compound words are lexemes that are composed of multiple constituents. In principle, an infinite number of compounds can be formed and it is therefore impossible to list all possible compounds in a lexicon. When translating between a language that uses compounds and another one that does not, it is possible that a word that is represented by multiple words in the non-compounding language should be translated into a single word in the compounding language. The question then arises: how can novel compounds that have not yet been observed and that do not have a known translation be formed?

This thesis explores how novel compounds in the computer science domain can be formed when translating from English into Norwegian. For this purpose, a custom compound splitter is implemented and integrated into the translation pipeline of a statistical machine translation system. Specifically, the split-and-merge approach is adopted to form novel compounds, by splitting compounds prior to training and then merging them back together during postprocessing. The resulting translations are assessed by native speakers of Norwegian and rated in terms of fluency and semantics preserved. The compound splitter correctly splits 73.89 per cent of ambiguous compounds. The splitter is especially adept at recognising lexicalised *s* in Norwegian compounds, but struggles with the concept of epentheses. It also lacks support for hyphens, proper names, abbreviations and coordinated compounds. The compound splitter can assist Norwegian part-of-speech taggers in identifying constituents, in particular the compound head which usually determines the grammatical features of the compound.

The translation system's suggested translations are partially successful: translations are often accepted by native speakers when the constituents have been observed during training, but there are also interesting cases in which untranslated English words are assessed as fluent translations. On the other hand, the translation system is challenged by common problems like lexical ambiguity and out-of-vocabulary words. This is largely attributed to lack of in-domain data, despite efforts to acquire such a corpus.

Samandrag

Samansetjingar er leksem som er samansette av fleire ledd. Prinsipielt sett er det mogleg å forme eit uendeleg antal samansette ord, og det er derfor umogleg å ramsa opp alle moglege samansetjingar i eit leksikon. Når ein omset fra eit språk som nyttar samansetjingar til eit anna som ikkje gjer det, kan det hende at eit leksem som vert skrive som fleire ord på det fyrste språket skal samanskrivast på målspråket. Det fylgjande problemet oppstår då: korleis kan nye samansetjingar som ikkje har vorte observerte enno og som ikkje har ei kjend omsetjing verte framstilte?

Denne masteravhandlinga granskar korleis nye samansetjingar i datavitskapsdomenet kan verte framstilte når engelsk vert omsett til norsk. For å få til dette har ein spesialtillempa ordkløyvar vorte implementert og innlemma i omsetjingsprosessen til eit statistisk maskinomsetjingssystem. Ein framgangsmåte kjend som "kløyv-og-set-saman" vert nytta, der samansetjingar vert delte før læring og så ihopskrivne etter omsetjing. Dei resulterande omsetjingane vert deretter vurderte av morsmålstalarar og rangerte etter kor naturlege dei er og kor mykje av den opphavelege tydinga som vert halden på. Ordkløyvaren delar 73,89 prosent av tvitydige ihopskrivingar rett. Kløyvaren er særleg godt skikka til å kjenna att leksikalisert *s* i norske samansetjingar, men strevar med å dela opp ord med fugemorfem. Den manglar også stønad for samansetjingar med bindestrek, eigennamn, forkortingar og koordinerte samansetjingar. Ordkløyvaren kan hjelpa norske ordklasse-taggarar med å identifisera ledd, og då særleg hovudet av samansetjinga som vanlegvis fastset dei grammatiske eigenskapane til samansetjinga.

Omsetjingssystemet sine føreslegne omsetjingar er til dels vellukka: omsetjingar vert ofte godtekne av morsmålstalarar når ledda har vorte observerte under opptreningsfasen, men det finst også interessante høve der uomsette engelske ledd vert oppfatta som flytande omsetjingar. Omsetjingssystemet har likevel fleire vanlege problem som leksikalsk tvitydigheit og manglande ordforråd. Desse veikskapane vert i stor grad knytta til manglande relevante data, trass i forsøk på å skaffa fram eit slikt korpus.

Preface

This is the report for a Master's Thesis in the Computer Science programme at the Norwegian University of Science and Technology (NTNU), written during the Autumn semester of 2019. The thesis is written on the topic of Computational Creativity and has been supervised by Björn Gambäck.

I would like to thank my supervisor Björn Gambäck for invaluable feedback and guidance throughout the project, as well as my friends, family, and partner, Christopher, for feedback and encouragements during the writing. I would also like to express my gratitude to everyone who participated in the survey and who provided vital data and comments.

Maren Helene Orvedal
Trondheim, 12th January 2020

Contents

Glossary **xiii**

Conventions **xvii**

1. Introduction **1**
1.1. Background and Motivation 1
1.2. Goals and Research Questions 2
1.3. Research Method 3
1.4. Contributions 3
1.5. Thesis Structure 4

2. Background Theory **5**
2.1. A Brief History of Machine Translation 5
2.2. Fundamentals of Statistical Machine Translation 7
2.3. Practical Introduction to the Norwegian Language 8
2.4. The Essentials of Norwegian Compounds 8

3. Related Work **11**
3.1. Literature Review of Related Work on Compound Splitting 11
3.1.1. Johannessen and Hauglin (1998) 11
3.1.2. Koehn and Knight (2003) 12
3.1.3. Sjöbergh and Kann (2004) 13
3.1.4. Alfonseca et al. (2008) 13
3.1.5. Macherey et al. (2011) 14
3.1.6. Clouet and Daille (2013) 14
3.1.7. Stymne et al. (2013) 15
3.1.8. Riedl and Biemann (2016) 15
3.2. Related Work on Statistical Machine Translation 16
3.2.1. Claveau (2008) 16
3.2.2. Niehues and Waibel (2011) 16
3.2.3. Arcan et al. (2012) 17
3.2.4. Stymne et al. (2013) 17
3.2.5. Cap et al. (2014) 18
3.2.6. Gujral et al. (2016) 18

4. Data Sets and Knowledge Sources **19**
4.1. Knowledge Sources for a Compound Splitter . . . 19
4.1.1. Toolkits and Projects Concerning Compound Splitting . . . 19
Toolkits for Compound Splitting . . . 19
Compound Splitting Projects . . . 19
4.1.2. Corpora and Lexica for a Compound Splitter . . . 20
Bokmålsordboka . . . 20
Norsk Ordbank . . . 20
The Oslo-Bergen Tagger . . . 21
The Norwegian Dependency Treebank . . . 21
NoWaC . . . 21
HaBiT . . . 21
The Oslo Corpus of Tagged Norwegian . . . 22
The NST Lexical Database . . . 22
4.2. Knowledge Sources and Data Sets for a Translation System . . . 22
4.2.1. The English-Norwegian Parallel Corpus . . . 23
4.2.2. LOGON . . . 23
4.2.3. OPUS . . . 23
4.2.4. TED . . . 24
4.2.5. The ELRC-SHARE Repository . . . 24
4.3. Training Sets . . . 25
4.3.1. Training Set for the Compound Splitters . . . 25
4.3.2. Errors and Inconsistencies in the NST Corpus . . . 25
4.3.3. Training Set for the Machine Translation System . . . 26
4.4. Test Sets . . . 27
4.4.1. Test Sets and Evaluation for the Compound Splitters . . . 27
4.4.2. Evaluation Metrics of the Compound Splitters . . . 29
4.4.3. Test Sets for the Machine Translation System . . . 30

5. Architecture **31**
5.1. Architecture of the Splitters . . . 31
5.1.1. Architecture of the Statistical Approach . . . 32
Architecture of the Baseline Method . . . 32
Architecture of the POS method . . . 32
Architecture of the N-Gram Method . . . 35
Architecture of the Hybrid Method . . . 36
5.1.2. Architecture of the Machine Learning Approach . . . 36
5.2. Choosing a Compound Splitter . . . 36
5.3. Architecture of the Translation System . . . 38
5.3.1. Preparing Training Data . . . 39
5.3.2. Tagging the Training Data . . . 40
5.3.3. Training Phase . . . 42
5.3.4. Decoding Phase . . . 43

5.3.5. Postprocessing 44

6. Experiments **45**
6.1. Experiments and Results of the Splitters 45
6.1.1. Experimental Plan 45
6.1.2. Experimental Setup 45
6.1.3. Experimental Results 46
6.2. Experiments and Results of the Translation System 48
6.2.1. Experimental Plan 48
6.2.2. Experimental Setup 49
6.2.3. Experimental Results 51

7. Evaluation **55**
7.1. Evaluation of the Compound Splitters 55
7.1.1. Correctness of Splitting Compounds Found in the Corpus 55
7.1.2. Analysis of the Baseline Performance 56
7.1.3. Analysis of the POS Method Performance 56
7.1.4. Analysis of the 4-Gram Method Performance 57
7.1.5. Analysis of the Hybrid Method Performance 58
7.1.6. Analysis of SECOS Performance 59
7.1.7. Error Analysis of the Compound Splitters 59
7.2. Evaluation of the Translation System 59
7.2.1. Evaluation of the Machine Translations 59
7.2.2. Evaluation of the Control Translations 60
7.2.3. Error Sources 62
Errors in the Training Data 62
Errors Made by the Tagger 63
Errors Concerning Lexical Ambiguity 64
Human Errors 64

8. Discussion **65**
8.1. Discussion of the Implemented Splitters 65
8.2. Discussion of the Translation System 66
8.2.1. Effect of Out-Of-Vocabulary Words 67
8.2.2. The Effect of Lexical Ambiguity 68
8.2.3. Effects of the EPOS Tagset 69
8.2.4. The Effect of Preprocessing 69
8.2.5. The Effect of Postprocessing 70
8.2.6. The Effect of Contrasting Training and Test Data 71
8.2.7. The Effect of Corpus Size 72
8.2.8. The Effect of Compound Splitting 72
8.2.9. The Effect of Domain Adaption 73
8.2.10. Producing Known Compounds 74
8.3. Limitations of the Translation System 74

9. Conclusion and Future Work **77**
9.1. Contributions 77
9.2. Conclusion 78
9.3. Future Work 79

Bibliography **81**

Appendices **89**

A. Complete List of Ambiguous Compounds **91**

B. Full Test Results **111**

C. Installation Guide **137**
C.1. Contents of the Attached Code 137
C.2. Requirements for the Compound Splitters 141
C.3. Installing Moses and Requirements 141
C.4. Installing OBT 142
C.5. Final File Hierarchy 144

D. Reproducing Experiments **147**
D.1. Reproducing the Splitter Experiments 147
D.2. Generating the Training Data 148
D.3. Training the Decoder 148
D.4. Reproducing Novel Compounds 151

List of Figures

1.1. An example of Google Translate making a lexical ambiguous mistake. . . 2

5.1. Logic flow diagram of the baseline method. 33
5.2. Logic flow diagram of the POS method. 34
5.3. Logic flow diagram of the n-gram method. 35
5.4. Logic flow diagram of the hybrid method. 37
5.5. Visual representation of the translation process architecture. 38
5.6. Logic flow diagram of the modified compound splitter. 41

6.1. An example of a survey question. 50
6.2. Assessments of the worst machine translations. 53

7.1. Assessments of the control translations. 61

List of Tables

4.1. Distribution of POS combinations in NST. 26

6.1. Performances of the implemented splitters. 47
6.2. Performance of 4-gram and Hybrid using highest probability. 48
6.3. Best rated machine translations. 52

A.1. Full list of ambiguous compounds. 92
A.2. Full list of the *Epenthetic e* test set. 96
A.3. Full list of the *Epenthetic s* test set. 97
A.4. Full list of the *Johannessen* test set. 99
A.5. Full list of the *Lexicalised e* test set. 100
A.6. Full list of the *Lexicalised s* test set 101
A.7. Full list of the *No epenthesis* test set. 102
A.8. Full list of the *No split* test set. 104
A.9. Full list of the *Representative* test set. 105

B.1. English input terms and Norwegian translations. 111
B.2. English control translations and human Norwegian translations. 114
B.3. Translations and tagged output made by the translation system. 115
B.4. Ratings of the machine translations. 121
B.5. Ratings of the control translations. 134

Glossary

affix A bound morpheme that attaches to a root or stem to form a new lexeme (derived form) or an inflected form or stem of an existing lexeme, e.g., prefixes or suffixes that are attached to the beginning or the end of a lexeme, respectively. 12, 21

ambiguity A situation in which a word has more than one related meaning. Often called lexical ambiguity to be distinguished from structural ambiguity, which occurs at sentence level (in which a sentence has more than one related meaning) which can be induced by lexical ambiguity. 22

articifical intelligence Abbreviated AI. The study of "intelligent agents"; i.e., any device that perceives its environment and acts in a way that maximises its chance of successfully achieving its goal. Colloquially, the term often describes machines and/or computers that mimic "cognitive" functions that are associated with the human mind, such as learning and problem solving. 6, 80

calque A word or phrase adopted from another language by literal, word-for-word translation. For instance, many languages use their native word for 'mouse' as a translation for 'computer mouse', due to the English precedence. In Norwegian, it is indeed called *mus*. 75

compound A derived form resulting from the combination of two or more lexemes. 8, 10–15, 17, 18, 20, 21, 25, 36, 39, 42, 55, 56, 63, 65, 67, 69, 70, 73–75, 77

compound merging The opposite process of decompounding; i.e., reassembling decomposed consituents into a correctly formed compound. 17, 18, 39, 44, 70, 77

compound splitter A program that attempts to automatically split a compound into its constituents and epentheses (if any). 3, 4, 11, 13, 14, 19, 31, 36, 45, 65, 72, 77, 79, 144

constituent The lexemes that a compound word is composed of. 9–11, 14, 15, 17–21, 25, 26, 32, 36, 42, 45, 55, 56, 59, 64, 66, 67, 74, 75, 79

content word A word that refers to objects, events, and abstract concepts; contrasts with function word. Also called lexical word. 12, 73

corpus A large and structured set of texts, used for statistical analysis and gathering linguistic features. 6, 7, 11, 12, 14, 15, 17–20, 23, 25, 26, 32, 39, 42, 56, 59, 62, 63, 68, 71, 72, 77, 79

decompounding The process of splitting a compound into its constituents and epentheses (if any). 3, 11, 14, 22, 31, 55, 66, 77, 78, 91

derivation The creation of a new lexeme from one or more other lexemes through some morphological process like affixation or compounding. Also called lexeme formation and word formation. Derivation contrasts with inflection which concerns the same lexeme. 10, 16, 71

endocentric compound A type of compounds whose meaning equals that of the sum of its constituents; e.g., *blåbær* (lit. blue berry) is a berry that is blue. 55, 75

epenthesis The insertion of a phonological segment or segments between morphemes. In this text, it is used to mean the additional letter(s) sometimes added between the constituents of a compound (see Section 2.4). In Norwegian, an epenthesis is known as *fuge* or *fugemorfem*. 2, 9, 11, 14, 17, 20–22, 28, 32, 34, 36, 56, 57, 59, 65–67, 74, 77–79

etymology The study of the history of words; i.e., the origin of some particular word. 10, 20, 55, 73

exocentric compound A type of compounds whose meaning is not equal to that of the sum of its constituents; e.g., *bjørnebær* (lit. bear berry) is not a berry made of bears. 55, 73

function word A word, such as a determiner, conjunction, or modal, that has a grammatical function and is best characterised by this function. 9

head The rightmost constituent in a compound. It (usually) determines grammatical features of the compound; e.g., *bær* (berry) in *blåbær* (blueberry). 21, 26, 35, 41, 42, 58, 63, 70, 73, 77

inflection The formation of grammatical forms of a single lexeme. Is, are, and being are examples of inflected forms of the lexeme BE. 10, 20, 25, 66

language model Abbreviated LM. A probability distribution over sequences of words used to ensure fluent output in machine translation. 7, 27, 42, 62, 63

lemma Generally, the form of a lexeme that is listed in a dictionary, usually the infinite form for verbs and the singular form of nouns. 20, 80

lexeme A word with a specific sound and a specific meaning. Its shape may vary depending on syntactic context, i.e., different inflectional forms. E.g., the lexeme PERFORM has inflected forms *perform*, *performs*, and *performed*. 21, 22, 25, 32, 55–57, 59, 62, 63, 65–67, 71, 75, 78, 79

lexical ambiguity A situation in which a word has more than one related meaning. Often called lexical ambiguity to distinguish it from structural ambiguity, which occurs at sentence level (in which a sentence has more than one related meaning). Structural ambiguity can be induced by lexical ambiguity. 2, 4, 11, 13, 27, 62, 64, 68, 71, 73, 78, 79, 91

lexicon A corpus' inventory of lexemes; i.e., a list of the vocabulary contained in that corpus. 11, 12, 17, 21, 32, 42, 55, 64, 72, 79, 142

loanword A word adopted from one language, directly incorporated into another language without translation. 75, 79, 121

memoisation An optimisation technique used to speed up programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again. 34, 79

modifier The first constituent(s) in a compound. It adds some new meaning to the other part(s); e.g., *blå* (blue) in *blåbær* (blueberry). 21, 26, 35, 43, 63, 67, 69, 70, 72, 73, 77

morpheme A minimal unit of grammatical structure. In this text it conveys the meaning of the smallest part that carries (part of) the meaning of a whole word. 10, 14, 26

morphology The linguistic field of words, how words are formed, and their relationship to other words in the same language. Morphology concerns the structure of words and parts of words, such as stems, root words, prefixes, and suffixes. 9, 14, 16, 20, 21, 63, 80

natural language processing Abbreviated NLP. A combined field of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages. 2, 19

neural machine translation Abbreviated NMT. A variant of the SMT approach that aims to build a single neural network that can be jointly tuned to maximise the translation performance. Practically all recent work has been based on the attention-based encoder-decoder model of Bahdanau et al. (2015). 6, 7, 17

ontology The representation, formal naming and definition of the categories, properties and relations between concepts, data and entities that substantiate one, many or all domains of discourse. 17

orthography The set of conventions for writing a language, including norms of spelling, hyphenation, capitalisation, word breaks, emphasis, and punctuation. 20, 22, 64, 67, 70, 71

part-of-speech Abbreviated POS. The categorisation of words (lexemes) of similar grammatical behaviour. Also called word class. 12, 17, 21, 22, 26, 32, 36, 40, 42, 44, 56, 63, 66, 69, 78, 142

phoneme A speech sound in a specific language that, if swapped with another phoneme, could change one word to another. 75

phrase-based statistical machine translation Abbreviated PBSMT. A variant of the SMT approach which aims to reduce the restrictions of word-based translation by translating whole sequences of words. These phrase sequences have no inherent relation to linguistic phrases. 7, 12, 17, 42

portmanteau A blend of words, in which parts of multiple words or their phonemes are combined into a new word. 75

prefix A non-root morpheme that precedes the root of a word. 10

productivity The degree to which speakers of a language use a given grammatical process. In this report it is used to mean how much this process contributes to word formation (lexicalisation) of new compounds. 8, 16, 17, 55, 69, 78

root The "core" of a word, i.e., the morpheme that makes the most precise and concrete contribution to the word's meaning. 10, 32, 42

semantics The meaning that different components of a language carry. 5, 9, 10, 18, 39, 75

statistical machine translation Abbreviated SMT. A machine translation approach in which translations are made from statistical models whose parameters are derived from the analysis of parallel corpora. 3, 4, 6, 7, 11, 12, 15, 17, 22, 42, 63, 67, 78

suffix A non-root morpheme that follows the root of a word. 10, 58, 59

syntax The set of rules, principles, and processes that govern the structure of sentences in a language. 7, 8, 10, 21

translation memory A database that stores "segments"; i.e., sentences, paragraphs or sentence-like units that have previously been translated, for the purpose of aiding human translators. 25, 26, 39

Conventions

Throughout this thesis, it will sometimes be necessary to exemplify Norwegian words to demonstrate certain aspects of the system. For consistency, the following conventions are used in such discussions:

- Italics: italicised text represents a Norwegian word; e.g., *norsk*.
- Parentheses: when a Norwegian word has been introduced and a translation is appropriate, the translation will be encapsulated in parentheses; e.g., *norsk* (Norwegian).
- Double quotation marks: text enclosed in double quotation marks, “ and ”, either indicates direct speech, or that the quoted text should not be understood literally; e.g., the root is the “core” of a word.
- Single quotation marks: text enclosed in single quotation marks, ‘ and ’, indicates an emphasis on this text, like below for dashes and square brackets.
- Dashes: dashes, ‘-’, are used to point out where a word is split; e.g., *norsk-lærer* (Norwegian teacher).
- Square brackets: square brackets, ‘[’ and ‘]’, either point out context for a word or other grammatical details; e.g., *elger* (moose [plural]).
- Asterisk: the symbol ‘*’ represents the fact that a given split or grammatical formulation is incorrect; e.g., *telefon-svarer*, not **telefon-s-varer*.

1. Introduction

The goal of any machine translation system is to translate new, previously unseen input. However, this objective is most commonly understood on the sentence level; i.e., all words that are needed in the new sentence have already been seen during training, and need only be rearranged into an unseen permutation. This raises the question of what a translation system should do when a term in the source language has not been observed during training, or it has been seen, but makes up part of a larger compound term. This question is especially prevalent in domain-specific texts, where limited amounts of training data is available, and that data may contain entirely newfangled terms.

This thesis presents the hypothesis that it is possible to translate such unseen terms, in particular into compound terms, based on rearranging previously seen compound parts into composite words, just like one would words into novel sentences.

1.1. Background and Motivation

The problem of translating compound terms is well explored in the literature, and extensive research exists on the topic. However, the problem description differs based on whether the translation direction is from a compounding source language into another compounding target language, from a compounding source language into a non-compounding target, or from a non-compounding source into compounding target. This thesis concerns the latter case, translating from English into Norwegian.

As mentioned above, the problem of translating unknown terms is especially common in domain-specific translation. One domain in the Norwegian language which has traditionally employed a large number of loanwords from English is the technical and computer-related domain. *Språkrådet* (the Norwegian Language Council) maintains a list of computer terminology translated into Norwegian[1], but the list is in no way exhaustive, and it is not frequently updated[2]. However, there has recently been an increase in the demand for Norwegian terminology in this field. In November 2018, Norwegian politicians Iselin Nybø and Trine Skei Grande held a conference on how to strengthen the Norwegian language in academia, see Kulturdepartementet and Kunnskapsdepartementet (2018). The politicians presented challenges with the popular belief that English is easier to use in academic publications because it has a larger vocabulary, and asked for a more conscious relationship concerning what materials are to be published in Norwegian and English.

[1] `https://www.sprakradet.no/sprakhjelp/Skriverad/Ordlister/Datatermar/`

[2] At the time of writing, the list had not been updated for four years (since 17 December 2015).

No more than two months later, Nikolai Astrup was named Minister of Digitalisation[3]. In an interview with *Språknytt*, Botheim (2019), he declared that "language is important for digitalisation. To achieve regulations that are suited for digitalisation, we need a language that is suited for digitalisation."

Furthermore, existing tools are not yet adequately equipped for the challenges of lexical ambiguity between domains. Figure 1.1 shows a screenshot of a translation made by Google Translate[4] in December 2019; the general phrase 'milk and cookies' is erroneously translated into the Norwegian equivalent of 'milk and stateful information'. *Informasjonskapsler* is indeed a valid translation for 'cookies', but only in certain domains; here, the contextual clue provided by 'milk' makes this translation nonsensical to native speakers. Lexical ambiguity is one of the major unsolved problems in machine translation and natural language processing.

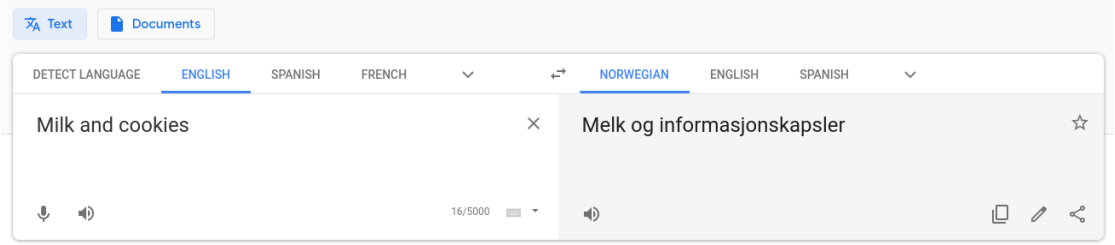


Figure 1.1.: An example of Google Translate making a lexical ambiguous mistake.

1.2. Goals and Research Questions

The demand for Norwegian technical terms motivates the development of a system that can automatically produce such novel expressions. The goal of this thesis has therefore been formulated as follows:

Goal *Investigate the split-and-merge strategy's capability of producing unseen terms in the computer domain when translating from English to Norwegian, focusing on compound terms.*

As will be shown subsequently in this thesis, the common consensus in the NLP (natural language processing) community today is that the best way to translate into compounding languages is by using the split-and-merge strategy: compounds are split prior to training and then merged back together in a postprocessing step. Successful experiments with this approach have already been conducted, but not on languages with limited available resources, like Norwegian. Furthermore, compound forming is a complex process in Norwegian, often involving epentheses (phonemes interposed between

[3] *Digitaliseringsminister*

[4] `https://translate.google.com/#view=home&op=translate&sl=en&tl=no&text=Milk%20and%20cookies`

compound parts for ease of pronunciation; see Section 2.4 or the glossary), and it is therefore important to verify whether the strategy is able to handle these irregularities.

Research question 1 *How applicable are current approaches of domain-specific translation to Norwegian translations?*

Domain-specific translation is a well-explored field which is proven to improve translation quality. However, techniques used for translating one language pair may not work as well for other ones. Thus, it is necessary to investigate whether the most popular state-of-the-art approaches do indeed produce useful results for English-Norwegian translation.

Research question 2 *Does the split-and-merge approach give comparable results for Norwegian in terms of producing novel, unseen compounds?*

As will be shown in Chapter 3, several approaches to producing novel compounds already exist. However, no published materials have been explicitly applied to Norwegian, and most work has been done on compounding languages with readily available large data sets. This thesis will look into whether this affects the degree to which one can produce novel compounds, and whether these results are comparable to research on other languages.

Research question 3 *How natural do native speakers find the novel translations?*

Evaluating novel translations with automatic frameworks that compare against reference translations is impossible because there are no translations to compare against. The simplest way to get meaningful evaluation on novel output is to ask native speakers whether they find these suggestions to be acceptable translations. Automatic surveys will be distributed to measure this.

1.3. Research Method

The research presented in this thesis consists of integrating a statistical machine translation system and a compound splitter, training the system on in-domain data, and evaluating the output by the use of native speakers. The compound splitter is implemented as a customised version of previous research in the field, adapted to Norwegian compounds. The splitter supports a popular Norwegian part-of-speech tagger by treating unknown compounds and decomposes them before translation. The ensuing evaluations are conducted using online surveys. Subsequently, the answers are collected and compared against the survey participants' assessment of human-made translations in the same field.

1.4. Contributions

This project's first and foremost contribution is a custom Norwegian compound splitter that can assist Norwegian part-of-speech taggers that lack decompounding capabilities.

The splitter is implemented based on the part-of-speech method presented by Sjöbergh and Kann (2004). Despite struggling with the concept of epentheses, the splitter performs very well for compounds with lexicalised *s*. The splitter has an accuracy of 0.7389 on a test set of various types of compounds (ambiguous compounds, compounds with epentheses, and compounds without epentheses). Nine test sets designed to identify what type of errors the compound splitter makes are also attached to this project's source code.

This splitter is integrated into the translation pipeline of a statistical machine translation system. This translation system uses the Moses decoder, see Koehn et al. (2007). The system is able to produce compounds that were not seen during training, but native speakers' assessments of these translations are conflicted.

Finally, Chapter 4 provides a detailed review of available data sources that can be used in similar projects. Chapter 8 also discusses some non-ideal choices made during experimentation that should be avoided in similar projects.

1.5. Thesis Structure

This thesis concerns background material, the development of a custom compound splitter, the integration of an SMT (statistical machine translation) system and the splitter, in addition to the evaluation and discussion of said translation system. Due to the tightly coupled nature of the compound splitter and its role in the translation system, chapters 3 through 8 are dual: the first part of each chapter concerns the decompounder whereas the second one discusses the translation system as a whole. Here follows a description of how the chapters of this thesis are laid out.

Chapter 2 presents some basic theory of SMT systems, and some linguistic principles that are necessary for following the discussions throughout this thesis. Chapter 3 discusses similar work that has looked into how to produce compound splitters, and translation systems that have dealt with the translation of compounding languages. In Chapter 4, potential data sources for a custom decompounder are explored, and available resources for an English-Norwegian translation are evaluated. The first half of Chapter 5 presents a number of splitters that are tested in the following chapter. The second part shows how one of these splitters is integrated into the translation system's architecture to produce Norwegian translations. Subsequently, translation experiments and experiments on the splitters are conducted in Chapter 6. The experiments result in native speakers' assessments of machine translations which are presented in Chapter 7. These results are closely examined in Chapter 8. Finally, the conclusions for this thesis as well as suggestions for future work are propounded in Chapter 9.

The appendices present additional details on the topics discussed throughout the thesis. Appendix A lists all ambiguous compounds that were used in the evaluation of the splitters, and Appendix B presents the complete experimental results from Section 6.1. Appendix C presents installation instructions for setting up the environment used in the same experiments. Ultimately, instructions on how to reproduce the presented results are found in Appendix D.

2. Background Theory

This chapter serves as an orientation of how the research field of machine translation has ended up where it is today. The following section presents a timeline of translation approaches and their corresponding responses and advancements. Section 2.2 introduces the fundamental workings of modern machine translation systems, which the implementation in Section 5.3 makes use of. Sections 2.3 and 2.4 present some linguistic principles that are especially prevalent for the Norwegian language and its usage of compounds.

2.1. A Brief History of Machine Translation

This presentation is largely based on that of Hutchins (1997) and Koehn (2010). Further details can be found in the respective publications.

According to Hutchins (1997), the first time machine translation (MT) is mentioned is in a letter from Warren Weaver, who, in March 1947, wrote to cyberneticist Norbert Wiener about the problem of translation. Weaver wondered whether the problem could be cast as a cryptographic one: looking at an article in a foreign language, he suggested that it was in fact written in English, but had simply been encoded with strange symbols. Translating it back to English would thus be a task of decoding.

In 1948, Andrew Booth and Richard Richens did some tentative experiments, and the following year Weaver wrote his now-famous memorandum to launch research on MT, published in Weaver (1955).

At this point in time, computers were very large and expensive and required multiple maintenance engineers and a dedicated staff of operators and programmers. Most of the work was mathematical, either done for military institutions or for university departments of physics and applied mathematics, but with strong links to the armed forces.

In light of these circumstances, it may be unsurprising that much of the earliest work on MT was supported by military and intelligence funds, and that it was not intended for public use. In the United States, a large portion of the research emphasised translating from Russian to English (and vice versa in the Soviet Union).

Koehn (2010) tells how, in the Georgetown experiment of 1954, such a Russian-English translation system was demonstrated. This led many to believe that the problem of translation was almost solved. However, the system was limited to a 250-word vocabulary and six grammar rules, and already then, sceptics claimed that the problems concerning semantic disambiguation were impossible to solve automatically. Still, work continued with great optimism.

In 1966, MT research was nearly terminated upon the issue of the ALPAC[1] report, see National Academy of Sciences (1966). In this document, the US funding agencies were informed that post-editing machine translations was not cheaper or faster than full human translation. The fact was that there was very little Russian scientific literature worth translating and that there were plenty of human translators apt for the task. The committee thus concluded that there was no advantage in using machine translation systems, and suggested that funding rather be directed into basic linguistic research and the development of methods to improve human translation. The authors of the report had compared the quality of current systems with the artificially high quality of the Georgetown demonstration, making current efforts look bleak. As a consequence, funding for MT in the United States stopped almost entirely.

Research continued on a much reduced scale until 1970, when the Systran Russian-English translation system was adopted by the US Air Force. A French-English version was bought by the European Commission in 1976, and systems for more European language pairs soon followed.

In the 1980s, MT efforts resumed all over the world. Syntactic formalism grew more sophisticated, including reversible grammars that may be used for both analysis and generation. One research trend that emerged during this period was the focus on interlingua systems that represent meaning independently of a specific language. The idea of representing meaning in a formal way tied research from both AI (articifical intelligence) and computational linguistics together. Language-independent representations were used because a proper theory of meaning seemed to address the problem at a more fundamental level than mapping lexical or syntactic units at a low level. To this date, formally representing meaning remains one of the grand challenges in AI.

Proceeding into the 1990s, corpus-based methods, notably the introduction of statistical methods and example-based translation further reinvigorated MT research. In these methodologies, henceforth referred to as SMT (statistical machine translation), the translation system attempts to find a sentence similar to the input sentence in a parallel corpus and make appropriate changes to the stored translation. Since then, statistical and corpus-based techniques have made the limitations of rule-based approaches more tractable. Still, on their own, neither statistical nor rule-based methods have proved to be the ultimate solution to the problem of translation, but they have improved output quality beyond what seemed attainable just a decade before.

The most recent addition to MT methods is NMT, neural machine translation. This technique aims to build a single neural network that can be jointly tuned to maximise the translation performance. A variety of approaches were initially proposed, but practically all recent work has been based on the attention-based encoder-decoder model of Bahdanau et al. (2015). NMT shows superior performance on public benchmarks and has been adopted by Google, Systran and others. However, NMT struggles in some important areas: especially important for this thesis is the demand for large amounts of training data, see Koehn and Knowles (2017). NMT is not ideal for translation tasks in languages with limited available resources, such as Norwegian, so SMT methods will be used instead.

[1] Automatic Language Processing Advisory Committee

2.2. Fundamentals of Statistical Machine Translation

In 2010, Koehn wrote that the currently best performing statistical machine translation systems were based on phrase-based models (Koehn, 2010, p.127). As mentioned above, today NMT is the best performing benchmark, but the methodology is not suitable for the task presented here. SMT is thus the best alternative and the particular variant of phrase-based translation (PBSMT) will be presented in some more detail.

Early models for machine translation were based on the translation of individual words. However, singular words may not be the best candidate for the smallest unit of translation. Sometimes a given word in a foreign language translates into two or more words in the target language, or even none at all. Word-based models break down in these cases.

To build an SMT system for a particular language pair, the first thing that is needed is a training corpus. This is usually in the form of a parallel corpus. The parallel texts are used in building the translation and reordering model, and the monolingual target side of the data is used by the language model to ensure fluent output.

Once training data has been collected, the text is tokenised. For English and other European languages, this typically involves converting the text to lowercase, and separating punctuation from words. This generalises the input: without tokenisation, the first word of a sentence would only be known in that capitalised form (assuming it did not occur elsewhere in the corpus), and it would be translated with this convention even when the word did not occur at the beginning of the sentence in the translated output. The equivalent holds for punctuation at the end of a sentence. Tokenisation is therefore necessary to allow a word to occur in other contexts than it did in the training data.

As mentioned above, the target side of the training corpus is used for building a language model. Language models (LMs) learn what is considered fluent output in the target language by giving these word orderings higher output probability. The most common LMs are built on n-gram models where the probability of n consecutive words is calculated based on the frequency of such sequences in the training corpus. N-gram models can be built on several factors (see Subsection 5.3.3), e.g., surface words (as they appear in the training corpus) or surface words and part-of-speech tags. The latter will encourage the translation system to output sentences on forms that are "grammatically correct" as far as this was expressed in the training sentences.

Next, there is the issue of alignment: which words correspond to which ones in the other language? This is done automatically with a variety of probabilistic models. This thesis uses GIZA++, Och and Ney (2003), which trains IBM Models 1-5 of Brown et al. (1993), and a Hidden Markov Model word alignment, Vogel et al. (1996).

After alignment, phrases are extracted and paired. Current phrase-based models have no deep linguistic notion of what exactly a phrase is. For example, an entry in a Norwegian-English phrase table could be '*moro med*' mapped to 'fun with the'. This is a rather unusual syntactic grouping in linguistics, where most syntactic theories would segment the sentence into the noun phrase 'fun' and the prepositional phrase 'with the [indirect object]'. However, learning the translation of '*moro med*' as 'fun with the' is very useful because prepositions often do not match very well between languages: *med* is for

instance translated with no preposition at all in *gratulerer med dagen* (happy birthday). Thus, learning this particular phrase ensures that different usages can be distinguished, and that similar sentence structures can be correctly translated (presumably). Learning translations at the phrase-level is much simpler than trying to map individual words into a varying number of different words.

The phrase extraction gives the phrase table, sometimes called the translation table. The pairs in this table are scored by having probabilities assigned to them. The score is given by relative frequency, i.e., the quotient of the frequency of the phrase pair and the sum of the frequency of the source phrase and all the words in the target phrase.

The syntactic orderings of words may vary between different languages. Reordering models are therefore needed in translation. Reordering is still an open research area, but in Chapter 5, the word-based extraction model by Koehn et al. (2005) is used, which determines the orientation of two phrases based on word alignments at training time.

Finally, during decoding, the translation system tries to find the best translation given the trained models. In Chapter 5, the Moses decoder, Koehn et al. (2007), will be used.

As for most machine learning systems, the final step in the process of training a system is the tuning. For SMT, this usually involves weighting the different models used in the decoder. In this thesis, no tuning was performed, so this step will not be discussed in any further detail here.

2.3. Practical Introduction to the Norwegian Language

An introduction to the Norwegian language and its peculiarities is useful for understanding which features must be paid close attention to when creating an English-Norwegian SMT system. Norwegian is a Germanic language spoken mainly in Norway, where it is the official language. Here, it has roughly 5.2 million native speakers[2]. The Swedish, Danish and Norwegian languages are closely related and native speakers can usually speak their own language and be mutually intelligible.

Besides the 26 letters used in English, Norwegian has three additional letters: *æ*, *ø*, and å. These letters are not supported by the standard ASCII format, and so an extended character set is needed to handle these. `UTF-8` is used in this thesis.

Norwegian has two written standards: *Bokmål* and *Nynorsk*. Both standards are official forms of the same language, but they differ in some vocabulary and morphosyntactic rules. In this report, unless stated otherwise, 'Norwegian' refers exclusively to Bokmål. This is because Bokmål is the most common variety, and the most material exists in this variant.

2.4. The Essentials of Norwegian Compounds

Norwegian is a language that uses closed compounds productively. This means that it is a language in which one can observe the phenomenon where new word formations

[2] `https://www.ethnologue.com/language/nor`

that are not found in any dictionary are still perfectly valid. Closed compounds are written as single words without spaces or other word boundaries, as in *gullring* (gold ring). Languages that make use of such formations will henceforth be referred to as compounding languages. These include, but are not limited to, many Germanic (e.g., German and Swedish) and Uralic languages (e.g., Finnish and Hungarian).

In English, on the other hand, compounds are generally open, i.e., written as two or more words, like 'gold ring', or coordinated with function words like 'of', 'for', etc.

Faarlund et al. (1997) present a comprehensive collection of Norwegian grammar, in which compounds play an important role. By their definitions, a compound has two parts. These parts, henceforth referred to as constituents, can recursively contain other compounds. The first constituent is called the modifier and adds some new meaning to the other part(s). The rightmost one, the head, (usually) determines grammatical features of the compound. For example, in *tyttebærsyltetøy* (lingonberry jam), the constituents are *tyttebær* and *syltetøy*, which are again composed of *tytte*, *bær*, *sylte*, and *tøy*. *Tytte* modifies the head *bær*, *sylte* modifies *tøy*, and *tyttebær* modifies *syltetøy*.

Despite the fact that *tytte* has no meaning on its own, and *tyttebær* thus cannot be split into two constituents found in a lexicon, Faarlund et al. argue that the similar structures found in other names of berries, e.g., *blåbær* (blueberry) and *jordbær* (strawberry), justify the claim that *tyttebær* is a valid compound.

The constituents of Norwegian compounds can be separated by one or more characters, usually for ease of pronunciation. Such characters are called epentheses. NST's lexical database, see Andersen (2005), lists the following epentheses:

- *s*: as in *morsrolle* [*mor-s-rolle*] (role of a mother)
- *e*: as in *barnehage* [*barn-e-hage*] (kindergarten)
- *n*: as in *rosenkål* [*rose-n-kål*] (Brussels sprout)
- *er*: as in *berlinerbolle* [*berlin-er-bolle*] (Berliner Pfannkuchen, pastry)
- *ar*: as in *laugardam* [*laug-ar-dam*] (a pond for bathing)
- *a*: as in *ferdafolk* [*ferd-a-folk*] (travelling folk)
- *me*: as in *lammekotelett* [*lam-me-kotelett*] (lamb chops)

However, epentheses make for a particularly tricky case in Norwegian morphology: the epentheses often coincide with inflected forms of the constituents, making it hard to distinguish inflected constituents from those that have a root form with an interposed epenthesis. Distinguishing an epenthesis from an inflected form can imply different semantics for the compound. For example, *melkemaskin* (milking machine) carries the meaning of a machine that milks cows only if it is split into *melke-maskin* ([to] milk machine). If the compound were to be split with an epenthetic *e*, the semantic interpretation would be that of a machine that produces milk or one that is made out of milk. The epenthesis can also give rise to a completely different semantic interpretation if the compound is not correctly split. Consider the following possible splits of *telefonsvarer* (telephone answering machine):

1. *telefon-svarer* (telephone answering machine)

2. **telefons-varer* (telephone's merchandise)
3. **telefon-s-varer* (telephone merchandise)

Johannessen (2001) proposes using stem member analysis[3] to deal with coinciding inflections, meaning that the constituents cannot contain inflected forms, and that the compound itself determines its grammatical features. Her argument for using this analysis is rooted in the theory that constituents do not have to be independent words. However, she argues differently than Faarlund et al. for this. Firstly, she points out that there are compounds that have different grammatical features from the head, for example *krypinn* (hiding place), which is composed of *kryp* (crawl) and *inn* (in), where *inn* is a preposition, but the full compound is a noun. Secondly, some constituents do not appear isolated, but only in compounds, like *tytte-* in *tyttebær* or *-øyd* in *treøyd* (having three eyes). Thus, without the need for constituents to be lexical words, stem member analysis allows analysing these parts. If each part had to be a valid lexeme, traditional syntactic and semantic analysis of *tyttebær* and *treøyd* would not be possible.

A consequence of stem member analysis is that the compound determines its own grammatical features. This can be seen by observing the following: whereas the inflection of a compound usually coincides with that of its head, this is not always the case. For example, *e-post* (email) is composed of *e* (abbreviation of *elektronisk*, electronic) and *post* (post, mail), but whereas *e-post* is countable (*en e-post, to e-poster*), *post* is not (**en post*, **to poster*).

It should be noted that compounds are not the same as words containing prefixes or suffixes. Faarlund et al. (1997, p. 90) categorise a derivation[4] as a word derived from another word using an affix that is attached to the latter one's root. An example is *umulig* (impossible), which consists of *u-* (not, un-, im-) and *mulig* (possible). Usually, dictionaries (those that mark compounds as such) make this distinction, and do not split based on affixes, so these should not pose a problem for the applications discussed in this thesis.

Johannessen and Hauglin (1998) also note a type of compound that contains a suppletive stem, for example *klesplagg* (piece of clothing). Like *tytte-*, *kles-* has no intrinsic meaning; *tytte* probably used to have a particular meaning in an older form of the language, but the semantics have since been lost on modern speakers. However, most native speakers would be able to identify *kles-* as etymologically related to *klede* (cloth), so it can be argued that this constituent has individual meaning and is a morpheme. Nevertheless, for the purpose of splitting compounds into constituents, the absence of the morpheme in a lexicon renders systems that use dictionaries as knowledge source unable to handle this kind of compounds.

[3] Called *stammeleddsanalyse* in Johannessen's Norwegian paper.
[4] *Avledning* in Norwegian.

3. Related Work

As will be discussed in Chapter 5, insufficient decompounding capabilities of the Norwegian part-of-speech tagger used gives rise to the need for a custom compound splitter. Compound splitting has already been studied in many other research projects, and Section 3.1 explores some of them. As this thesis focuses on translating and forming novel compounds, Section 3.2 investigates earlier work that has been done on this topic. The research focuses on statistical machine translation (SMT).

3.1. Literature Review of Related Work on Compound Splitting

Compound splitting is the process of identifying the separate constituents and epentheses (if any) in a compound. To determine implementation details of a compound splitter, a literature review of similar projects was carried out. This section details articles that have come up with effective schemes for splitting compounds, and/or projects that have focused on treating Norwegian compounds. The articles are presented in chronological order.

3.1.1. Johannessen and Hauglin (1998)

Most literature related to splitting Norwegian compounds is either rule based or makes use of statistical features. Perhaps the earliest work on computational splitting of Norwegian compounds is done by Johannessen and Hauglin (1998). They develop a module used in an automatic morphosyntactic tagger that can analyse Norwegian compounds. However, their focus lies on the linguistic clues that constrain compound analysis and not the computational aspects of the task. In finding all possible analyses of all compounds in a text, an incorrect analysis is reported in 1.1 per cent of the test cases. Johannessen and Hauglin attribute these errors to the fact that their analyser lacks certain stems used in the compounds of the test corpus, because these stems do not appear in the lexicon.

Johannessen and Hauglin's work is used in several theses trying to find solutions to the problem of compound splitting. Lindbråten (2015) uses two modules, one for finding potential splits of the compound, and one for finding the most likely correct interpretation of these splits. These modules make direct use of Johannessen and Hauglin's rules. The compound splitter makes correct splits in all test cases, whereas the interpreter makes the correct decision in 93.68 per cent and 85.36 per cent on an easy and a hard test set, respectively. The easy set contains no epentheses and typically has only two constituents, whereas the hard set contains ambiguous splits and epentheses. Lindbråten does not

deem this performance to be adequate. Errors made consist mainly of incorrectly splitting non-compounds (e.g., **et-ter*, **si-er*), and the fact that longer tails of the compounds are prioritised (e.g., **før-titall*, **fli-slegger*).

Thorsen Ranang (2010) introduces *Verto*, a module for automatic analysis of Norwegian compounds based on the linguistic principles presented by Johannessen and Hauglin (1998) and Johannessen (2001) with some alterations. Upon analysing the compounds presented in the first paper, he achieves a success rate of 87.5 per cent.

3.1.2. Koehn and Knight (2003)

Koehn and Knight (2003) introduce methods to learn splitting rules from monolingual and parallel corpora for machine translation purposes. Evaluated against a gold standard, they achieve 99.1 per cent accuracy, and performance gains for machine translation of 0.039 BLEU points, see Papineni et al. (2002), on translating noun phrases from German to English. For splitting, they try to find known words and fillers between words spanning the entire length of the compound. Known words are restricted to have at least length three.

Koehn and Knight create a baseline picking the splitting option with the highest number of splits made, which they try to improve in steps. A first attempt considers frequencies, and picks the split with the highest geometric mean of the word frequencies of its parts. Another approach uses a parallel corpus where, for each splitting option, the parts are checked for translations. This happens at sentence level, which leaves out potential splits like *Frei-tag* (literally 'free day') because only the full translation, Friday, and not the translations of the individual parts, appears in the parallel text.

Yet another method uses a second translation table. First, German words in the parallel corpus are split using the frequency method described above. Then, a translation lexicon is trained from the parallel corpus with split German and unchanged English. This generates a vast amount of splitting knowledge. Using this knowledge, an educated guess based on the most frequent option can be used when trying to split words in new texts.

Finally, a method imposing limits on POS (parts-of-speech) is used. Koehn and Knight observe that affixes are often split off and incorrectly translated because the corresponding English translation is very common (e.g., *den* and 'the'). The algorithm interprets this as an indication that this is likely to be the correct split. To prevent this eager splitting, POS information about the words is used: a compound is not allowed to be split into parts that are prepositions or determiners, only content words.

The most accurate method is the one using a parallel corpus and POS information, achieving a score of 99.1 per cent. As for word based statistical machine translation systems, the methods using splitting knowledge from a parallel corpus improve translation compared to unsplit data, but the frequency based splitting is the best (0.317 BLEU points, 0.011 points higher than the remaining methods).

In contrast to this again, in a phrase-based statistical machine translation system, the baseline method using maximal splits performs the best. Koehn and Knight find this somewhat surprising, seeing how this method achieves extremely low precision values

against the gold split of compounds. The frequency based method performs comparatively well to the maximal splitting.

3.1.3. Sjöbergh and Kann (2004)

Sjöbergh and Kann (2004) present a method for splitting Swedish compounds and several methods for choosing the correct interpretation of ambiguous compounds. They report that 99 per cent of all compounds in their test data are split, and that 97 per cent of these are interpreted correctly. They use a statistical approach, in which Swedish compounds are automatically split using a modified spellchecker. They also evaluate and compare several different methods for choosing the correct interpretation. All their evaluations are carried out on Swedish compounds, but they claim that all the methods (except for some language-specific ad hoc rules) should work on any language with similar properties. However, no material is provided to back up this claim, meaning it is unknown whether their methods can be applied to Norwegian.

3.1.4. Alfonseca et al. (2008)

Alfonseca et al. (2008) compare five splitting methods across seven languages (Dutch, Danish, German, Norwegian, Swedish, Greek, and Finnish). They show that a combination of these methods, previously applied to German, is also useful for other compounding languages. Furthermore, they find that models learned from a gold standard created for some language can be applied to other languages, sometimes producing better results than when a model is trained and tested in the same language. This implies that, to train a generic decompounder, it is not necessary to create a gold standard in more than a single language. It is argued that this compensates for the fact that the system is supervised. It also increases the availability of decompounders for smaller languages.

The work of Alfonseca et al. is partly motivated by the need to know whether the results reported for German are easy to reproduce in other languages. Indeed, the results are very similar across languages, having precision and recall values over 80 per cent for most languages. The supervised model gives much better results than the unsupervised ones, but it requires constructing a gold standard from which to train, which is costly. With this in mind, another experiment is run to see whether the models trained from some languages can be applicable to other ones as well. Alfonseca et al. comment that the results are "rather good" for all pairs of training and test languages, with only a few exceptions. For precision, training with either the Norwegian or the Finnish data produce very good results for most languages. These languages also have the best results in the monolingual experiments. Alfonseca et al. believe these trends are due to the quality of the training data, but unfortunately, they do not specify which training data they use.

In addition, Alfonseca et al. conclude that the size of the training data does not seem to play a large role in giving the best results.

3.1.5. Macherey et al. (2011)

Macherey et al. (2011) present an unsupervised method for learning compound constituents and morphological operations needed to split compounds into their constituents. The method uses a bilingual corpus to learn the operations, and a monolingual corpus to learn and filter constituent candidates. The approach is evaluated based on translating Danish, German, Norwegian, Swedish, Greek, Estonian, and Finnish texts into English. Several of these languages are from different language families, but they all benefit from compound splitting, which is taken to mean that the approach is highly versatile.

The goal of Macherey et al. is to design a language-independent compound splitter that is useful for machine translation. They want to automate the step of finding and extracting linking morphemes (epentheses) by using an unsupervised method that can provide these as additional knowledge to the decompounding algorithm. The most important knowledge source for their splitting algorithm is a word-frequency list of constituents. This list computes the split costs. The frequency lists can be extracted from monolingual data for which the language model word frequency lists are used, and the extraction is subdivided into two passes. The first pass, the Bootstrapping pass, generates word frequency lists derived from news articles for multiple languages. The phase outputs a table with preliminary compound constituents together with their respective counts for each language. The second pass, the Filtering pass, further reduces and filters the compound part vocabulary, where a language model vocabulary based on arbitrary web texts for each language is generated. Next, a compound splitter based on the vocabulary list that was generated in the previous phase is built. Every word of the web vocabulary can then be attempted split based on the compound splitter model from the Bootstrapping. For the constituents found in the compound splitter output, it is determined how often each one is used and the algorithm outputs only those constituents whose frequencies exceed a predefined threshold.

Since the authors want to use the algorithm in a machine translation setting, they measure their results using the BLEU evaluation score. For Norwegian, applying the splitting algorithm raises the BLEU score from 42.77 to 44.58. In comparison, Greek achieves the lowest improvement (0.06 points), and Danish the highest (1.84 points).

3.1.6. Clouet and Daille (2013)

Clouet and Daille (2013) present a compound splitting algorithm that combines language independent features (similarity measures and word frequencies in a corpus) with language dependent features (component boundary transformation rules). They apply this method to German and Russian, and find that it outperforms their baseline method of matching word components in a dictionary. Using transformation rules, but no corpus, they achieve 93.04 per cent precision in splitting. With rules and corpus they actually achieve a lower precision of 87.44 per cent, but in 95.51 per cent of the cases, the correct split was found amongst the algorithm's top five highest ranked splits.

Clouet and Daille thus conclude that using a specialised corpus allows correctly splitting some additional compounds including components that are unknown to the dictionary,

and that this somewhat compensates for the lack of transformation rules. On the other hand, using a higher number of rules gives better ranking of splits. The authors claim that their algorithm can be applied to other languages by changing the lexical sources and, optionally, editing the transformation rules, but like Sjöbergh and Kann (2004), no evidence to back up this claim is presented.

3.1.7. Stymne et al. (2013)

Stymne et al. (2013) propose several new methods for compound merging, based on heuristics and machine learning, which outperform previously suggested algorithms known at the time. They use a split-merge strategy, where splitting is done as a preprocessing step before training their statistical machine translation system. Then, they investigate several different splitting methods for this purpose. The schemes perform similarly on the metrics, and no clear difference in the error analysis is observed. Their conclusion is that translation is not very sensitive to the quality of the splitting strategy chosen, and they proceed to use their so-called *arith* method in their other experiments for merging. The method is only applied to words of minimum length six characters, split into parts of minimum three characters, by allowing all splits whose parts are found in a corpus and tagged as content words. The best splitting option (which can be no split) is chosen based on the arithmetic mean of the corpus frequencies of its parts. The compound head is also restricted to have the same POS tag as the full word.

3.1.8. Riedl and Biemann (2016)

Riedl and Biemann (2016) present a method they call *SECOS* (*SEmantic COmpound Splitter*), which is based on the hypothesis that compounds are similar to their constituting word units. The algorithm is built on a distributional thesaurus. The thesaurus is computed with a monolingual background corpus, and does not require any language-specific rules or preprocessing.

Riedl and Biemann compare the performance of *SECOS* against that of an unsupervised baseline and knowledge-based systems. Tested on 700 German nouns, *SECOS* achieves the highest precision, recall and F1-measure. For another data set containing some 54,000 German nouns, the method significantly outperforms its competitors. It does obtain a slightly lower recall than two other methods, but it still surpasses unsupervised baselines and yields the highest overall precision.

SECOS is freely available[1], and precomputed models are provided for several languages, including Norwegian. The models have been computed using Wikipedia[2], and similarities are computed with either JoBimText[3] or Word2Vec of Mikolov et al. (2013).

[1] `https://github.com/riedlma/SECOS`
[2] `https://www.wikipedia.org/`
[3] `http://ltmaggie.informatik.uni-hamburg.de/jobimtext/`

3.2. Related Work on Statistical Machine Translation

The Research Goal of this thesis focuses on domain-specific translation and production of compounds. As the articles presented in this chapter will show, the general consensus in the MT (machine translation) research community is that when working with translation tasks in a particular domain, it is beneficial to use techniques that are adjusted for this setting. Using such specialised techniques is called domain adaption. Considering Research Question 1, the topic of technical translation is of particular interest here. Furthermore, since this thesis aims to translate into Norwegian—a productively compounding language—articles that look into how to correctly form compound terms and generate novel compounds are also of interest. The comparatively limited textual resources is also a topic that warrants attention.

3.2.1. Claveau (2008)

Whereas domain adaption is a well-explored research field, these efforts are often aimed at full-text translation as opposed to directly translating single terms. Claveau (2008) is one of few that work on individual terms: he presents a fully automatic system for translating biomedical terms between a variety of language pairs. New or unknown terms can be translated after learning regular patterns found in expressions in this domain. Such rules can be learnt by exploiting the fact that, in many languages, biomedical terms usually share a common Greek or Latin basis, and that their subsequent morphological derivations are very regular. However, this is a trait that is only stated for this particular domain, and it is thus not generally applicable.

3.2.2. Niehues and Waibel (2011)

Niehues and Waibel (2011) are concerned with translating university lectures on the topic of computer science from German into English. To acquire translations for terms specific to this domain, they fetch such translations from Wikipedia[4]. The 'interlanguage' links, i.e., links from a Wikipedia page in one language to the equivalent page in another language, are used for this purpose. The vocabulary extension is also successfully combined with genre adaption, in which a parallel corpus is built from a TED corpus (made of the subtitles and translations of the talks on the TED Website[5]). Niehues and Waibel reduce the number of OOV (Out-Of-Vocabulary) words by up 50 per cent, and improve the BLEU scores. Additional improvements are achieved by learning rules for different morphological transformations in the source and target languages. However, the setup presented does not consider the context of the target language, so morphologically rich languages are only handled as input language, and it is not known how well the approach deals with compound generation and related problems.

[4] `https://www.wikipedia.org/`
[5] `https://www.ted.com/`

3.2.3. Arcan et al. (2012)

Arcan et al. (2012) investigate term translation between English and German restricted to domain-specific vocabulary. They find that a domain-specific resource produces better results than bigger, more general ones. The terms of a financial ontology are used as starting point, and a cross-lingual terminological lexicon is built from Wikipedia[6]. The entries of the ontology are looked up on Wikipedia, and then additional information—like the article title, the categories the article is classified under, and the interlanguage links—is used to extract other relevant articles to build a domain-specific lexicon. Their experimental results outperform Google Translate[7] on some evaluation metrics, implying that hybrid translation systems, combining bilingual terminological resources and SMT, can indeed improve translation of domain-specific terms. It should be noted that at the time Arcan et al. conducted this study, Google Translate was a phrase-based SMT system. In 2016, Google made a turn for the NMT (neural machine translation) methodology, and it is therefore not guaranteed that the above results hold anymore. See Wu et al. (2016) for details on Google's NMT system.

3.2.4. Stymne et al. (2013)

Traditional SMT systems can only translate words as they have occurred in the training sets. The fact that productive compounds cause unseen words therefore poses a challenge for these systems, but listing all possible compounds is not a viable option. Even if assuming that all constituents were known (which is not always the case), listing all possible combinations would cause the number of entries in a translation table to grow quadratically with the number of possible constituents. Furthermore, this estimate disregards the use of epentheses as well as the fact that compounds may contain more than two parts, making an extensive translation table even more intractable.

The most successful works treating compounds do not use an exhaustive list of compounds, but rather a split-and-merge strategy where compounds are split prior to translation, translated as separate words and then merged back together in a post-processing step. This approach is explored by for instance Nießen and Ney (2000), Koehn and Knight (2003), Popovic and Ney (2006), and Holmqvist et al. (2007).

Stymne et al. (2013) was already discussed in Subsection 3.1.7, but only in terms of the split-merge strategy they use on compounds. The paper also discusses the effect of compound splitting on statistical machine translation, so this aspect is worth examining in isolation. In their experiments, Stymne et al. (2013, pp.1085–1104) give several approaches to producing novel compounds; i.e., compounds not observed in training. To treat compounds differently from normal lexicalised words, they experiment with different part-of-speech markups for modifiers in the form of modified POS tagsets. Their experiments yield good results when using an extended tagset (containing additional tags for modifiers that indicate the POS of the compound head) in sequence models. However, on a smaller domain-specific corpus, a restricted tagset (distinguishing only modifiers,

[6] `https://www.wikipedia.org/`
[7] `https://translate.google.com/`

compound heads, and nouns from all other tokens) works well. After translating the split compounds, a sequence labeller is used for merging the constituents back together.

3.2.5. Cap et al. (2014)

Cap et al. (2014) make translations into a compounding language by splitting compounds into the corresponding constituents for training. Their work builds on Stymne et al., but they also project features from the source language to support compound merging predictions, and reduce compound parts to an underspecified representation which allows for maximal generalisation. This approach is able to produce multiple correct compounds that were unseen in the training data. Unlike Stymne et al., the underspecified generalisation enables correctly conjugating words that might only have been seen in one grammatical case in the training data into their proper form in the output translation.

3.2.6. Gujral et al. (2016)

Norwegian is not as well represented as other languages in terms of readily available parallel corpora. Gujral et al. (2016) present a system that applies a combination of language-independent techniques to translate Out-Of-Vocabulary (OOV) words for languages lacking extensive parallel corpora. They use the techniques Levenshtein Distance, Word Embeddings and Transliterations. Levenshtein Distance looks for morphological or spelling variants of the unknown word in the training data, and the Word Embeddings try to find candidates that are closely related in a semantic vector projection. These methods thus pick the best candidate among previously known words, and do not produce any novel terms. The transliteration method does produce unseen words by learning a model for word pairs that are transliterations of each other. However, these alignments happen at character level, and are most appropriate for translating named entities and loanwords from English, as opposed to completely novel terms or productive compounds.

4. Data Sets and Knowledge Sources

All natural language processing (NLP) systems require some kind of knowledge source. For translation systems, this typically involves parallel corpora. In addition, the custom compound splitter requires word lists with compounds and their constituents to determine what splits to make. Here follows a discussion of such available tools and corpora along with an evaluation of their suitability for the two tasks at hand. The evaluation of the texts constitutes examining the size of each resource, manually inspecting a subset of the contents, and performing initial experiments if the two previous qualities are promising. Further details are supplied in each subsection. The evaluation concludes with two training sets and two test sets presented in Section 4.3 and Section 4.4, respectively.

4.1. Knowledge Sources for a Compound Splitter

This section details how to provide a custom compound splitter with the information it needs to be able to identify potential splits.

4.1.1. Toolkits and Projects Concerning Compound Splitting

Toolkits for Compound Splitting

The Natural Language Toolkit (NLTK), see Bird et al. (2009), is a collection of Python modules, data sets, and more that supports research and development in NLP. The implementation of the splitters described in Section 5.1 will make use of this toolkit's built-in language processing features. NLTK includes a wide selection of texts and different types of corpora, but a limited number of these contain Norwegian texts. Furthermore, these do not provide any additional information about compounds.

NLP Norway maintains a comprehensive list of NLP resources for Norwegian[1], but neither of the resources listed support marking compounds.

Compound Splitting Projects

Several potential data sources are considered, among them the MORBO/COMP project of Guevara et al. (2006); a research project aiming to be the first empirical database on which studies on compounding are to be possible. It is supposed to contain data

[1] `https://github.com/web64/norwegian-nlp-resources`

on descriptions of different types of compounds, structural complexity, presence and typology of linking elements, plural formations, and distribution of different structures. The project encompasses more than 20 languages, including Norwegian, and presents itself as a very valuable data source. Unfortunately, the project seems to have stalled since the article describing it was published, and attempts at contacting the development team or otherwise acquiring the data sources failed.

Many of the previously mentioned papers use the Europarl Parallel Corpus, see Koehn (2005), but this corpus is not available in Norwegian. This is because the Europarl corpus is extracted from the proceedings of the European Parliament, which Norway is not a member of. The Europarl corpus is available in 21 of the 24 official European languages (excluding Irish, Maltese, and Croatian).

4.1.2. Corpora and Lexica for a Compound Splitter

A corpus, a collection of text, is the most common type of knowledge source used to build translation and language models. Corpora can be organised by theme, as monolingual or multilingual dictionaries, or as frequency lists.

As pointed out by Stymne et al. (2013), to create new unseen compounds, knowledge about compound formation, specifically their constituents, is needed. This subsection explores available resources which may contain this kind of information.

Bokmålsordboka

Bokmålsordboka, Kjelsvik (2016), is probably the most popular monolingual Norwegian dictionary. Its web edition shows current orthography, morphology, senses, usage examples and brief etymologies. The University of Bergen and The Norwegian Language Council (*Språkrådet*) own and run the dictionary portal. The Language Council is the government agency for language issues, and carries out official Norwegian language policy on behalf of the Ministry of Culture. The Language Council is responsible for maintaining the official orthographies of Bokmål and Nynorsk, which makes it a very reliable and trustworthy source. The online edition[2] marks the split of a compound with a vertical bar, '|', and thus seems like a very promising data source. However, there is no way to automatically extract this data. In addition, the dictionary only splits compounds once, even when the compound is composed of more than two constituents or has epentheses. For instance, *andregradsligning* (quadratic equation) is marked as *andregrads|ligning*; to be useful for this project, it would have to be split into three constituents in addition to an epenthetic *s*: *andre-grad-s-ligning*.

Norsk Ordbank

Provided by Språkrådet and University of Bergen, *Norsk Ordbank* is a database of lemmata and inflection patterns. It provides the basis for *Bokmålsordboka*'s online search

[2] `https://ordbok.uib.no/`

functionality. The word bank has been accessible under the *Creative_Commons-BY (CC-BY)*[3] licence since 2011 and can be downloaded for free. One of the files in the download is `leddanalyse`, which contains information about compounds, including modifiers, heads, epentheses (if any) and POS (parts-of-speech) of their constituents. Constituents do not have to be lexicalised words; e.g., *tytte* is listed as the head of *tyttebær*. However, a major disadvantage of this database is that affixes are marked in the same way as constituents, making it hard to distinguish these when extracting compounds.

The Oslo-Bergen Tagger

The most popular Norwegian tagger is the Oslo-Bergen Tagger of Johannessen et al. (2012), henceforth referred to as OBT. OBT is a morphological and syntactic tagger developed at the University of Oslo and at Uni Computing in Bergen. The tagger consists of three main modules: a preprocessor (composed of a multitagger and a compound analyser), a grammar module for morphological and syntactic disambiguation, and a statistical module that resolves any remaining morphological ambiguity. The multitagger uses *Norsk Ordbank* as lexicon. The tagger's compound analyser can mark compounds as such, using the tag `samset`, but provides no information about the relevant constituents.

The Norwegian Dependency Treebank

The Norwegian Dependency Treebank (NDT), Solberg et al. (2014), is a syntactic treebank for Norwegian with manual syntactic and morphological annotation, developed at the National Library of Norway in collaboration with the University of Oslo. The text is annotated with morphological features, syntactic functions and hierarchical structure. Tagged by OBT, it marks compounds with `samset`, but contains no information about constituents or epentheses. Whereas it is possible to extract the compounds and manually annotate the constituents here, this would be very costly in terms of time and effort. The process would also be error prone, so this approach is not viable.

NoWaC

The Norwegian Web as Corpus, Guevara (2010), is a web-based corpus of Norwegian with about 700 million tokens. Seeing as the tokens are tagged using OBT, the corpus marks compounds but not their constituents, and it is thus not very useful in this context.

HaBiT

'Harvesting Big Text data for under-resourced languages'[4] is a collaboration between Masarykova univerzita in Brno, NTNU in Trondheim, the Text Laboratory from the University of Oslo, Addis Ababa University, and Hawassa University. It is a collection of corpora from the Web for under-resourced languages, among them Norwegian Bokmål

[3] `https://creativecommons.org/licenses/by/4.0/`
[4] `http://habit-project.eu/`

and Nynorsk. For Bokmål, text was crawled from web domains in the national top level domain (`.no`) and some other general domains (`.com`, `.org`, `.net`, etc.). After collection, the data was processed and annotated for use in NLP tasks. The Bokmål corpus has more than 1.3 billion tokens, but as for many of the previously mentioned resources, the texts are tagged with OBT, which gives no special insight into compound formation.

The Oslo Corpus of Tagged Norwegian

The Oslo Corpus of Tagged Norwegian Texts[5] contains 18.5 million words taken from newspapers, magazines, novels and public documents. Again, tagged with OBT, compounds are marked but not decomposed. The corpus has an online search interface[6], but unfortunately, no API (application programming interface) or other automatic extraction tools, and no way to retrieve the constituents. However, it does have a RegEx (regular expression) search interface which proved very useful in finding ambiguous compounds that could be used in the test sets described in Subsection 4.4.1.

The NST Lexical Database

The NST[7] Lexical Database for Norwegian Bokmål, Andersen (2005), is a lexical database of more than 700,000 full form Norwegian lexemes. It was originally developed for use in speech technology which requires such a full-fledged lexicon. It contains vast amounts of information on orthography, pronunciation data, and decompositions of compounds. Relevant information found in this database includes compound lexemes, the part-of-speech of such compounds, decomposition details (constituents, including epentheses, if any), and parts-of-speech of the individual constituents. NST allows the epentheses *s*, *e*, *n*, *er*, *ar*, *a*, and *me*. A few disadvantages of the database were discovered and are commented on in Subsection 4.3.2; nevertheless, it remains the best knowledge source for compounds and their constituents for the purposes of this project.

4.2. Knowledge Sources and Data Sets for a Translation System

SMT (statistical machine translation) systems rely on corpora with equivalent content in two languages to train a translation model. The best models are trained on parallel corpora, i.e., collections of text, paired with exact translations in the other language. If parallel corpora are unavailable, one may resort to comparable corpora, says Koehn (2010, p. 55). Comparable corpora are collections of texts that contain similar content in the same domain, but that are not direct translations of each other. For instance, Wikipedia uses language links[8] to interconnect pages on the same topic, but the exact

[5] `http://www.tekstlab.uio.no/norsk/bokmaal/english.html`
[6] `https://tekstlab.uio.no/norsk/korpus/bokmaal/netscape/treord/oktntb.shtml`
[7] *Nordisk språkteknologi holding AS*, Nordic Language Technology Holding
[8] `https://en.wikipedia.org/wiki/Help:Interlanguage_links`

content of the page in each language may vary. Learning from comparable corpora has been shown to be effective to some degree, for instance by taking advantage of the fact that a word that occurs in a particular context has a translation that occurs in a similar (translated) context. However, if available, parallel text remains a much more powerful tool.

How large does the corpus size need to be to achieve satisfactory translation quality? In this thesis, Moses, see Koehn et al. (2007), a free SMT decoder, will be used. In Moses' Baseline System demonstration[9], the French-English parallel news corpus used has almost 4,000,000 tokens on the French side, and 3,500,000 on the English side. However, it is supported by multiple studies that the size of the parallel corpus in SMT tasks is not as important as the quality of the text, see for instance Yıldız et al. (2014) and Gavrila and Vertan (2011). Specifically, Gavrila and Vertan conclude that "for technical domains a small, manually corrected corpus can be successfully used for obtaining a reasonable translation output". The training sets used in their studies counted as few as 27,889 tokens. The goal of this subsection is therefore to identify parallel high-quality English-Norwegian corpora with in-domain vocabulary, of the largest size possible.

4.2.1. The English-Norwegian Parallel Corpus

The English-Norwegian Parallel Corpus, ENPC, see Oksefjell (1999), consists of Norwegian texts and their English translations. It is intended as a general research tool, available for applied and theoretical linguistic research. The corpus was completed in 1997 and contains 50 "novels and fairly general non-fictional books", resulting in some 600,000 words in each language. The language in the corpus may thus be outdated, and is very unlikely to contain modern terms used in the computer science community.

4.2.2. LOGON

The LOGON parallel tourist corpus, Lønning et al. (2004), consists of Norwegian texts from several sources paired with English translations. All texts are from the tourist domain, and some are specifically from the hiking domain. The corpus consists of 180,000 words in each language, but like ENPC, it probably does not contain computer science vocabulary.

4.2.3. OPUS

The OPUS project, Tiedemann and Nygaard (2004), is an open parallel corpus spanning a range of languages. To provide the community with a publicly available parallel corpus, the project converts and aligns free online data with added linguistic annotation. Searching the OPUS resources for parallel texts in English and Norwegian, 9 corpora are listed, of which the largest one is `OpenSubtitles v2018`, which contains 9.4 million sentences, 77.3 million English tokens, and 62.7 million Norwegian tokens. This is a

[9] `http://www.statmt.org/moses/?n=Moses.Baseline`

collection of translated movie subtitles from OpenSubtitles[10]. These numbers constitute the largest known parallel English-Norwegian corpus, but manual inspection showed that the texts contain large portions of short, spoken language, meaning lower-quality material for training SMT systems. Furthermore, since the project is open sourced, most of the subtitles are user-supplied, resulting in texts that are often misspelled or grammatically incorrect.

4.2.4. TED

TED[11], an acronym for technology, entertainment, and design, is a nonprofit organisation whose mission is to spread ideas. The organisation is popularly known for their short, powerful talks published as freely available videos online. Their talks cover a wide range of topics—from science to business and global issues—in more than 100 languages.

Many of the videos have subtitles in a variety of languages and, in October 2019, 271 translated transcripts were available in Norwegian. All of the above-mentioned talks are held in English, and a transcript of the original talk is also available. Transcripts and subtitles may be used under the CC BY-NC-ND 4.0 license (Creative Commons license, Attribution-Non Commercial-No Derivatives 4.0 International) in conjunction with the TED Talk video, and may thus be collected and used as a parallel corpus in academic work.

As one of the most popular topics of TED Talks is technology, these talks represent a potentially valuable source for domain specific vocabulary. However, it is important to note that these translations are supplied by volunteer translators, and that the transcripts are not vetted. This affects the quality of the corpus. For the purpose of initial experiments, the transcripts of videos that were tagged with the technology topic and that had an available Norwegian transcript were scraped and agglomerated into two parallel text files with English and Norwegian transcripts.

The contents of the transcripts are not separated at the sentence level, but in suitable chunks of what the speaker is saying at that particular time. The transcripts were therefore concatenated, and split into separate lines on occurrences of full stop (“.”), exclamation mark (“!”), or question mark (“?”) followed by a space. This approach was successful in separating full sentences, but incorrectly split sentences containing abbreviations. Abbreviations are composed of a full stop followed by a space, thus indistinguishable from the end of a sentence without further preprocessing. Initial experiments proved that this made for a corpus that could not be sentence aligned without manual intervention, and the idea of using these transcripts was therefore abandoned.

4.2.5. The ELRC-SHARE Repository

The ELRC-SHARE repository[12] is used for documenting, storing, browsing and accessing Language Resources that are collected through the European Language Resource Co-

[10] http://www.opensubtitles.org/
[11] https://www.ted.com/about/our-organization
[12] https://elrc-share.eu/

ordination. The website provides a search interface in which one can search for parallel corpora in a number of languages. For Norwegian and English, the largest corpora found is the "Translation memories from The Ministry of Foreign Affairs of Norway"[*sic*] with 733,081 "Translation Units". In terms of size, this corpus is followed by "Medicines descriptions in English and Norwegian from the European Medicines Agency"[*sic*] with 347,414 units and the "Bilingual English-Norwegian parallel corpus from The Norway's Governments website"[*sic*] with 340,840 units.

The translation memories contain Norwegian translations of parts of the European Union (EU) law and obligations, Acquis Communautaire. The translations are provided by the Ministry of Foreign Affairs' Unit of EEA Translation[13] and are thus guaranteed to be high-quality. The corpus is organised into 19 files, of which the file `nb_no_telekommunikasjon.tmx` (telecommunications) is of particular interest. Using these files does require a certain amount of text cleaning, but initial experiments showed that this process could be automatically scripted, thus not posing a large challenge. Details on the cleaning are described in Subsection 5.3.1.

4.3. Training Sets

This section describes what training sets will be used by the systems implemented in Chapter 5.

4.3.1. Training Set for the Compound Splitters

Based on the above discussion, the NST corpus (Section 4.1.2) emerges as the data source that provides the most information about compounds and their constituents. A corpus based on the NST database is constructed using the NLTK `CorpusReader` class. The method used to do so is described by Bjerke-Lindstrøm (2017, Ch. 6). This generated corpus contains all entries found in the NST database (both compounds and simple lexemes), and their corresponding POS tag. An additional text file, extracting only the compounds and their constituents from the NST database is also constructed. These two resources provide the language knowledge of the implemented splitters.

The constructed corpus has 723,001 entries, whereof 325,816 are compounds. Table 4.1 shows the 10 most common classes and their distribution in the corpus.

4.3.2. Errors and Inconsistencies in the NST Corpus

The NST database is not consistent in its approach to splitting compounds. For example, the lexeme *tyttebær* is never split in any of its forms. In contrast, *Norsk Ordbank* consequently splits it into *tytte-bær*. On the other hand, in NST, *blåbær* is usually marked as a simple noun, except for its plural definite form which has been split into *blå-bærene*. There is no obvious reason as to why this inflection has been treated differently.

[13] *Enhet for EØS-oversettelse ved Utenriksdepartementet*

Part-of-speech combination	Number in NST	Percentage in NST	Percentage of compounds
Noun	212,259	29.36	-
Noun-noun	155,578	21.52	47.75
Verb	93,160	12.89	-
Adjective	90,502	12.52	-
Noun-epenthesis-noun	65,636	9.07	20.15
Verb-noun	17,910	2.50	5.50
Adjective-noun	17,757	2.50	5.45
Noun-adjective	10,918	1.51	3.35
Adjective-adjective	6,179	0.85	1.90
Noun-noun-noun	4,861	0.67	1.49

Table 4.1.: Distribution of POS combinations in NST.

The inconsistency is also seen in the treatment of words that have a constituent that is never used in isolation; e.g., *-snutet* in *storsnutet* (having a big nose) or *hoved-* in *hovedperson* (main character). *Storsnutet* is not split in the neuter entry of the adjective, but it is indeed split in the masculine/feminine entry. In contrast, *Hoved-* never appears as a constituent in NST, but *Norsk Ordbank* consistently treats it as a modifier.

Furthermore, some part-of-speech (POS) tags are clearly erroneously marked. For example, *elkraft* is split into *el-kraft*, but the constituents are tagged as *'+NN'*. This supposedly indicates that the modifier has no POS while the head is a noun. This could be attributed to the fact that *el* is an abbreviation of *elektronisk* (electronic), and that the database does not have a way of handling these morphemes. However, there are other errors, like *sydvesten*, whose constituents are marked simply as '+'. The error is thus more likely to be a human one. In all probability, the above list is not exhaustive, and there may be yet other types of errors present in the database. Regardless of the error cause, this introduces noise in the compound splitters' data source.

4.3.3. Training Set for the Machine Translation System

For the translation system, the ELRC-SHARE repository's "Translation memories from The Ministry of Foreign Affairs of Norway" is the corpus with the highest quality and the most in-domain vocabulary available. It is available in the TMX[14] format, an XML format where both languages are represented in the same file, paired in `tu` (Translation Unit) and `tuv` (Translation Unit Variant) tags. This format is well suited for extraction of a parallel corpus. Training the translation model requires one file for each language, in which each line corresponds to the equivalent line in the other language. A script for automatically constructing these files is implemented. In addition, some text cleaning is done, stripping unnecessary whitespace characters and removing markup tags.

[14]Translation Memory eXchange

As mentioned in Subsection 4.2.5, the parallel corpus consists of 19 different files. The custom tagger implemented in Chapter 5 is unoptimised and very slow, especially for longer compounds. Thus, for the tagging to be tractable, it is decided to use only a subset of these 19 texts in the training of the translation system. These are:

- `nb_no_eu_utvidelser.tmx` (3903 translation units),
- `nb_no_off_innkjop.tmx` (17614 units),
- `nb_no_prodcom.tmx` (13168 units),
- `nb_no_sanksjonsforordninger.tmx` (490 units),
- `nb_no_standardtekst.tmx` (286 units),
- `nb_no_telekommunikasjon.tmx` (10039 units), and
- `nb_no_ymse.tmx` (11825 units).

Manual inspection of the translation memories showed that the files `nb_no_telekommunikasjon.tmx` and `nb_no_off_innkjop.tmx` contain the most vocabulary related to the computer domain. The remaining files are included because of their limited size, and serve as high-quality general language text from which the Norwegian language model is built. The seven listed files are concatenated and represent the training corpus.

4.4. Test Sets

This section describes the test sets that are to be used during the experiments in Chapter 6. For the compound splitting, some important points concerning the evaluation of the experiments are also highlighted.

4.4.1. Test Sets and Evaluation for the Compound Splitters

The test sets for the evaluation of the compound splitters were compiled manually by the author of this thesis (who is a native Norwegian speaker). This excludes the sets *Johannessen* and *Representative*, which are based on specific sources (see below). All the test sets come in the form of a gold standard where the compound is listed in a tab separated file along with its correct interpretation. To test the splitters' ability to handle ambiguous compounds, all the test sets were constructed with ambiguity in mind. Compounds with only one possible split are trivial for any splitter to interpret, so these provide little to no insight as for how the splitter interprets ambiguous lexemes. The test set must also be distinct from the training set, seeing as it is the splitter's ability to handle unseen compounds that is evaluated.

Gathering such unknown ambiguous compounds proved a laborious task that required a large amount of time and manual effort. The sizes of the test sets are therefore very limited. However, the fact that the test sets were constructed with quality and not quantity in mind makes the results of the experiments more valuable than if the splitters were to be tested on a more "typical" selection of compounds. To verify this, a test set of "randomly" selected compounds was also created (*Representative*).

Here follows a brief description of the test sets that will be used in the splitter experiments in Section 6.1. Notice that a sample may appear in more than one test set. Consider the compound *koppe-vaksine*: the sample has no epenthesis, so it appears in the test set *No epenthesis*, but the *e* at the end of the first constituent is a part of the modifier (and not an epenthetic *e*), so the sample also appears in *Lexicalised e*. The complete list of ambiguous compounds are found in Appendix A.

- *Ambiguous*: 118 manually compiled ambiguous compounds. The ambiguity may be due to splitting into wrong constituents, mistaking epentheses for lexical compounding or vice versa. All compounds that appear in one of the other test sets are ambiguous, so they are also included in this set.
- *Johannessen*: 37 ambiguous compounds, based on the examples given in Johannessen and Hauglin (1998).
- *Epenthetic e*: 8 manually compiled ambiguous compounds, where the ambiguity should be interpreted as epenthetic *e*; e.g., *boks-e-kjøtt* (tinned meat), not **bokse-kjøtt* (boxing meat).
- *Epenthetic s*: 50 manually compiled ambiguous compounds, where the ambiguity should be interpreted as epenthetic *s*; e.g., *drap-s-alarmen* (attack alarm), not **draps-alarmen* (murder's alarm) or **drap-sal-armen* (murder saddle [the] arm).
- *Lexicalised e*: 17 manually compiled ambiguous compounds, where the ambiguity should be interpreted as part of one of the constituents; e.g., *bokse-talent* (boxing talent), not **boks-e-talent* (box talent).
- *Lexicalised s*: 16 manually compiled ambiguous compounds, where the ambiguity should be interpreted as part of one of the constituents; e.g., *bok-selger* (book seller), not **bok-s-elger* (book moose [plural]) or **boks-elger* (box moose [plural]).
- *No epenthesis*: 56 manually compiled ambiguous compounds, where the ambiguity should be interpreted as two constituents with no epenthesis; e.g., *banekapasiteten* ([the] capacity [of a] [sports] field), not **ban-e-kapasiteten* (clear [imperative] capacity).
- *No split*: 16 manually compiled lexemes that can be interpreted as compounds but are single lexemes; e.g., *morsommere* (funnier), not **mor-sommere* (mother summers).
- *Representative*: 203 compounds manually compiled by extracting the first 500 compounds returned from Oslo-korpuset's web interface[15]. The quality of being a compound is determined by the Text Laboratory's multitagger[16], and obvious errors, like *euroen* (the Euro), were left out. Coordinated compounds, e.g., *hotell- og feriekonge* (the king of hotels and holidays), were shortened to the last constituent. Furthermore, compounds containing hyphens, abbreviations and proper nouns were omitted because these are not handled by the splitters. Finally, all entries were transformed to lowercase and duplicates were removed. This left a test set where 53 of the compounds had already been observed in the corpus and 150 were unknown.

[15] `https://tekstlab.uio.no/norsk/korpus/bokmaal/netscape/treord/oktntb.shtml`
[16] `http://www.tekstlab.uio.no/norsk/bokmaal/english.html#tagger`

4.4.2. Evaluation Metrics of the Compound Splitters

This section presents the performance metrics used in Subsection 6.1.3. These are important to discuss because the metrics (accuracy, precision, recall, and F_1) do not apply in the same way to all the splitters. The following definitions were used in the evaluations:

- **Class**: compounds are classified by the evaluation of the question "should this compound be split?".
- **True positive** (TP): true positives correctly answer "yes" to the question "split this compound?"; i.e., these compounds are split correctly, at the correct location.
- **False positive** (FP): false positives incorrectly answer "yes" to the question "split this compound?"; i.e., these compounds are split when they should not have been split. The category was extended to also comprise compounds that are split at the incorrect location.
- **True negative** (TN): true negatives correctly answer "no" to the question "split this compound?"; i.e., these lexemes are correctly not split.
- **False negative** (FN): false negatives incorrectly answer "no" to the question "split this compound?"; i.e., these compounds are incorrectly not split when they should have been split.
- **Accuracy**: Accuracy measures the ratio of correctly split compounds to the total number of lexemes (both compounds and not).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- **Precision**: Precision is the ratio of correctly split compounds to the total number of compounds split by the splitter. High precision relates to a low false positive rate.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: Recall is the ratio of correctly split compounds to all compounds that should have been split. Recall above 0.5 is generally considered adequate.

$$Recall = \frac{TP}{TP + FN}$$

- F_1: The F_1 score is the harmonic mean of precision and recall, which considers both false positives and false negatives.

$$F_1 = \frac{2 * Recall * Precision}{Recall + Precision}$$

One of the test sets, *Not compounds*, uses a different metric. If the splitter performs well, it will not split the compounds in this test set, which will give a high number of true negatives, but no true positives. All performance metrics, except for accuracy, will then

be 0, so the evaluation class is modified. Here, the class is "should this compound not be split?" or equivalently "should this compound be kept as a whole?". Now, true positives are those samples that correctly answer "yes" to this question; i.e., these compounds are correctly not split when they should be kept whole. False positives are those compounds that are not split when they should be split. True negatives correctly answer "no" to the question of whether they should not be split; i.e., these compounds are split when they should be. Finally, false negatives are those compounds that are split when they should not have be split, or they are split in the wrong position.

SECOS, see Subsection 3.1.8, has a built-in evaluation tool. It is unclear how this tool calculates the values it presents; as can be seen in Table 6.1c, a precision of 1.000000 is reported despite having no correct splits in the *Epenthetic e* test set. However, the inner workings of *SECOS* is beyond the scope of this thesis, so the implementation is left as-is, and the evaluation results are used as reported by the tool. Accuracy is the only metric that is seemingly calculated in the same way, which is to be kept in mind when comparing against the other results.

4.4.3. Test Sets for the Machine Translation System

The test set for the translation system is made up of the translations that native speakers will rate in Section 6.2. It was constructed by pulling page titles from Wikipedia from the subcategory section on the Computer Programming Portal[17]. This yielded 2323 pages. Page titles with proper names (e.g., 'Alexandra Cardenas' and 'Codecademy'), abbreviations and acronyms (e.g., 'SWIG' and 'DLL injection') were manually omitted. Words with foreign symbols and/or numbers (e.g., 'Эль-76' and '0x10c') were also left out. In addition, page titles prefixed with 'Category:' or 'Template:' were normalised, and disambiguating specifications in parentheses, e.g., 'Map (parallel pattern)', were pruned. This left 1239 page titles, of which 94 had a language link to a Norwegian page (that was not simply the same title). These titles were interpreted as good translations, due to being supplied by humans and being open for vetting. Out of the English page titles without translations, 100 were randomly selected for translation and evaluation. The English side of the training corpus contained 16 of these terms.

[17] `https://en.wikipedia.org/wiki/Portal:Computer_programming`

5. Architecture

This chapter discusses the architecture of the custom compound splitter that is used to extend the Norwegian part-of-speech tagging, as well as the implementation of the translation system. Section 5.1 describes five different splitting approaches which will be evaluated in Section 7.1. Section 5.2 makes a choice of which splitter to implement in the translation pipeline based on the results and subsequent discussion in Section 8.1. Finally, Section 5.3 describes the architecture of the translation system as a whole.

5.1. Architecture of the Splitters

The need for a custom compound splitter is a result of OBT's (the Oslo-Bergen Tagger's) lack of a decompounding capability. In Section 4.1.2, it was noted that OBT does have a compound analyser which can mark compounds with the tag `samset`, but this does not provide enough information to train a system on compound parts. Since few of the articles discussed in Section 3.1 are applied to Norwegian, it is necessary to investigate whether the methods achieve comparable results for this language. Hence, experiments with some of the above algorithms will be carried out.

The articles discussed can be roughly classified as either rule-based or machine learning approaches. In the interest of comparing different types of splitters, it is seen as ideal to test at least one algorithm from each category.

The work of Sjöbergh and Kann (2004) is a good rule-based candidate due to the degree of implementation details available. The article also mentions several different variants that may be compared against each other. Implementing these methods will also provide a way to verify whether the claim that the algorithm is easily applicable to other languages holds. It is decided to use the methods discussed in Sjöbergh and Kann (2004, pp. 3–4) called 'Baseline', 'Part of Speech of Components', 'Character n-grams', and 'Hybrid' as baseline and improvements to conduct Norwegian experiments on, respectively.

Alfonseca et al. (2008) obtain very good results for Norwegian and commend the quality of the corresponding training data, making this work a very attractive machine learning candidate to compare against. However, neither the project's source code nor the training data described could be located, so unfortunately these results cannot be used.

The fact that *SECOS*, see Riedl and Biemann (2016), is available online, along with pretrained models, makes it a very promising candidate to compare against. With no obvious disadvantages, *SECOS* is chosen to serve as a machine learning approach to compare Sjöbergh and Kann's methods against.

All the splitters, except *SECOS*, use the NST (Nordisk Språkteknologi, see Section 4.1.2) Lexical Database as knowledge source for their compound splitting.

5.1.1. Architecture of the Statistical Approach

Architecture of the Baseline Method

Four of Sjöbergh and Kann's methods have been chosen for implementation and are compared. The most basic solution, choosing the suggestion with as few components as possible, serves as baseline for comparison. A logic flow diagram of its implementation can be seen in Figure 5.1 on page 33.

First of all, the corpus has to be loaded. A number of list structures that hold information about what compounds are being treated, their correct splits and what suggested split the method will return also have to be initialised. Next, the method reads a tab separated file containing the compounds and their correct splits. Line by line, the file is read and fills the above lists. For finding the suggestion with the fewest constituents, each compound is recursively parsed; starting at the beginning of the compound, increasingly longer substrings are checked to see whether one is contained in the corpus. If it is, the parsing continues from the next character, and so on, until the end of the string is reached. All substrings of the compound have to be identified as either a lexeme or an epenthesis, or else the suggestion is discarded. This approach contrasts with the arguments of Johannessen (2001) and Faarlund et al. (1997), as discussed in Section 2.4, where a constituent does not have to be lexicalised. Unfortunately, resources indicating root forms would be needed to enable the kind of analysis they suggest. For the purpose of this thesis, no such lexica are available, so the above reasoning is disregarded, and only those compounds whose constituents are found in a dictionary are considered valid.

Upon completion of the parsing, all suggestions are ordered by their number of constituents. If two suggestions have the same lowest number of constituents, the one with the longest final member is chosen as the best split. This suggestion is added to the list of suggested splits before the next line in the input file starts parsing.

Sjöbergh and Kann report that this method yields 90 per cent accuracy on ambiguous Swedish compounds.

Architecture of the POS method

For the second method, the suggestion whose constituents make up the most frequent combination of parts-of-speech (POS) is chosen. These frequencies are collected from the compounds in NST. Data from the same source determines the POS of the constituents of the suggested splits. This implementation is shown in Figure 5.2 on page 34.

Like the baseline method, the POS method starts by loading its corpus, but it also needs to calculate the n-gram distributions found in said corpus. Once this is done, the same list structures that are used by the baseline method are initialised, and the compounds and their correct splits are extracted. Parsing is also done in the same way,

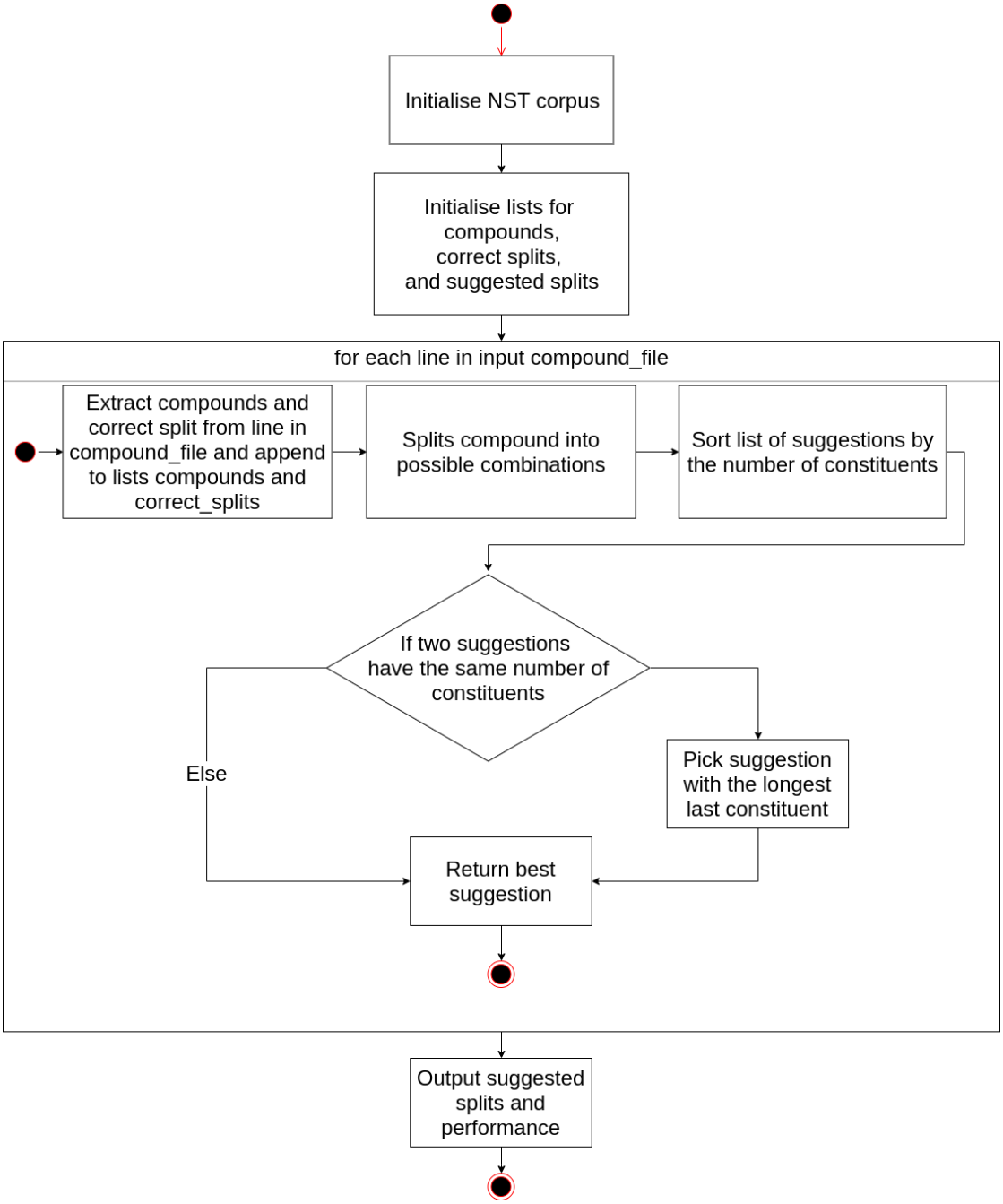


Figure 5.1.: Logic flow diagram of the baseline method.

but once all suggestions are collected, these are looked up in the corpus and stored in a tuple along with their POS. Some constituents may have more than one possible POS. To be able to analyse multiple classes of a constituent, extra entries have to be created

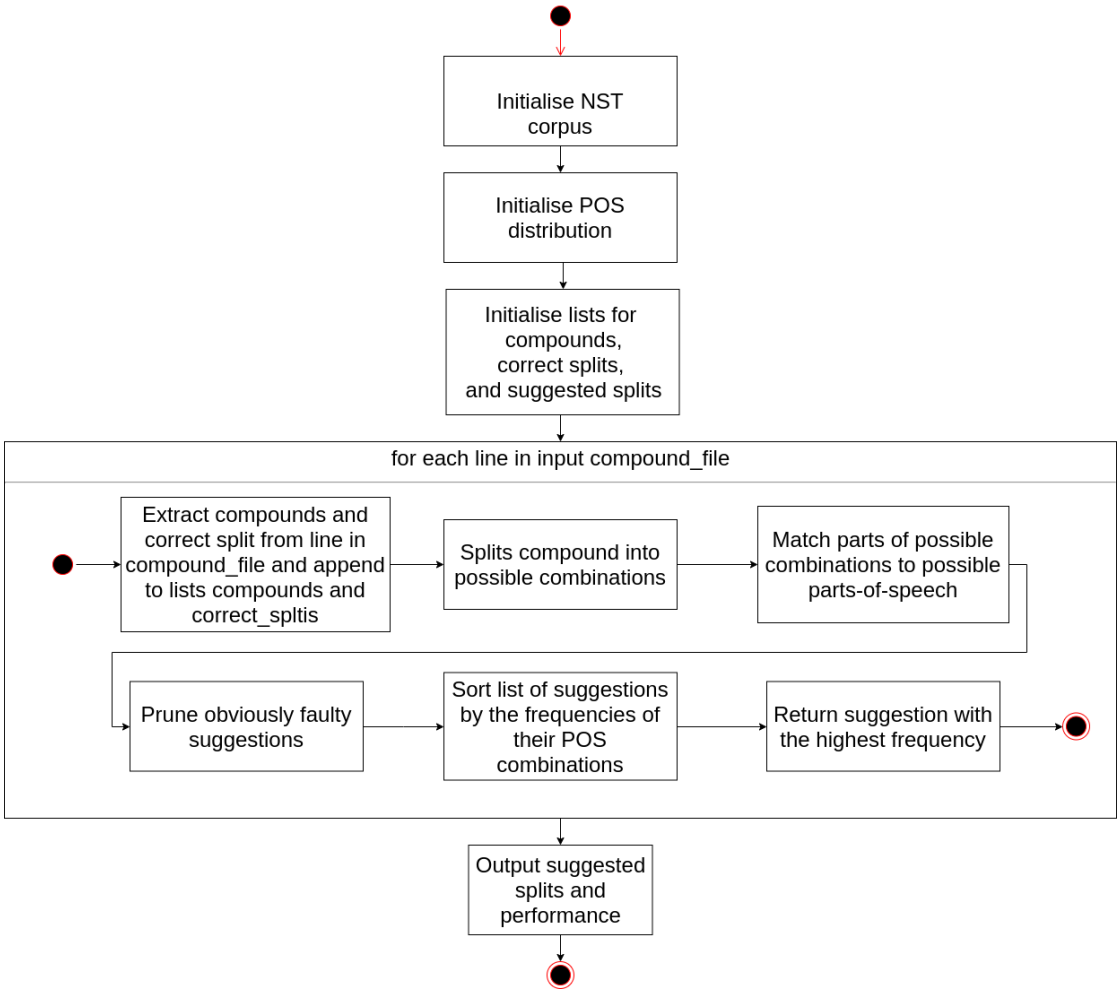


Figure 5.2.: Logic flow diagram of the POS method.

in the list of suggestions if this situation arises. This means that the size of the list can grow very quickly. To avoid having to look up the same part multiple times, memoisation is used: a data structure that keeps track of lexemes that have been looked up earlier is put in place. The POS method only consults the corpus if the constituent and its POS is not found in this structure.

Once all suggestions and their constituents have been matched with a POS tag, this list is pruned for obviously faulty suggestions, e.g., those where two epentheses follow each other or an epenthesis is the first or last of the suggested constituents. This is possible because the POS method is able to distinguish epentheses from other substrings using the POS tag. When the suggestions have been pruned, the remaining splits are sorted against the POS distribution of the corpus. Finally, the suggestion with the highest ranked POS combination is returned.

Sjöbergh and Kann report 91 per cent accuracy on Swedish compounds for this method.

Architecture of the N-Gram Method

According to Sjöbergh and Kann (2004, p. 901), the third method achieves 91 per cent accuracy on Swedish compounds. It uses character n-grams, which is fixed as $n = 4$ in this thesis. This is the parameter Sjöbergh and Kann suggest using.

As can be seen in Figure 5.3, like in the two previous methods, initialising NST is the first step to be taken. Now the corpus is used to find the frequencies of all character n-grams in compound heads and modifiers. The n-grams intentionally leave out the overlap between the parts. Suggested splits are parsed like in the two previous implementations.

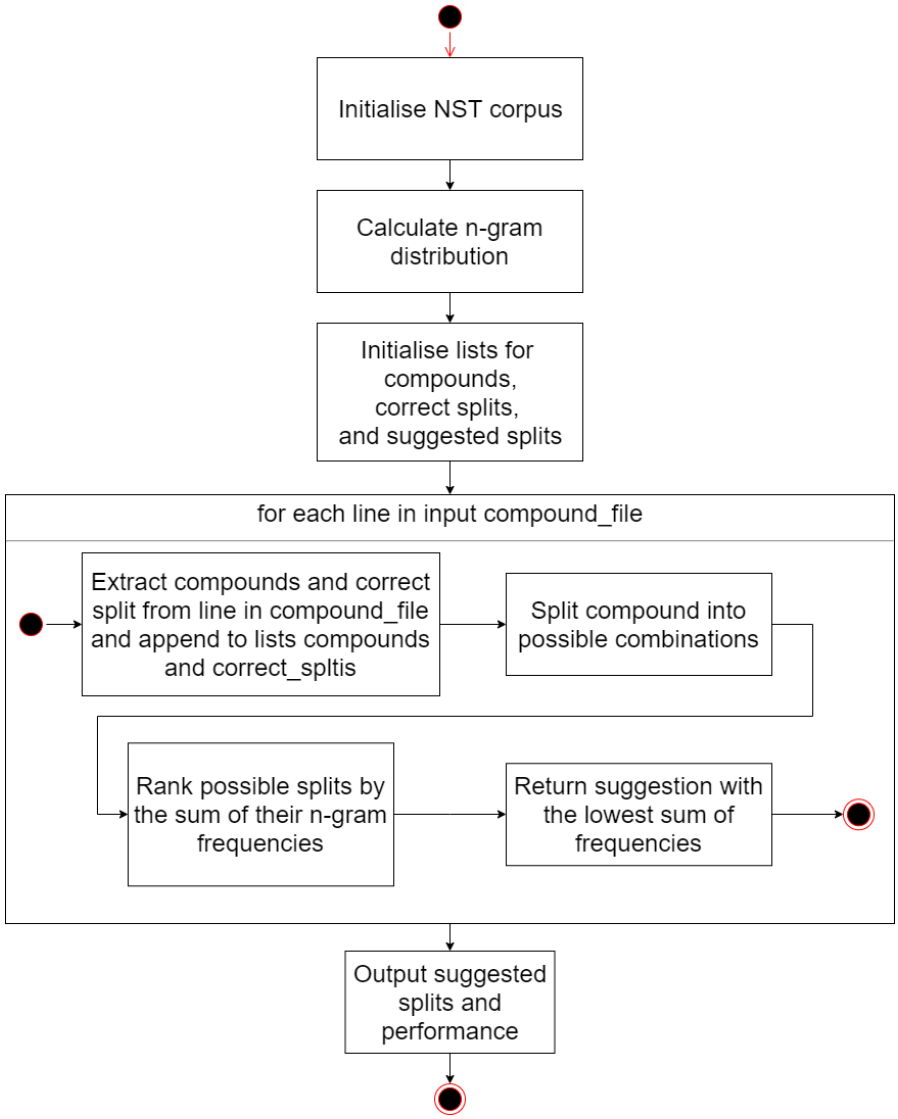


Figure 5.3.: Logic flow diagram of the n-gram method.

Then, the n-grams of all substrings of the suggested constituents are compared against the frequency distribution of the corpus. The n-gram frequencies are added, and the suggestion with the lowest sum is chosen. This suggestion has splits at the positions most unlikely to not contain a split, i.e., the splits are at the most likely positions.

Architecture of the Hybrid Method

Finally, a hybrid method is implemented, combining the n-gram and POS methods. The suggestion from the n-gram method takes priority, but if the POS method has a suggestion with a probability that is more than five times higher than the probability of the 4-gram suggestion, the POS suggestion is used. The logic flow of this method can be seen in Figure 5.4 on page 37. In Sjöbergh and Kann's experiments, this approach gave an accuracy of 94 per cent on Swedish ambiguous compounds.

Sjöbergh and Kann also suggest using some *ad hoc* rules to deal with problems like three consecutive identical consonants being reduced to two and thus being split incorrectly. This rule has not been implemented in this thesis, but to reduce complexity and excessive splitting of short constituents, all methods impose a minimum length of three characters on the constituents. Epentheses are exempt from this rule, of course, since they can have the length of one or two characters.

5.1.2. Architecture of the Machine Learning Approach

Riedl and Biemann's *SECOS* was chosen as a machine learning approach to compare against because it is readily available on GitHub. It also comes with precomputed models for a handful of languages, including Norwegian. As mentioned in Subsection 3.1.8, it is built around the idea that compounds are similar to their constituents. Implementation details can be found in Riedl and Biemann (2016, pp. 618–619).

5.2. Choosing a Compound Splitter

Motivated by the articles studied in Section 3.1, Section 6.1 investigates how the different algorithms fare when applied to Norwegian compounds. The experiments show that neither of the implemented methods is perfect. However, as pointed out in Section 8.1, the intrinsic accuracy of the splitter does not correlate with the translation quality. Other features of the splitter should therefore be prioritised when deciding on which algorithm to use. Since the rest of the work in this thesis relies heavily on POS information, it is decided that the POS variant of Sjöbergh and Kann will be used in the splitting and tagging of compounds. This method reaches the highest average accuracy of the single-pass algorithms; the Hybrid method does reach a slightly higher average, but in return, it takes far longer to run since it effectively runs both the n-gram method and the POS method. This slight gain in accuracy is not considered an important enough trade-off. Despite struggling with the concept of epentheses, the POS method is therefore considered a satisfactory splitter for the task at hand.

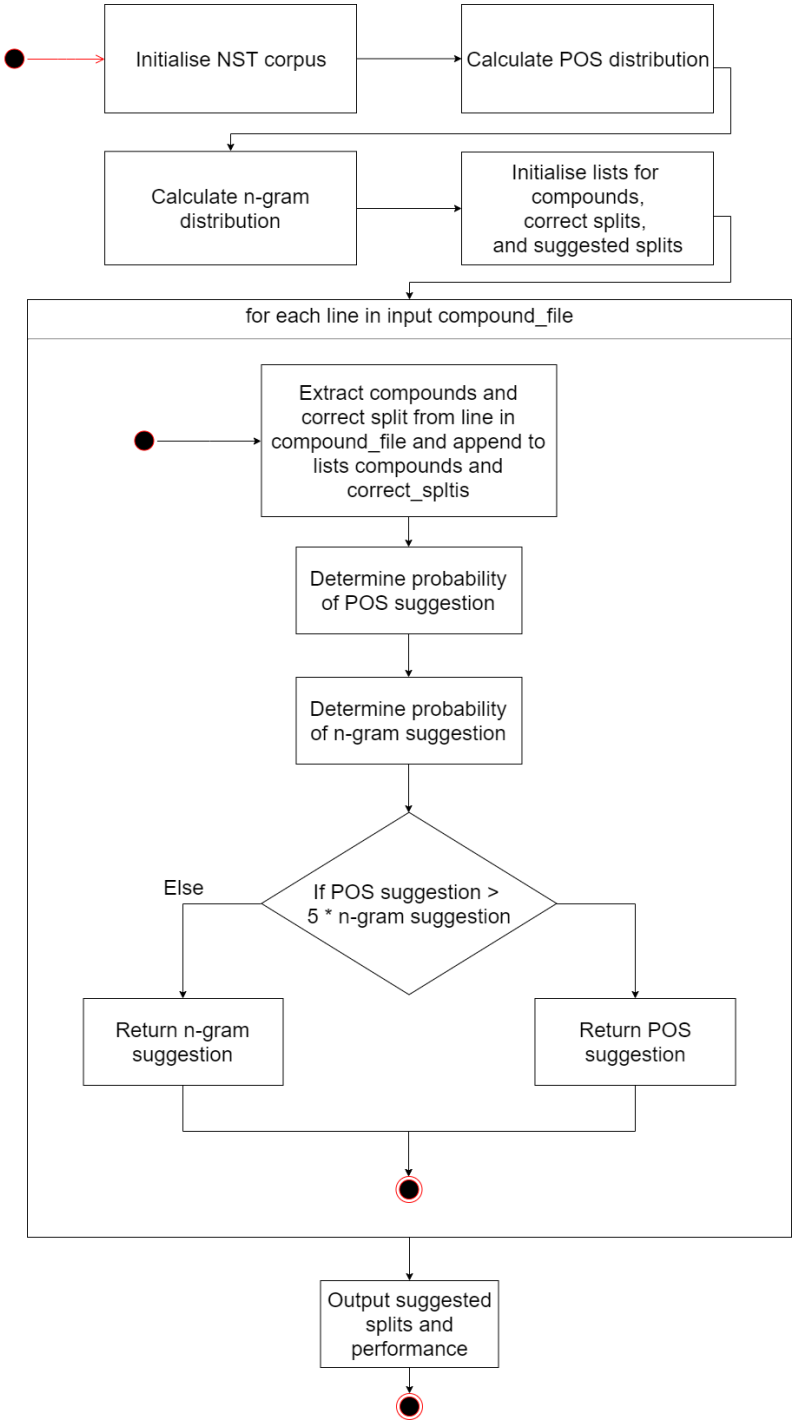


Figure 5.4.: Logic flow diagram of the hybrid method.

5.3. Architecture of the Translation System

The translation process presented in this thesis is based largely on that of Stymne et al. (2013). They adapt the split-merge approach to translate into a compounding language.

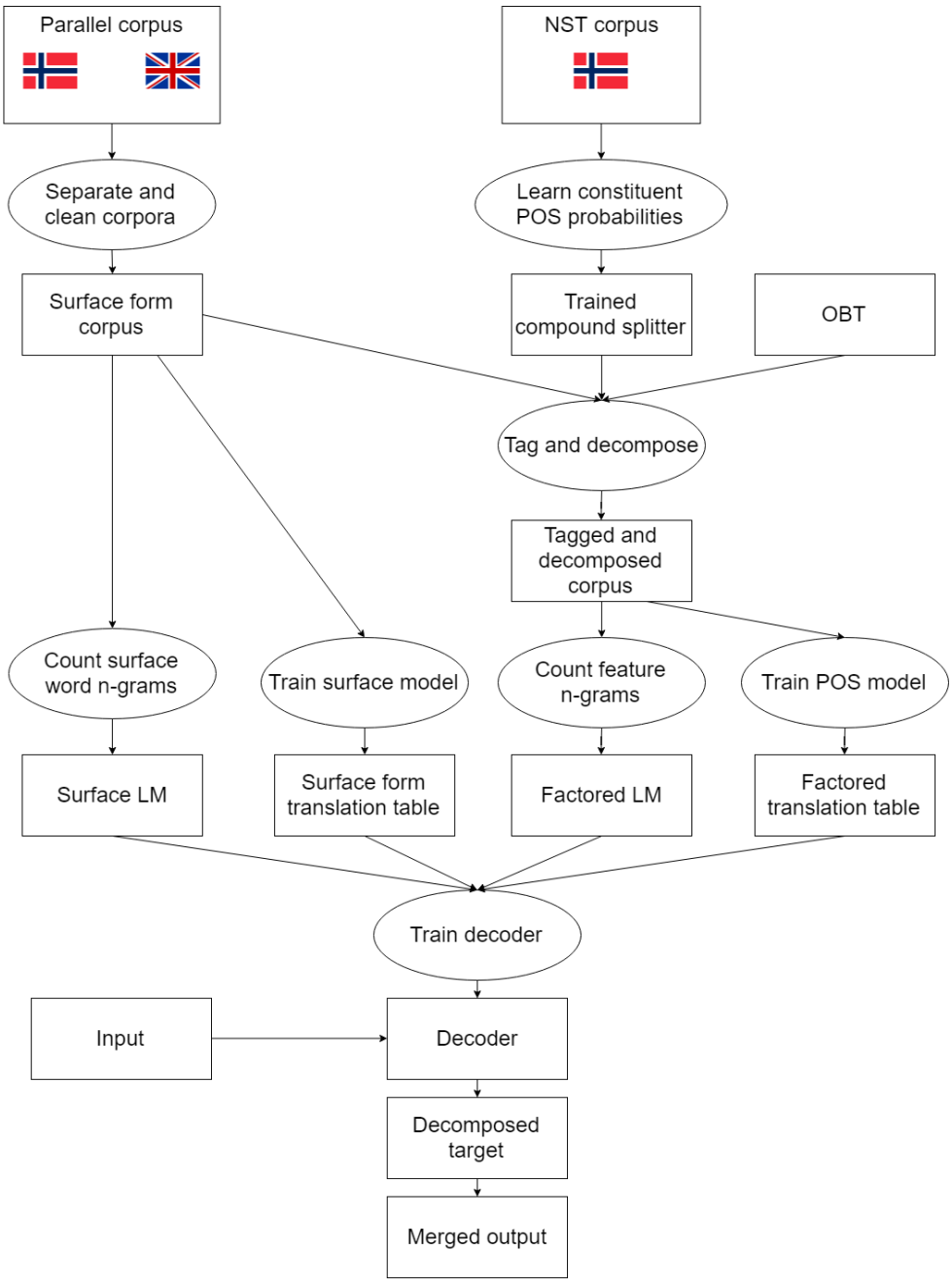


Figure 5.5.: Visual representation of the translation process architecture.

In this strategy, compounds in the target language training corpus (here, the Norwegian side of the corpus) are split and tagged. Then, like described in Section 2.2, a translation model is learnt from the source language (English) into the decomposed target language. After translation (decoding), the merge step reassembles disjoint constituents that make up a compound. Here follows a description of the architecture of the different steps of the translation process as they were implemented in this thesis. Figure 5.5 shows a schematic overview of the training and decoding of the system.

5.3.1. Preparing Training Data

As mentioned in Subsection 4.3.3, the training corpus is on the TMX format, where the two languages are paired in `tu` (Translation Unit) and `tuv` (Translation Unit Variant) tags. Each such pair of translations holds the same semantic content in Norwegian and English. To clean the translation memories, the contents of these tags are extracted using RegEx syntax. The length of the translation units vary: they are usually grouped as complete sentences, but they also appear in the form of single words, especially in the listing of items.

Once the tag contents have been extracted, they are stripped of any eventual XML tags, as well as control characters with index lower than 32 in the ASCII table (i.e., `0x00` through `0x1F`). The translation units are then written to the same line number in two complementing files representing the Norwegian and the English side of the corpus. Listing 5.1 shows an example of the input of the TMX format of one translation unit pair into cleaned text. The opening `tu` tag and the following `prop` tags show some metadata about the document that is not interesting for extracting only the Norwegian and English translations. The relevant content is located in the `seg` (segment) tag in each `tuv`.

```
<tu creationdate="20120626T103426Z" creationid="ALLEGRO"
    changedate="20120626T103426Z" changeid="ALLEGRO"
    lastusagedate="20120626T103426Z">
     <prop type="x-Origin">TM</prop>
     <prop type="x-OriginalFormat">TradosTranslatorsWorkbench</prop>
     <prop type="x-CLX:MultipleString">310D0267</prop>
     <prop type="x-BRK:MultiplePicklist">TELEKOMM</prop>

     <tuv xml:lang="en-GB">
         <seg>Guard band between FDD downlink band edge and FDD uplink
    band edge (duplex gap) (<bpt i="1" type="1" x="1" />2)<ept i="1"
    /></seg>
     </tuv>
     <tuv xml:lang="nb-NO">
         <seg>Beskyttelsesbånd mellom båndgrense for FDD nedforbindelse
    og FDD oppforbindelse (dupleksavstand) (<bpt i="1" type="1" x="1"
    />2)<ept i="1" /></seg>
     </tuv>
</tu>
```

Listing 5.1: Example input of the Translation Memory. This extract shows lines 619-630 in the file `nb_no_telekommunikasjon.tmx`.

Each segment contains the text of the translation unit variant, but also the following elements:

- zero, one or more `bpt` tags;
- the same number of corresponding `ept` elements (`bpt` and `ept` are abbreviations of 'begin paired tag' and 'end paired tag', and demark paired sequences of native code which begin and end in the same `seg` element);
- zero, one or more `it` elements (isolated tag; paired native code that is isolated from its partner, possibly due to segmentation);
- zero, one or more `ph` elements (place holder; marking a stand-alone native code); and
- zero, one or more `ut` elements (unknown tag; demarking a native code that cannot be identified by the TMX processor).

These tags are not part of the actual translations; they provide native codes that mark formatting. They are therefore stripped away using RegEx and only the actual content of the translation variant unit is kept (marked with black font in the `tuv` tag in Listing 5.1). For more details on the TMX format, see the specifications[1].

Listing 5.2 and Listing 5.3 show the equivalent lines of English and Norwegian text that are output from the cleaning script and used in the training corpus. This is done for all seven files listed in Subsection 4.3.3, which are then concatenated into the two training corpus files `concat.en` and `concat.nb`.

```
Guard band between FDD downlink band edge and FDD uplink band edge
    (duplex gap) (2)
```

Listing 5.2: Result of the cleaned English TUV of Listing 5.1.

```
Beskyttelsesbånd mellom båndgrense for FDD nedforbindelse og FDD
    oppforbindelse (dupleksavstand) (2)
```

Listing 5.3: Result of the cleaned Norwegian TUV of Listing 5.1.

5.3.2. Tagging the Training Data

Next, the Norwegian side of the training corpus is tagged with POS (part-of-speech) tags and decomposed accordingly. The training data is tagged using OBT (Oslo-Bergen-Taggeren, see Section 4.1.2). If a word is unknown to OBT, it will be tagged as `samset`, short for Norwegian *sammensetning*, compound. The word will also be tagged with the conventional POS tag (verb, noun, adjective, etc.) that OBT finds it the most likely to belong to. How OBT determines this is beyond the scope of this thesis. The `samset` tag is further extended as either `leks` or `analyse`. The first option, presumably an abbreviation of *leksikalisert* (lexicalised), includes examples like *uvanlig* (unusual, extraordinary), *mørkegrønn* (dark green), and *overleve* (survive), indicating that OBT recognises this word as a compound, but that it is so common that it is found in the

[1] `http://xml.coverpages.org/TMX-SpecV13.html`

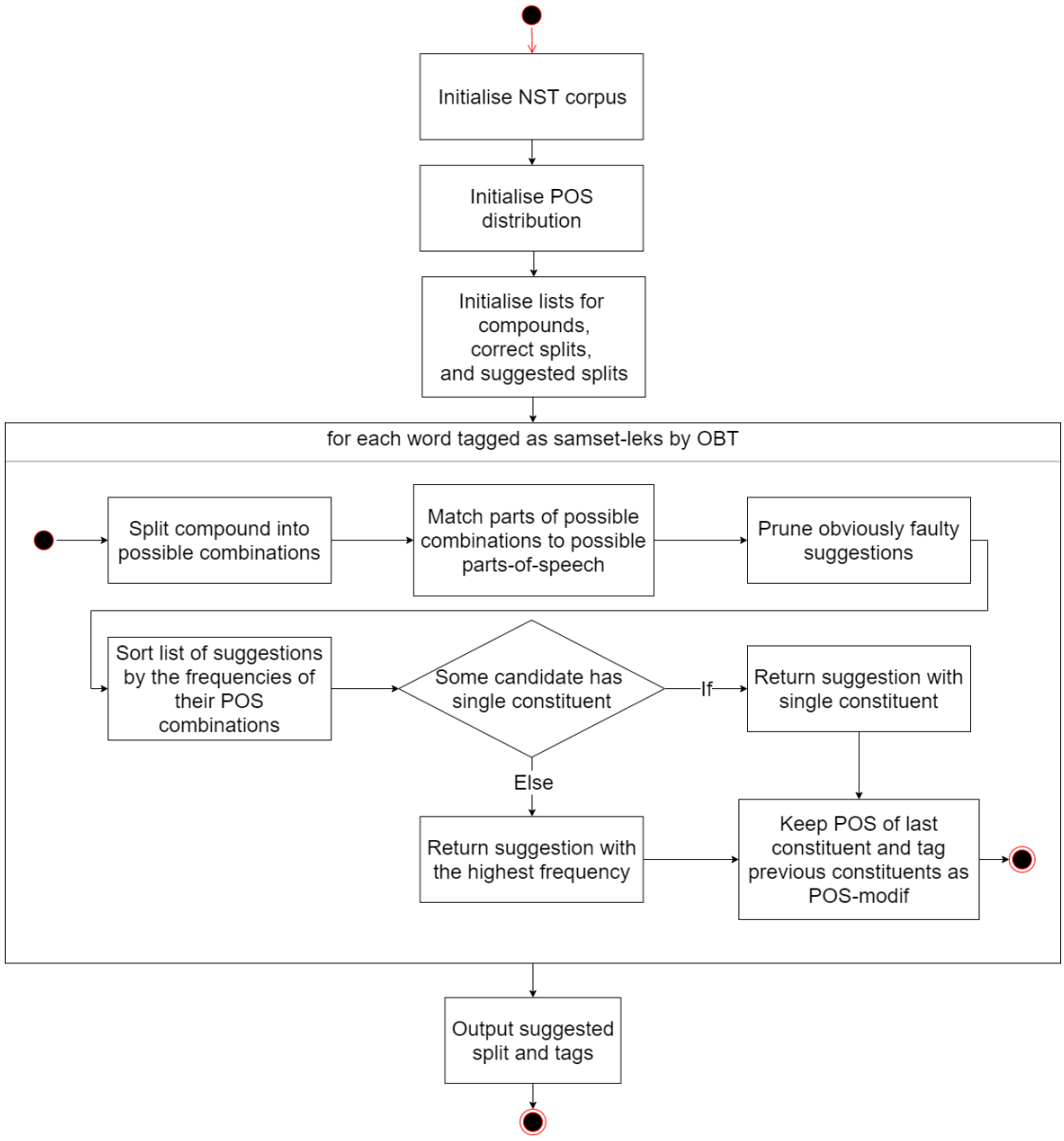


Figure 5.6.: Logic flow diagram of the modified compound splitter for POS tagging.

Norwegian dictionary OBT uses. Compounds labelled with this tag are therefore not analysed by the custom splitter. On the other hand, the `samset-analyse` tag shows that OBT assumes that this word is a compound, but that it is not contained in the tagger's lexicon. The unknown compound is followed by an additional `<+>` tag which indicates the purported POS of the compound head. However, this suggestion may be incorrect, so it is not used by the custom splitter.

Furthermore, OBT's tagging uses *Norsk Ordbank* (Norwegian Word Bank) for multi-

tagging, whereas the custom splitter's word list is based on the *Nordisk språkteknologi* (Nordic Language Technology) database. The contents of these two resources differ somewhat and it is thus possible that NST contains some words that *Ordbank* does not. This means that the word bank may mark a word as a compound when it is in fact lexicalised. The custom splitter is therefore modified to return a word as a single lexeme (no splits) if such a candidate is found, even if there are other candidates whose constituents' POS have a higher probability. A more detailed justification for this is discussed in Subsection 7.1.1. The updated logic flow of the POS compound splitter can be seen in Figure 5.6.

Compounds that are split are tagged in accordance with Stymne et al. (2013, p. 1076): the constituents are marked with a special `X-modif` tag, indicating that they are part of a compound and modify the compound head. Stymne et al. call this the EPOS (extended part-of-speech) tagset. The `X` in the `X-modif` tag represents the POS of the head, as well as the POS of the compound as a whole. This disregards the observations made by Johannessen (2001), where in certain cases the head does not determine the compound's POS (see Section 2.4). These cases were deemed to be so rare that the practicality of assuming the same POS outweighed the strict correctness. Similarly, if the compound consists of more than two parts, the head, i.e., the last constituent, will be marked as `X`, and all preceding parts are marked as `X-modif`, where `X` represents the POS of the head as suggested by the custom splitter. If the custom splitter is unable to find any possible splits, the suggestion from OBT is used.

Tagging the concatenated training corpus leaves 961,116 decomposed Norwegian tokens, 827,682 Norwegian tokens (compounds not split) and 938,666 English tokens.

5.3.3. Training Phase

Koehn and Hoang (2007) describe how, in factored decoding, each word is represented by a vector of factors, e.g., stem forms of the word or POS tags. This enables extra linguistic information to be added to a phrase-based system. When training a model without POS tags, each word is only represented by its surface form (as it appears in raw text). In this thesis, Moses, Koehn et al. (2007), which is a free SMT (statistical machine translation) system that supports factored translation, is used. In particular, Moses is trained with the parameter `--translation-factors 0-0,1`. This indicates the input and output factors for the translation table. For the input, the surface form is the only factor (index 0 of the factors in the table), and the output factors are the surface form (index 0) and POS (index 1). This will cause the phrase table to contain POS tags only on the target side. A language model trained on POS tags will prefer certain sentence structures based on the order of the POS of the words.

Training Moses on both surface forms and POS tags requires two language models (LMs): a POS-sequence model in addition to the standard surface form model. For the purpose of the experiments presented in Section 6.2, 3-gram LMs are built with SRILM, of Stolcke (2002).

Next, the training corpus is sentence aligned to identify the relationships between words and phrases in the two languages. As mentioned in Subsection 5.3.2, compound

modifiers are marked with the custom `X-modif` POS tag. Training a factored model on this special tag improves the coalescence of compound parts; i.e., it encourages the system to output such tokens close to other compound tokens because this is the only way they occur in the training data. Moses uses the Giza++ toolkit, by Och and Ney (2003), for word alignment.

5.3.4. Decoding Phase

Moses is trained with the following parameters supplied to the `train-model.perl` script:

```
~/mosesdecoder/scripts/training/train-model.perl
    --root-dir root_dir \
    --external-bin-dir ext_bin_dir \
    --corpus clean_corpus \
    -f en -e nb \
    --lm 0:3:surface.lm \
    --lm 1:3:pos.lm \
    --translation-factors 0-0,1
```

The option `--root-dir` indicates where the output files will be stored once the decoder is trained; similarly `--external-bin-dir` specifies where to find the binaries for the external tools Moses uses. The cleaned corpus file (excluding the file extension) is chosen with `--corpus`. The `--e` and `--f` switches indicate the extensions of the target and foreign clean corpus files, respectively. The parameters in `--lm <factor>:<order>:<filename>` specify how many factors there are in the language model. For `surface.lm`, this number is zero, because it contains only the raw text. For `pos.lm` it is one, because this model contains all words as well as a POS tag. The second parameter indicates that both language models are learnt using 3-grams of words from the training corpus. Finally, `--translation-factors 0-0,1` instructs Moses that the input contains no factors, only the surface form, and that it should output the surface form and its first factor on the target side.

Once the training is complete, these parameters will cause Moses to output decomposed Norwegian with POS tags. This is done by supplying the desired input to the Moses configuration file, `moses.ini`:

```
echo "input text" | ~/mosesdecoder/bin/moses
    -f ~/corpus/model/moses.ini
```

Before the input is passed to the decoder, a preprocessing step is performed. This usually involves tokenising the input, but the possibility of splitting the input on hyphens was also explored. Details on the effects of this preprocessing step are closely examined in Subsection 8.2.4.

5.3.5. Postprocessing

As the trained model only translates into decomposed Norwegian, an additional postprocessing step is necessary to construct correctly formed compounds. In Stymne et al. (2013), the CRF++ toolkit[2] and the Matrax decoder of Simard et al. (2005) are used for merging compounds with sequence labelling. Due to time constraints, the postprocessing was cut short in this thesis: since the parameter `--translation-factors 0-0,1` indicates to Moses that it is to produce the Norwegian surface form and POS tags in the translation output, it is possible to exploit the output `X-modif` tag, and simply concatenate with the following term (if any).

In addition, any spaces between hyphens and letter sequences are stripped. Both these steps are automated in a simple script and the results are collected for evaluation.

[2] `https://taku910.github.io/crfpp/`

6. Experiments

This chapters present the experiments performed on the compound splitters discussed in Section 5.1 and the translation system of Section 5.3. The choice of which splitter to use in Section 5.2 is based on the results from this chapter.

Section 6.1 presents the experiments conducted on the splitters, whereas Section 6.2 tests the translation system as a whole.

6.1. Experiments and Results of the Splitters

The translation system translating from English to decomposed Norwegian relies on its underlying compound splitter to learn constituent parts. It is therefore important to examine the different splitters' capability of correctly splitting compounds, and identifying their strengths and weaknesses.

This section addresses these concerns. The evaluations are carried out on the implementations described in Section 5.1, and measures their performance on Norwegian compounds.

6.1.1. Experimental Plan

As mentioned in Section 5.1, three independent methods and one hybrid method based on the work of Sjöbergh and Kann (2004) have been implemented. These methods will be compared against the *SECOS* system, as described by Riedl and Biemann (2016). Preliminary testing included the two test sets of Johannessen and Hauglin, but it was quickly established that this provided little insight into what kinds of mistakes the different splitters were making. The test sets were therefore expanded to encompass the typical errors described in Subsection 4.4.1, and the splitters are tested on these nine sets. The results reported are calculated in accordance with the metrics from the same section. The compounds and the correct splits of the different sets are listed in Appendix A.

6.1.2. Experimental Setup

After having implemented the four methods from Subsection 5.1.1, a script automating the process of extracting the performance measurements was put in place. Detailed instructions on how to use it and reproduce the results reported in the following section are located in Appendix D.1.

SECOS has its own tool set for use in experiments. Instructions on how to use these are described in the same appendix.

6.1.3. Experimental Results

This section presents the performance attained by the different splitters. Table 6.1a shows the results of the baseline and the POS method. Table 6.1b shows those of the n-gram method with $n = 4$ and the hybrid method. The best accuracies are highlighted in bold font. In Table 6.1c, the results reported by *SECOS* are listed. The definitions of accuracy, precision, recall and F_1 are given in Subsection 4.4.2.

For the methods used, Sjöbergh and Kann (2004) report the following accuracies: number of components (90 per cent), POS (parts-of-speech) (91 per cent), n-grams (91 per cent), and hybrid (94 per cent).

Except for in the test set *No split*, the baseline method only reaches an accuracy that is at least as high as Sjöbergh and Kann in the *Lexicalised s* test set, which is the set it performs the best on. The POS method also only reaches comparable results on this test set. On the other hand, these accuracies are a little higher than what Sjöbergh and Kann report.

The hybrid method also performs the best on this set, but here the results are a little lower than in the original paper. For *SECOS*, with the exception of *No split*, the results obtained here are not comparable to the original results.

The most striking feature of the results is that, contrary to Sjöbergh and Kann's results, the n-gram method performs the worst. It only comes out the best in the *Epenthetic e* test set, but the scores here are extremely low for all methods. It is nowhere near attaining the performance reported for Swedish compounds.

Upon inspecting the potential splits that are found by the method and ranking them by their corresponding probabilities[1], it is revealed that the correct split is often the one with the highest probability. However, Sjöbergh and Kann (2004, p. 4), specifically state that "[f]or each suggestion from the compound splitter all frequencies of the character 4-grams spanning the suggested splits were added. The suggestion with the **lowest**[2] sum was then selected."

The n-gram method is modified to return the item with the highest probability. A single instance of this modified version is then applied to *No epenthesis*, which increases the accuracy from 0.0536 to 0.8036. Precision is raised by the same amount and it obtains perfect recall. F_1 is increased from 0.1017 to 0.8911. This motivates the additional experiment detailed in Table 6.2. The modification substantially raises the performance levels for most methods. Those results that are improved are marked in boldface.

With the exception of *No split*, the performance of the n-gram method is still not as high as reported by Sjöbergh and Kann, but the scores are substantially improved compared to using the suggestion with the lowest sum.

Seeing as these experiments evaluate Norwegian compounds, the results are compared against the articles that do the same. This is especially relevant for Thorsen Ranang (2010) and Lindbråten (2015) who test on the same test set, *Johannessen*. However, nor

[1] To see these scores, run the n-gram method with the debug flag enabled:
`python3 ngrams_sjobergh.py -debug -f test_set`.
The splits and their probabilities are listed under 'Splits and scores'.

[2] Emphasis added for the purpose of this discussion

	Baseline				POS			
	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
Johannessen	**0.2162**	0.3636	0.3478	0.3556	**0.2162**	0.3636	0.3478	0.3556
Ambig.	0.4407	0.4483	0.9630	0.6118	**0.4492**	0.4609	0.9464	0.6199
Epenth. e	0.1250	0.1250	1.0000	0.2222	**0.5000**	0.5000	1.0000	0.6667
Epenth. s	0.0000	0.0000	0.0000	0.0000	0.0200	0.0200	1.0000	0.0392
Lexic. e	**0.8824**	0.9375	0.9375	0.9375	0.7059	0.7500	0.9231	0.8276
Lexic. s	**0.9375**	0.9375	1.0000	0.9677	**0.9375**	0.9375	1.0000	0.9677
No epenth.	**0.8571**	0.8571	1.0000	0.9231	0.8241	0.8364	0.9787	0.9020
No split	**1.0000**	1.0000	1.0000	1.0000	0.8750	1.0000	0.8750	0.9333
Repr.	0.7291	0.6447	0.9899	0.7809	**0.7389**	0.6623	0.9804	0.7905

(a) Performance of the baseline and POS method of Sjöbergh and Kann.

	4-grams				Hybrid			
	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
Johannessen	0.0541	0.0556	0.6667	0.1026	**0.2162**	0.3478	0.3636	0.3556
Ambig.	0.1102	0.1111	0.9286	0.1985	**0.4492**	0.4569	0.9636	0.6199
Epenth. e	0.0000	0.0000	0.0000	0.0000	**0.5000**	0.5000	1.0000	0.6667
Epenth. s	**0.1600**	0.1600	1.0000	0.2759	0.0200	0.0200	1.0000	0.0392
Lexic. e	0.0000	0.0000	0.0000	0.0000	0.7059	0.7500	0.9231	0.8276
Lexic. s	0.0000	0.0000	0.0000	0.0000	**0.9375**	0.9375	1.0000	0.9677
No epenth.	0.0536	0.0536	1.0000	0.1017	0.8214	0.8214	1.0000	0.9020
No split	0.0000	0.0000	0.0000	0.0000	0.8750	1.0000	0.8750	0.9333
Repr.	0.1281	0.1244	0.9615	0.2203	0.7291	0.6447	0.9899	0.7809

(b) Performance of the 4-gram and hybrid method of Sjöbergh and Kann.

	SECOS			
	Acc.	Prec.	Rec.	F1
Johannessen	0.189189	0.942308	0.521277	0.671233
Ambig.	0.161017	0.924242	0.570093	0.705202
Epenth. *e*	0.000000	1.000000	0.464286	0.634146
Epenth. *s*	0.000000	0.892473	0.494048	0.636015
Lexic. *e*	0.470588	0.931034	0.750000	0.830769
Lexic. *s*	0.312500	1.000000	0.656250	0.792453
No epenth.	0.321429	0.952381	0.666667	0.784314
No split	**1.00000**	1.00000	1.00000	1.00000
Repr.	0.290640	0.800570	0.587866	0.677925

(c) Performance as reported by *SECOS*' built-in evaluation tools.

Table 6.1.: The performances of the different implemented splitters.

	4-grams				Hybrid			
	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
Johan.	**0.2432**	**0.4091**	0.3750	**0.3913**	**0.2434**	**0.3913**	**0.3913**	**0.3913**
Ambig.	**0.4153**	**0.4224**	**0.9608**	**0.5868**	0.4492	0.4569	0.9636	0.6199
Ep. e	**0.1250**	**0.1250**	**1.0000**	**0.2222**	0.50000	0.5000	1.0000	0.6667
Ep. s	0.0000	0.0000	0.0000	0.0000	0.0200	0.0200	1.0000	0.0392
Lex. e	**0.8821**	**0.9375**	**0.9375**	**0.9375**	0.7059	0.7500	0.9231	0.8276
Lex. s	**0.6875**	**0.6875**	**1.0000**	**0.8148**	0.9375	0.9375	1.0000	0.9677
No ep.	**0.8036**	**0.8036**	1.0000	**0.8911**	0.8214	0.8214	1.0000	0.9020
No spl.	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.8750	1.0000	0.8750	0.9333
Repr.	**0.7438**	**0.6623**	**0.9901**	**0.7937**	**0.7488**	**0.6689**	**0.9902**	**0.7984**

Table 6.2.: Performance of the 4-gram and hybrid method of Sjöbergh and Kann, using the 4-gram suggestion with the highest associated probability. Those results that were improved are marked in boldface.

here do any of the implemented methods attain the same performance, viz., 90, 91, 91, and 94 per cent, respectively.

6.2. Experiments and Results of the Translation System

The constructed translation system attempts to construct unseen compounds in the technical domain of computer science. The experiments on the translation system are designed to reveal whether these translations are deemed adequate by native speakers. This section describes how these experiments were conducted by the use of surveys, and presents a subset of the results. The full description of the results can be found in Appendix B.

6.2.1. Experimental Plan

To conduct the experiments, the translation system has to be trained, test data must be translated, and translations need to be distributed to native speakers.

The Moses decoder, Koehn et al. (2007), is first trained on a high-quality decomposed parallel corpus as described in Section 5.3. English technical terms without a known Norwegian translation are then collected from Wikipedia and supplied to the decoder. The translation system then outputs a presumably likely translation along with POS tags. The decomposed translation goes through a postprocessing step, and words are concatenated or not based on their output features.

Subsequently, the output translation is assessed by native speakers of Norwegian. Evaluating the output with an automatic framework like BLEU, see Papineni et al. (2002), is impossible, because these methods are based on comparisons against reference translations. The very essence of creating unseen compounds is that there are no previous

translations which can be used for comparison. The native speakers decide whether they find the translation to be understandable, of native quality, and grammatically correct.

6.2.2. Experimental Setup

The test set is composed of 99 machine translations, from which surveys containing ten translations each are constructed. Out of these ten translations, one is a human-translated term from the Wikipedia page titles that do have a Norwegian equivalent (see Section 4.4), representing control translations. This gives a total of 11 surveys. See Table B.1 and Table B.2 in Appendix B for the contents of the different surveys.

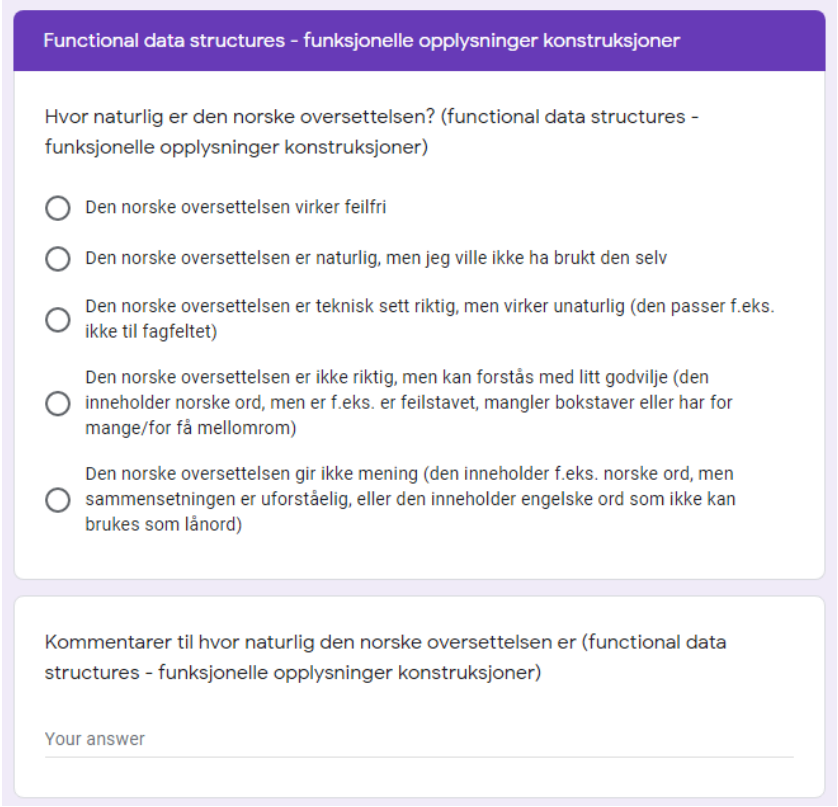
Seeing as the control terms are open for vetting, they are presumed to be of acceptable translation quality. The control translations are used to determine what level of adequacy a translation has to reach to be on par with a satisfactory human translation. The surveys are constructed and answers are managed using Google Forms[3]. The surveys are given in Norwegian, since they ask native speakers to assess Norwegian translations. Each participant, henceforth referred to as an 'evaluator', is assigned to a randomly chosen survey. To avoid confusion with the evaluations made in Chapter 7, the term 'rating' will be used to refer to an evaluator's assessment of a translation.

The ratings are conducted blindly: the evaluators do not know that the translations have been produced by a machine translation system. First of all, the evaluators are introduced to the purpose of the survey and instructions on how to answer. The instructions explicitly state that the survey does not seek better translations, but simply to rate the ones presented. It also indicates that the intended participants are native Norwegian speakers with knowledge of the computer science domain, but no countermeasures are made to prevent participants not fulfilling these criteria from submitting their answers. The quality of the answers is therefore largely based on the assumption that the participants respond honestly. The survey intentionally does not gather any personal information like age, gender or level of education. This is because the survey only seeks to gather ratings of the translations, not to determine why the evaluators make the assessments they do. The survey's translations are then presented, one by one.

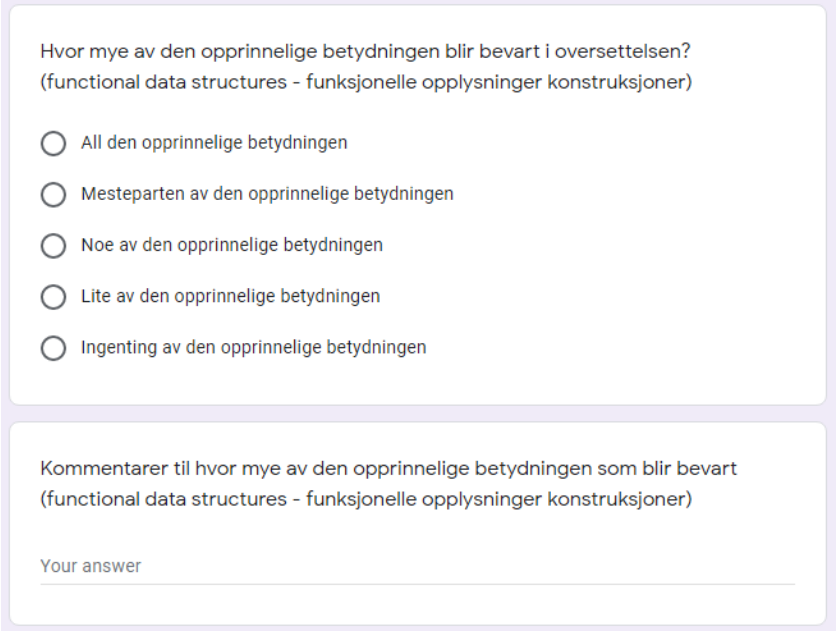
For each translation, an initial question determining whether the participant is qualified to rate the Norwegian translation is asked: are they familiar with the English term? No Norwegian translation is presented at this point. Choosing between three radio buttons, the participant answers either (1) yes, (2) no, but I understand what it means, or (3) no, and I do not understand what it means. If the term is not understood, the question is skipped, and they are asked the same question about the next English term.

If the term is understood, the participant is asked to assess the suggested Norwegian translation for this English term. Here, they are asked to rate how natural the Norwegian translation is, choosing one of five alternatives:

[3] `https://www.google.com/forms/about/`



(a) Question about naturalness of the Norwegian translation.



(b) Question about meaning preserved in the Norwegian translation.

Figure 6.1.: An example survey question of the translation *funksjonelle opplysninger konstruksjoner* (functional data structures).

(1) the translation is perfect;
(2) the translation is natural, but I would not use it myself;
(3) the translation is fluent, but it seems unnatural; e.g., it is not suitable for the domain;
(4) the translation is incorrect, but is possible to understand; e.g., it contains Norwegian words, but has orthographic faults or has too many/too few spaces; or
(5) the translation is incomprehensible; e.g., it contains Norwegian words but the translation as a whole is nonsensical, or it contains English words that cannot be used as loanwords.

The second question posed is how much of the original meaning that is preserved in the translation. Again, five alternatives are presented:

(1) all of the original meaning;
(2) most of the original meaning;
(3) some of the original meaning;
(4) little of the original meaning; or
(5) no original meaning.

Both of these questions provide a comment section in which the evaluator may supply additional information as to why they make the ratings they do. The above process is repeated for all ten translations before the answers are aggregated and submitted. Figure 6.1 shows an example of the layout of a survey question.

The surveys are distributed to persons in the author's academic and professional network who are deemed suitable evaluators. A request to contact all students currently studying at the Department of Computer Science at NTNU (the Norwegian University of Science and Technology) was made to the study administration, but the request was not answered in time to make use of this commodity. It is therefore challenging to gather enough survey participants for the results to be statistically significant; 50 native speakers participated, giving each of the 110 translations an average of 4 ratings (rounded down).

6.2.3. Experimental Results

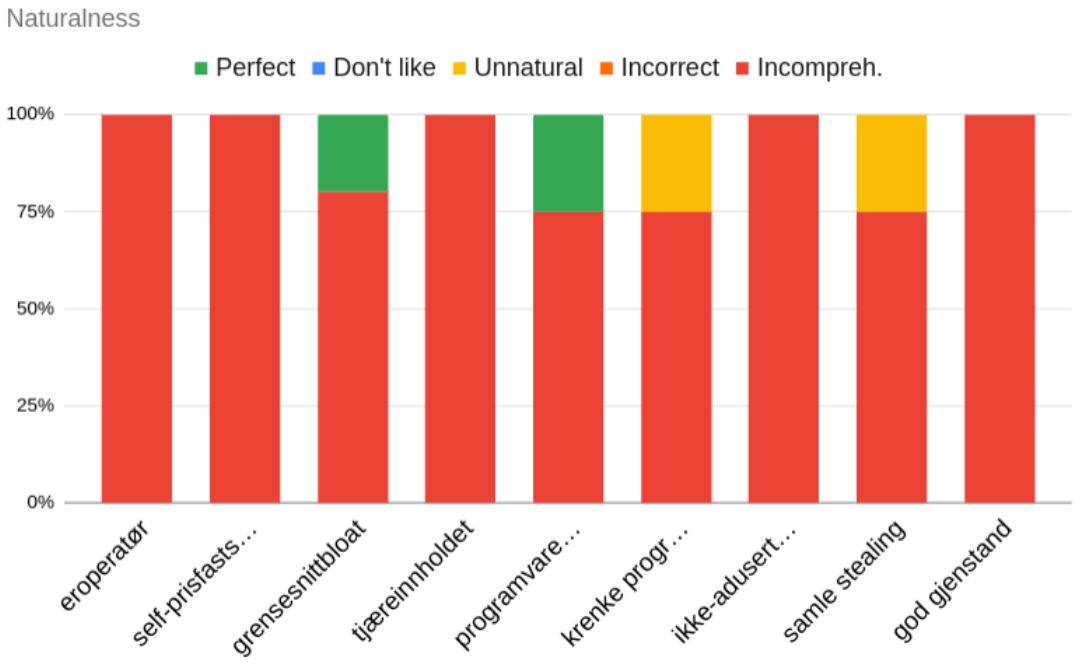
Table 6.3 shows the translations that were the best received by the native speakers; in 100 per cent of the cases they were rated as natural translations and preserving all of the original meaning. The table shows the Norwegian translations, the ID of the translation (which is used to refer to the translations in Table B.1 and Table B.3 in Appendix B), and the distribution of the ratings of said translation. Out of these, *søke-algoritmer*, *hardkoding*, *matematikk-biblioteker*, *rolle-orientert programmering* and *vitenskapelige programspråk* form novel compounds from decomposed translations. The full test set showing the English terms, the Norwegian translations, the tagged Norwegian translations and their ratings are found in the tables in Appendix B.

The translations that received the worst ratings (assessed as incomprehensible by 75 per cent or more of the evaluators) are shown in Figure 6.2a. The ratings of the same

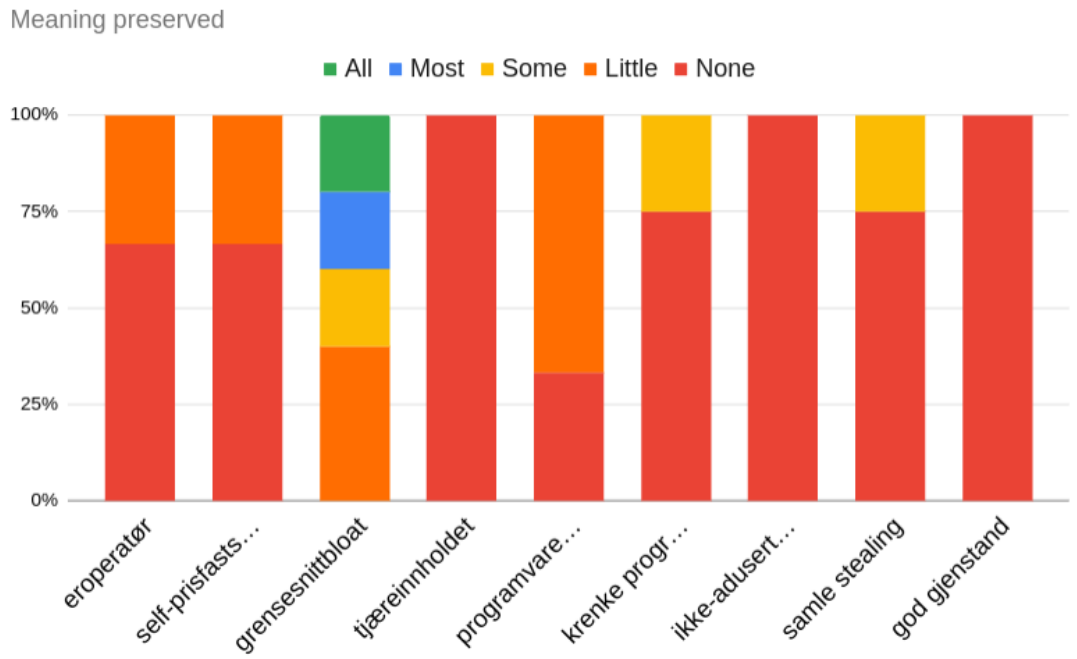
translations' meaning are shown in Figure 6.2b. These and other translations exhibiting interesting traits are further elaborated on in Chapter 7 and Chapter 8.

Table 6.3.: Best rated machine translations.

ID	Norwegian Translation	Norwegian tagged output
15	reduksjon	reduksjon|subst
16	søke-algoritmer	søke-|subst algoritmer|subst
17	hardkoding	hard|subst-modif koding|subst
27	konstant	konstant|adj
44	matematikk-biblioteker	matematikk-|subst biblioteker|subst
60	programmeringsspråk laget av kvinner	programmeringsspråk|subst laget|subst av|prep kvinner|subst
61	akademiske programmeringsspråk	akademiske|adj programmeringsspråk|subst
66	funksjonelle språk	funksjonelle|adj språk|subst
69	sikre programmeringsspråk	sikre|adj programmeringsspråk|subst
76	rolle-orientert programmering	rolle|subst-|strek orientert|adj programmering|subst
77	vitenskapelige programspråk	vitenskapelige|adj program|subst-modif språk|subst



(a) The distribution of the assessed naturalness of the worst translations.



(b) The distribution of the assessed meaning of the worst translations.

Figure 6.2.: The distribution of the assessed naturalness and meaning preserved of the worst machine translations; i.e., *eroperatør* (ID 1), *self-prisfastsettelse* (ID 6), *grensesnittbloat* (ID 18), *tjæreinnholdet* (ID 22), *programvare relieffkonstruksjon* (ID 57), *krenke programmering* (ID 80), *ikke-adusert kode* (ID 89), *samle stealing* (ID 90), *god gjenstand* (ID 94).

7. Evaluation

This chapter evaluates the results obtained in Chapter 6. In Section 7.1, it is first discussed how 'correct' is interpreted in terms of splitting. Then, the results are analysed in detail, and discrepancies between the results of the different methods are discussed. An error analysis also follows.

Section 7.2 describes some of the most interesting results gathered from the surveys. Where appropriate, it is discussed why unsatisfactory translations are output. Since the translations are assessed by human subjects, it is especially important to perform an error analysis; this is done in Subsection 7.2.3.

The discussion in Chapter 8 reviews what insights the results in these sections provide, and in what way they are limited.

7.1. Evaluation of the Compound Splitters

7.1.1. Correctness of Splitting Compounds Found in the Corpus

This thesis largely focuses on productive novel compounds; i.e., those that do not appear in any corpus. However, a complete splitter should also be able to handle lexemes that are contained in a dictionary. Thus, the following question arises: "should a compound that is found in a dictionary be treated as a single lexeme; i.e., not be split, or should it in fact be split into its constituents?". Seeing as the splitter is used in a translation system, it is decided that this will lay the basis for the decision: if a compound is contained in a dictionary, it is possible that a (better) translation for this compound that is not equal to the translation of its constituents exists. For instance, *pattedyr* can be split into *patte-dyr*, but the literal translation of its constituents, 'mammary gland animal', is not the same as that of the compound as a whole, mammal. Sometimes, the literal translation of the constituents does coincide with that of the whole compound, e.g., for *blåhval* and blue whale. This type of compounds whose meaning equals that of the sum of its parts is called endocentric compound. However, it is impossible to tell whether the compound is an endocentric one simply from looking at it, so it is not desirable to try and guess whether it is endocentric or not.

The same argument can be applied to exocentric compounds; i.e., compounds whose meaning in the source language is not equal to the sum of their constituents. One such example is *løvetann* (dandelion[1]), which literally translates to 'lion tooth'. Even

[1]Dandelion is indeed etymologically related to the Old French name of the plant, *dent de lion*, which has the same meaning as the Norwegian translation, but this does not affect the subsequent discussion, so the fact is disregard for the purpose of this thesis.

if the splitter were to make the correct decision; i.e., *løve-tann*, the translation of these constituents would be meaningless, unless one were indeed discussing the literal teeth of a lion.

Since keeping the compound as a single word if it is contained in the training corpus gives the correct translation in both of the above cases, not splitting compounds already found in the training corpus is justified. On the other hand, such compounds can of course still be used for gathering n-grams, investigating epenthetic patterns or other features of compounds.

7.1.2. Analysis of the Baseline Performance

The baseline results are shown in Table 6.1a. The baseline method chooses the suggestion that contains the fewest available splits, or, if two suggestions have the same number of constituents, the one with the longest last constituent.

With this in mind, it is obvious that this method will not split any compound that is contained in its corpus. This is precisely what happens in the *No split* test set, where it obtains perfect scores.

For epentheses, the baseline method prefers to adjoin it to the constituent before or after if possible, because this leaves fewer constituents. This causes the poor results of *Epenthetic e* and *Epenthetic s*, and the good results of the lexicalised sets. *No epenthesis* fares quite well, but fails when the last constituent is not the longest, e.g., in *plante-stasjon* (plant station, ID 18 in Table A.1 in Appendix A) which the method splits into **plan-testasjon* (plan testation).

In the *Ambiguous* test set, the baseline struggles with false positives, particularly splitting in the wrong position. This is largely due to the preference of lexicalising *s* and *e*. As for the more general test sets, i.e., *Johannessen* and *Representative*, the results still do not come close to those reported by Sjöbergh and Kann. In the *Representative* test set, the splitter still makes quite a few splits in the wrong location. Unsurprisingly, these are largely due to mistaking epentheses for lexicalised *e*'s and *s*'s.

7.1.3. Analysis of the POS Method Performance

The POS method's performance is shown in Table 6.1a. The POS method analyses the part-of-speech of the combinations of the suggested splits, and picks the most likely one based on occurrences in a corpus. The most frequent combinations are as follows: single word nouns (e.g., *tekst*), two concatenated nouns (e.g., *data-maskin*), single word verbs (e.g., *skrive*), single constituent adjectives (e.g., *trøtt*), two nouns separated by an epenthesis (e.g., *natt-e-søvn*), an adjective plus a noun (e.g., *hellig-dag*), a verb preceding a noun (e.g., *hulke-gråt*), an adjective following a noun (e.g., *hjerne-død*), two adjoined adjectives (e.g., *bitter-søt*), and three consecutive nouns (e.g., *blod-sukker-nivå*).

In effect, this makes the method prefer the same kind of words as the baseline: the most popular class is the single noun (as few constituents as possible), followed by two concatenated nouns (also as few constituents as possible, if a single one is not available). Often, the epenthetic *s* can be interpreted as part of a preceding noun, creating the

genitive case (indicating possession). Lexemes in the genitive case are contained in the full form NST corpus, see Section 4.1.2, so the splitter will prefer this interpretation to that of an epenthetic *s*. Thus, the same kind of low score for the epenthetic test sets are observed.

For the *Lexicalised e* test set, the POS method scores lower than the baseline. It makes mistakes like *boks-e-talent* (ID 31 in Table A.1) because the POS combination noun-epenthesis-noun is more common than the correct interpretation of verb-noun. **Plan-testasjon* (noun-noun) is also incorrectly split here. The correct interpretation, *plante-stasjon*, can be either noun-noun or verb-noun. The reason for why the incorrect noun-noun interpretation is chosen is that upon identifying potential splits, these are added to an ordered list by the length of their first constituent. *Plan-testasjon* is thus added before *plante-stasjon*. Upon having two equally likely candidates, the splitter simply picks the first member in its list of candidates. The order of this list is determined by the original one containing potential splits, thus the one with the shortest first member is chosen.

For *No split*, the POS method makes two mistakes in which it classifies an adjective as a noun-noun combination. Noun-noun is more probable than single-word adjectives, so the former is favoured.

7.1.4. Analysis of the 4-Gram Method Performance

The performance of the 4-gram method is shown in Table 6.1b. The 4-gram method takes the sum of the frequencies of the 4-grams of the suggested constituents counted in a corpus, and picks the suggestion with the lowest sum.

In contrast to Sjöbergh and Kann, the n-gram method is not the best method amongst the ones tested. The method suffers from heavy over-splitting. As in the original article, the method was tested with 4-grams. Furthermore, to reduce complexity, the splitter is restricted to only consider suggestions with constituents of minimum length three characters (excluding epentheses from this rule). The 4-gram of a three-character constituent is the empty string, whose frequency in a corpus is evidently zero. This means that the more short constituents a suggestion has, the more it will be favoured by the n-gram method, leading to over-splitting.

Experimenting with lowering n to 3 worsened the results. The splitter "takes advantage" of the fact that epentheses are exempt from the minimum length of three characters, and favours suggestions with consecutive *e*'s, *n*'s and *s*'s; e.g., **bar-n-e-s-kje* (ID J26 in Table A.4). In the POS method, such suggestions are pruned, because an epenthesis can never follow another, but the n-gram method does not distinguish the strings of epentheses from any other type of string and therefore cannot prune these obviously faulty suggestions. Pruning suggestions with consecutive one-letter strings could help, but would not be an infallible solution because this would still leave two-letter epentheses (*er*, *ar* and *me*). Pruning suggestions with consecutive one- and two-letter strings is also too strict because words like *vennskap-s-by* (sister city) would be disallowed.

Increasing the minimum length of each constituent to 4 characters does help a little bit, but not enough to reach satisfactory performance; the splitter still exploits epentheses

and over-splits the compounds.

Despite low accuracy and precision, the splitter sometimes reaches very high recall. These values can be misleading because they result from the splitter making very few mistakes categorised as false negatives (failing to split when it should have). Since recall is the ratio of correct splits to the total number of splits made (true positives plus false negatives), this fraction approaches 1 as the number of false negatives goes down.

7.1.5. Analysis of the Hybrid Method Performance

The performance of the hybrid method is shown in Table 6.1b. The hybrid method compares the probabilities of the suggestions returned by the POS and the n-gram methods. In Sjöbergh and Kann's results, the n-gram method performs the best out of these two, so this suggestion is used as default. The POS suggestion will only be used if it has a probability that is more than five times larger than that of the n-gram method. It is not clear how Sjöbergh and Kann calculate the probabilities for these two methods, so in this thesis it is implemented in the following way: for the POS method, for each suggestion, the frequency of the POS of each constituent is divided by the total number of samples in the distribution and added together. For n-grams, the analogous operation is done on the compiled distribution of n-grams.

The hybrid method almost always favours the POS interpretation. One exception occurs when the POS method returns probability zero. This happens when the method cannot find the POS combination in its corpus, like for *alkoroboten* (ID J36 in Table A.4). Here, *alko* is classified as an 'unknown' POS (most likely a foreign word) and *roboten* as a noun, but the POS distribution has no such POS combination, so its likelihood is 0. Another example is *vektpiningen* (ID 72 in Table A.1), where the method incorrectly splits it into **vekt-pin-ingen* (noun-verb-pronoun). This POS combination is not found either, so this too returns probability 0. This error was caused by the lexicon missing the head *piningen*[2]. By default, the hybrid method chooses the n-gram interpretation, so the split returned by this method will always be chosen if the POS method returns probability 0.

In *Representative*, the hybrid method splits **basket-misjonær-er* (ID R47 in Table A.9) and **basket-misjon-ære-n* (ID R48 in Table A.9) incorrectly. This is a rare case where the n-gram probability is more than five times larger than the POS probability without the latter one being zero. Here, it happens because NST does not contain an entry of the word *basket* (being the short form of the noun *basketball*), but only the past tense of *å baske* (to fight, to brawl). The verb-noun combination is fairly uncommon, returning a probability of 0.0248, whereas the probability of **basket-misjon-ære-n* is deemed to be 0.0058, which is larger than one-fifth of the former probability.

[2]However, *piningen* is a suffix that only occurs in other compounds like *selvpiningen* and *seigpiningen*, so it is unlikely that it would occur in any dictionary-based corpus.

7.1.6. Analysis of SECOS Performance

The best results Riedl and Biemann (2016) report are precision 0.9698, recall 0.9338, and F_1 0.9515. In Riedl and Biemann's experiments, 12.17 per cent of their compounds are split incorrectly. 55 per cent of these are "under-split", i.e., some additional splits are missed. 38.8 per cent of the errors are split too much, and 5.9 per cent are erroneously split.

SECOS generally does not fare as well in the experiments in this thesis, except for in the test set with no splits. *SECOS* does not seem to have any way to deal with the concept of an epenthesis, and tends to adjoin the epenthesis to the preceding constituent. Thus, it is heavily penalised by gold standards that test this very concept.

In *Lexicalised e* and *Epenthetic s*, all errors are due to *SECOS* not splitting the compound. This probably means that the training corpus *SECOS* uses contains more (or at least these) words. In *No epenthesis* too, the method avoids splitting some of the compounds, but it also struggles with splitting triple nouns, which it tends to split only once. However, there is one especially curious case, in which **skole-tilhør-igheten* (ID R52 in Table A.9) is split incorrectly. The final *-igheten* is not a lexeme and only occurs in words with the suffix *-heten* (-ness, -dom), so it is very peculiar that *SECOS* has decided to split it.

7.1.7. Error Analysis of the Compound Splitters

The largest error source in these experiments is the fact that the gold standards are manually annotated by the author of this thesis (with the exception of *Johannessen*). The author is a native speaker of Norwegian, but not a trained linguist, and errors as well as typos may thus have been made. As for *Johannessen*, the splits in this test set do not follow the conventions of Subsection 7.1.1, so compounds like *bilsyk* and *telefonsvarer* are scored incorrectly because NST contains these words.

As pointed out in Subsection 4.3.2, the training corpus also contains errors. Imperfect tagging causes noise and introduces errors.

Another possible error source concerns Sjöbergh and Kann's methods, which may have been implemented incorrectly. As mentioned in Subsection 6.2.3, despite the clear instructions to use the lowest sum, this causes very dissimilar results for the n-gram method. Thus, an error might have been made when the splitter was implemented for this thesis.

7.2. Evaluation of the Translation System

7.2.1. Evaluation of the Machine Translations

There is a large gap between the translations that the human evaluators consider good and those that they consider bad. The translations that the evaluators like are often translations that are commonly used (but that the translation system formulates without having seen them in training), but there is also disagreement between the native speakers

of which translations they find natural and not. For *overlapper kode*, for instance, equally many evaluators (one-seventh of the participants) find the translation to be either perfect, fluent but unnatural, or incomprehensible. The remaining four out of seven find it incorrect but comprehensible. *[U]fravikelig programmering* and *strenge programspråk* are also translations that receive varying ratings. See IDs 84, 72, and 78 in Table B.4 for the precise ratings of the examples above.

7.2.2. Evaluation of the Control Translations

The control translations are those page titles from Wikipedia that have a Norwegian translations. These translations are presumed to be good translations because they are human-supplied and open for vetting. The results show that this assumption is not always reliable. The distribution of the rated naturalness and degree of meaning conserved of the control translations are shown in Figure 7.1a and Figure 7.1b, respectively. The control translations in these figures are evaluated here, from left to right:

Splitt og hersk-algoritme (divide-and-conquer algorithm) is a perfectly received translation; it scores 100 per cent in natural translation and meaning conserved.

Syntaksfremheving (syntax highlighting) also receives a 100 per cent score in naturalness, and is consistently considered accurately conveying the meaning (all meaning in 6 out of 7 times asked, and most meaning in one case).

Kildekode-editor (source-code editor) is somewhat less accepted; evaluators either find that the translation is natural, but that they do not want to use it themselves, or they find it to be unfit for the domain. Either all or some of the meaning is conserved.

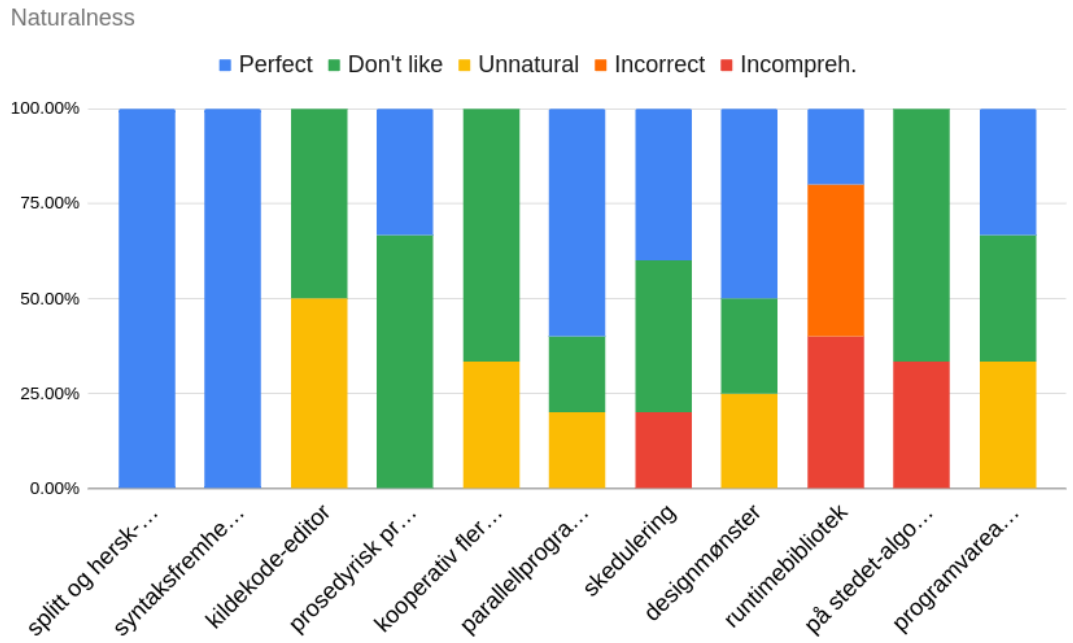
Prosedyrisk programmering (procedural programming) is generally well received, but the majority of the evaluators prefer not to use the term themselves. All evaluators find that the term conveys most or all of the original meaning.

Evaluators are somewhat sceptical to the translation *kooperativ fleroppgavekjøring* (cooperative multitasking), being divided between whether they like the translation but do want not use it themselves, or it being unfit for the domain.

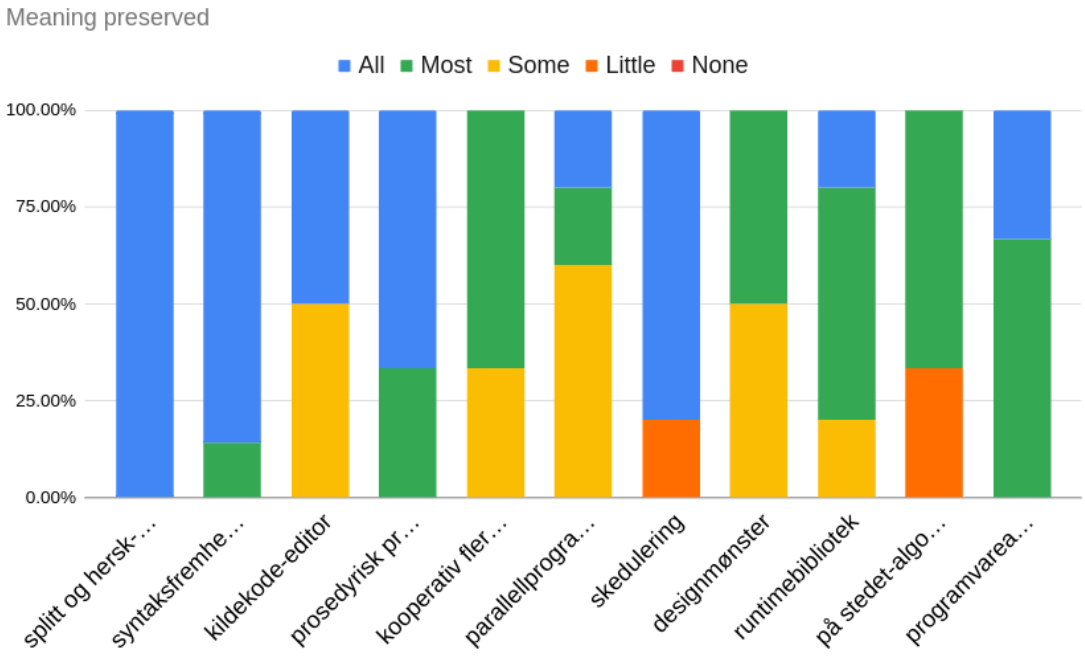
The majority of the evaluators accept *parallelprogrammering* (concurrent computing) as a fluent translation but they are split between whether they find the translation to convey all meaning or just some of it.

Skedulering (scheduling) is the Norwegian translation with the most mixed reception: it is either accepted as a natural translation which the evaluators do not want to use themselves, or it is flat out rejected. One evaluator commented that the translation "seems very unnatural", but others evidently find no problem with it.

Designmønster (software design pattern) is adequately received; 50 per cent of the subjects think it is a perfect translation, and the remaining half is divided between it being unfit for the domain or not wanting to use it. On the other hand, the translation is also rated as lacking some of the original meaning: half the evaluators answer that most of the meaning is conserved, and the other half answers that only some of it is. This is likely due to the fact that the English term specifies that the design pattern applies only to software; the Norwegian translation may be interpreted as the design pattern for anything.



(a) The distribution of the rated naturalness of the control translations.



(b) The distribution of the rated meaning of the control translations.

Figure 7.1.: The distribution of the assessed naturalness and meaning preserved of the control translations.

Runtimebibliotek (runtime library) is the translation the evaluators are the most displeased with: 80 per cent of the evaluators think the word is incomprehensible or barely comprehensible (2 out of 5 answers each), and only one evaluator finds the translation to be fluent. One comment points out that 'runtime' is not a word that is used in Norwegian. Consequently, the evaluators do not think that enough of the meaning comes through; 60 per cent think that most of the meaning is conserved, 20 per cent think that only some of the meaning is kept, and the final evaluator finds that all the meaning is preserved.

På stedet-algoritme (in-place algorithm) is adequately received and is rated as fluent and conserving most of the original meaning, in the majority of the cases, but the evaluators prefer not to use it.

Finally, *programvareavhengigheter* (coupling) is considered a good translation but there is disagreement between whether the translation is fluent or not; 33 per cent of the evaluators (1 out of 3) answer that the Norwegian translation is perfect, 33 per cent answer that it is a good translation, but that they would not use it themselves, and the remaining third regards the translation as unsuitable for the domain. On the other hand, the evaluators agree that the meaning is conserved (33 per cent think all of the meaning; 66 percent think most).

7.2.3. Error Sources

Errors in the Training Data

The main error source of the machine translations is the training data; teaching the translation system erroneous translations will cause it to output erroneous translations. This may happen in the form of bad alignments or lexical ambiguity. 'Dead code' (ID 83 in Table B.4) is for instance poorly translated into *encellede kode* because of an incorrect alignment extracted from line 21835 in the training corpus: here, the English line "Inactive yeasts and other dead single-cell micro-organisms" is paired with the Norwegian equivalent "*Inaktiv gjær, andre døde, encellede mikroorganismer*". 'Dead' should have been aligned with *døde* but was in fact aligned to 'single-cell', which causes the above error. Note that even if the words had been properly aligned, the translation would still be incorrect due to a disagreement between a plural adjective and a singular subject: **døde [plural] kode [singular]*. It is certain that the translation system would output this particular form of the adjective because the lexeme 'dead' appears only in this context in the corpus. Conjugational errors is further discussed in Subsection 8.2.8.

There is also the issue of formatted text in the corpus. For instance, on line 52999, the English line 'External suffocation hazards' is paired with the Norwegian equivalent '*F a r e f o r k v e l n i n g s o m f ø l g e a v y t r e b l o k k e r i n g a v l u f t v e i e n e*'. Similar formats concern multiple lines. Like mentioned in Subsection 5.3.3, SRILM of Stolcke (2002) is used to build language models with n-gram models in this project. It is not known how SRILM determines what distinguishes words in the n-gram models, but a plausible assumption is that it simply separates letter sequences and spaces. If this is the case, the language model will try to align English words to single Norwegian

letters here. This alignment will clearly be substandard. Indeed, translating 'external suffocation hazards' gives an unsound Norwegian translation: **ytre blokkering farer*.

Another major issue with the corpus is that it contains large portions of text in foreign languages. Most commonly, these extracts refer to legislature in other countries and their corresponding language. This can introduce noise into the language model because OBT (Oslo-Bergen Taggeren, see Section 4.1.2) tries to tag it as Norwegian text. Often, OBT is unable to understand the word and simply tags it as `ukjent` (unknown), but if there is coincidental morphology between the language and Norwegian lexemes, OBT may tag it accordingly. For instance, on line 14892, the French phrase 'sous la surveillance des communes' is tagged as `sous|subst la|verb surveillance|ukjent des|det communes|subst`. In particular the French article 'la' is interpreted as the past tense of the Norwegian verb *legge* (to lay). The foreign extracts are the same on the English and Norwegian side of the corpus, so if an English word has the same orthography as a French one, it is possible for the translation system to learn the same French word as a Norwegian translation. In the same way, the POS (part-of-speech) tagging of the Norwegian side can be exposed to noise from the faulty tagging of a foreign language. In the above example, 3-grams like 'noun verb unknown', 'verb unknown determiner', and 'unknown determiner noun' are introduced. Whereas other, more common 3-gram combinations from Norwegian text probably outweigh these data points, this kind of training data is unfit for an SMT (statistical machine translation) system.

Errors Made by the Tagger

There are certain compounds that the tagger is not equipped to deal with. This includes coordinated compounds (e.g., *Oversettelses- og tolkevirksomhet*), compounds with three consecutive identical consonants (*Busstoppstolper*, not **Bussstoppstolper*), compounds containing proper names (e.g., *Europa-avtalen*), and compounds containing abbreviations (e.g., *VVS-arbeid*, *CPV-referansenummer*). Abbreviations are frequently used in the computer science domain, and the splitter's inability to handle or extract parts from such compounds weakens its learning ability. The equivalent is true for proper names. Furthermore, a tendency for abbreviations and proper names is to be followed by a hyphen; the tagger does not handle these either.

Reducing triple consecutive letters to two in the output is trivial, but has not been implemented. On the other hand, learning constituents from such compounds requires more complex processing during splitting and is not handled either.

Coordinated compounds are a special type of compounds that have not been discussed in this thesis. Coordinated compounds are consecutive compounds that share a head, but instead of explicitly writing it in both words, the first one is shortened to the modifier that differs from the latter compound. One such example is found on line 26672 in the training corpus: the head *virksomhet* is shortened from the first compound in *[o]versettelses- og tolkevirksomhet*; i.e., not **[o]versettelsesvirksomhet og tolkevirksomhet*. As can be seen in the above example, these compounds are multiple-word expressions. The splitter is unable to handle these because only single-word compounds are analysed. Strictly speaking, this does not cause any errors in the translation sys-

tem, but it loses data on constituents that could have been learnt: the output in the above example, is tagged as `Oversettelses-|subst og|konj tolk|subst-modif e|subst-modif virksomhet|subst`. This means that only *tolk* is learnt as a potential constituent, and that *oversettelses* is lost.

Lastly, there is the issue of unknown compound parts: if a constituent is not contained in the lexicon that the splitter uses, the splitter is unable to extract any constituents from the compound. Like for the other compound types, this does not produce an error in the translation system, but it prevents the splitter from learning constituents from that compound. However, from the perspective of compound splitting and POS-tagging, these are indeed errors.

Potential solutions to the flaws pointed out in this section will be discussed in Section 9.3.

Errors Concerning Lexical Ambiguity

Lexical ambiguity is one of the main challenges in machine translation; it is very hard to instruct a machine that a word should be translated in a certain way under given circumstances, and how to distinguish these circumstances. One such example is the translation of 'yield': in the computer science domain, it is a term used in multithreading that describes the event where a processor is forced to give up the control of the currently running thread. This meaning is not captured in the training corpus; instead it was found in the context of legislation concerning cigarettes. The translation system has no way of distinguishing the 'yield' used in this context from any other domain, so the probabilities of the observed occurrences are calculated and the best estimate is output. A more in-depth discussion of this issue and other examples is given in Subsection 8.2.2.

Human Errors

Blindly trusting the evaluators is also a potential error source. Despite being native speakers, grammar and orthography are not inherent capabilities; they are skills that must be trained and maintained not to make linguistic errors. In particular, the error of writing compounds as separate words is so common that it has its own name in Norwegian: *særskrivingsfeil*[3]. Here, it is demonstrated with the translation of 'graphics libraries' into *grafikk biblioteker*. This translation is incorrect with regards to not being written as a closed compound; i.e., *grafikkbiblioteker*. However, in 40 percent of the ratings (2 out of 5), this translation is assessed as perfect. This shows that even though the translations are rated as good in the experimental results, they have to be verified by professional linguists to ensure complete reliability.

[3]Not to be confused with *orddelingsfeil* which describes putting a hyphen in the wrong location when a word is split over two lines.

8. Discussion

This chapter discusses the results evaluated in the previous chapter. Section 8.1 discusses the results of the experiments on the splitters, and decides which one is more appropriate for the task of tagging decomposed compounds in a translation task. Section 8.2 is concerned with how well the translation system as a whole has achieved the goals presented in Chapter 1. Weaknesses and errors of the system are also pointed out, which make the basis for the suggestions for Future Work in Section 9.3.

8.1. Discussion of the Implemented Splitters

In general, the splitters evaluated on mixed test sets do not come close to the accuracies reported by the publications from Section 3.1. However, directly comparing the results from Subsection 6.1.3 against the other publications must be done with extreme caution. It proved impossible to identify most of the test sets used in the other articles, so it is very hard to draw any conclusions without knowing what kinds of compounds these evaluate. The hybrid method fittingly illustrates this point: it has been shown that this method works very well on the test sets containing lexicalised *s* and no epentheses. If the experiments conducted were performed only on these kinds of compounds, it is likely that more comparable results would emerge. In contrast to previous research, the experiments here are designed to reveal what kind of compounds the different splitters perform well for. The results must therefore be evaluated in context.

The following list summarises the results of splitting compounds as reported by the articles discussed in Section 3.1.

- Johannessen and Hauglin (1998): 97.6 per cent success rate[1] on Norwegian compounds.
- Koehn and Knight (2003): 93.8 per cent precision, 90.1 per cent recall and 99.1 per cent accuracy translating from German to English.
- Sjöbergh and Kann (2004): 94 per cent accuracy on splitting Swedish compounds.
- Alfonseca et al. (2008): 83.56 per cent precision, 79.48 per cent recall, and 87.21 per cent accuracy for German compounds. 88.13 per cent precision, 93.05 per cent recall, and 90.40 per cent accuracy for Norwegian.
- Thorsen Ranang (2010): Reports a success rate of 87.5 per cent when attempting to split Johannessen and Hauglin's Norwegian compounds.
- Macherey et al. (2011): Increases BLEU score for translations from Norwegian into English by 1.81 per cent.

[1] Based on the discussion in Thorsen Ranang (2010, p. 68)

- Clouet and Daille (2013): 93.04 per cent splitting precision using only rules and no corpus in German.
- Stymne et al. (2013): 20.9 per cent precision, 57.2 per cent recall, and 92.1 per cent accuracy, and BLEU score 19.0 for German compounds using the *arith* method.
- Lindbråten (2015): 93.68 per cent accuracy on easy test set and 85.36 per cent on Johannessen and Hauglin's difficult compounds.
- Riedl and Biemann (2016): 96.98 per cent precision, 93.38 per cent recall, and 95.15 per cent accuracy.

It is hard to directly compare the results reported in the articles discussed in Section 3.1, as they have been applied to different languages, and on different training and test sets. However, the reuslts indicate that there are many different ways to split compounds that produce good results, and that the methods work for a range of languages.

The previous work also indicates that the optimal splitting strategy depends on the intended application. As pointed out by Koehn and Knight (2003), even though one method performs very well in terms of splitting a compound into its correct constituents, the same method might not do well in a machine translation task. This is supported in Stymne et al. (2013), where the intrinsic splitting strategy does not correlate with the translation quality. Furthermore, Alfonseca et al. (2008) observe that the supervised model they test performs better than the unsupervised ones, although it is more costly. This means that if the compound splitting is part of a larger system, it might be worthwhile choosing the algorithm based on computational cost and/or the specific features needed, as opposed to just accuracy scores. This is an especially important observation for the compound splitter's application in this project, where it is used in a translation system.

A tendency of several of the methods implemented in Section 5.1 is to suppress epentheses whenever possible. This often forces a noun followed by an epenthetic *e* to be interpreted as a verb, or for a noun to take the genitive case when followed by an *s*. In practice, these combinations seem unnatural to a native speaker, and they would rarely be interpreted in this way. It can be hypothesised that including more information about the POS (part-of-speech), e.g., verb tense as well as the number and case of nouns, could further improve the POS method. This is supported by the theory of Johannessen (2001) that constituents cannot contain inflected forms. This approach requires resources that list lexemes and their stems. The NST database, see Section 4.1.2, has fields for indicating whether an entry is an inflected form and, if so, what the root form is, but this information was not used in this project.

8.2. Discussion of the Translation System

This section discusses different aspects of the translation system that affects the output. It is also discussed whether the characteristics demonstrated by this system are consistent with those of similar translation projects and the reasons for these tendencies.

A major concern is that many of the translations that are evaluated as acceptable merely appear correct on the surface, but are in fact problematic.

8.2.1. Effect of Out-Of-Vocabulary Words

Out of Vocabulary words, OOV, are words that are not contained in the translation system's vocabulary file (and by extension, the translation table). They are simply words that the system has not been trained to handle. For the purposes of this thesis, Moses, see Subsection 5.3.3, is configured to pass OOV words unaltered through the decoding, but tagged with the unknown tag in the output, e.g., `emulator|UNK|UNK|UNK`.

The translation system's performance suffers whenever a word that was not seen during training is encountered. This is typical for SMT (statistical machine translation) systems. However, the system is indeed able to recognise in-vocabulary words in compounds and treat them as constituents from which it can construct new compounds; i.e., compounds that were not observed during training. This translation is well received, either being evaluated as perfect or fluent but with a personal preference of not using it. The conservation of the meaning is equally divided between retaining all or most of the original semantics, but the compound is not found in the training corpus. Inspecting the tags Moses outputs for this translation, it becomes clear that the constituent *beskyttelses* has been observed as noun modifier, and that the postprocessing attaches it to the following lexeme to form a compound. Despite being well received, the constituent *beskyttelses* is erroneously tagged: the trailing *s* is in fact an epenthesis, and not the genitive form of the word, which is its current meaning. However, since this thesis only seeks good translation quality, the incorrect splitting is not considered a major problem.

More interestingly, the final constituent, *byte*, is not translated at all. In Moses' output, it is tagged with the unknown tag, `UNK|UNK|UNK`, indicating that a corresponding entry was not found in the translation table. Yet, native speakers assessed this as a good translation. As mentioned in Section 1.2, Norwegian computer-related vocabulary has traditionally employed a great deal of loanwords. Many of these loanwords are typically spelled exactly the same as in English and simply pronounced differently. It is therefore likely that, since the preceding constituent is an easily-recognisable Norwegian word, the reader is inclined to interpret *byte* as the Norwegian equivalent and accept the full compound.

However, this is not a generally reliable translation approach. As mentioned above, Norwegian has a long tradition of employing English loanwords with little to no modification except for in pronunciation. Not all languages have the same approach: in Finnish, for instance, the computer byte is translated as 'tavu'. If the translation systems passes such foreign words unaltered, the output will be incomprehensible. The translation works in this case because it employs a frequently used loanword which is understood with the same orthography as it has in English.

The issue is accentuated when translating between languages with different character sets. For instance, if the system translated from Russian to Norwegian and used the unaltered input when unknown words were encountered, it would effectively concatenate a Russian term to the preceding output. This would clearly not produce a comprehensible

translation (**beskyttelses*байт).

In short, the approach only works in some particular cases of English-Norwegian translations because of coincidental vocabulary.

8.2.2. The Effect of Lexical Ambiguity

Lexical ambiguity is a translation issue that resembles the OOV problem: the translation table contains an entry for the input phrase, but that term translates into different terms in the target language, most often based on the contextual domain. Lexical ambiguity is a problem in the computer science domain too. Despite being trained on a corpus with computer vocabulary, the system produces some extremely poor translations because of domain differences. The most obvious error is that of 'yield', translated as 'the content of tar' (ID 22 in tables B.1, B.3, and B.4). This translation is learnt based on a section that concerns cigarettes in the file `nb_no_eu_utvidelser.tmx`. The translation system has no way of distinguishing the 'yield' used in this context from that of computer science or any other domain, so the probability of the observed occurrences is calculated and a best estimate is output. In this particular case, the only context in which 'yield' is used as a noun is when discussing tobacco regulations, hence the corresponding translation.

It is especially challenging for the translation system to determine the domain in these experiments because the test set presents it with terms without providing any context; without any preconfigured knowledge of the domain, this makes it close to impossible for any translation system to determine what domain this particular term belongs to. Thus, the experimental setup may be partially at fault for having aggravated the lexical ambiguity.

Another lexical error the translation system makes is translating 'data' into *opplysninger*. This is the case for *organisasjonen for laveffektopplysninger* (ID 50) and *funksjonelle opplysninger konstruksjoner* (ID 91). These translations score poorly both in terms of naturalness and meaning preserved. Indeed, in some contexts, translating 'data' into *opplysninger* is correct, but for the computer science domain, the Norwegian equivalent *data* is most commonly used. The translation system is trained on legal documents, where the term *opplysninger* is more commonly used as a translation for 'data'. For instance, on line 37506 in the training corpus, the English translation 'Data generated or processed when supplying the communications services concerned refers to data which are accessible' is paired with the Norwegian equivalent '*Opplysninger som genereres eller behandles ved levering av de aktuelle kommunikasjonstjenestene, gjelder bare tilgjengelige opplysninger*'. Here, two instances of the alignment of 'data' and *opplysninger* are observed, increasing the probability that this is a good translation for these lexemes.

On the other hand, the translation of 'data' into *data* is not completely lost: the translations *data-orientert konstruksjon* (ID 51) and *dataoverføring gjenstand* (ID 85) use the same lexeme, indicating that this alignment has indeed been learnt. This is the case because 'data' is sometimes translated as *data* in the corpus. For instance, on line 57277, '** Eksterne data betyr data som er beskyttet av en atskilt signatur som ikke inngår i e-innholdet til CAdES-signaturen*' is the Norwegian translation of '* External data means data protected by a detached signature that is not included in the CAdES

signature eContent'[2]. Nevertheless, the usage of *opplysninger* is about twice as common as *data* (outweighed by 2,376 to 1,018 occurrences in the corpus). The models reflect this, causing Moses to output the seemingly most probable translation.

8.2.3. Effects of the EPOS Tagset

As described in Subsection 5.3.2, the Norwegian data is tagged using POS (part-of-speech) tags and special `X-modif` tags when compounds are not recognised by OBT (*Oslo-Bergen-Taggeren*, see Section 4.1.2) and treated accordingly by the custom decompounder. This tagset is called the EPOS tagset (extended part-of-speech). It is clear that the compound splitting and marking of constituents allows the translation system to output novel compounds. The terms *konkurranseprogrammering*, *beskyttelsesbyte*, and *vitenskapelige programspråk* (IDs 55, 56, and 77 in tables B.1, B.3, and B.4, respectively) are examples of compounds that are well received by the evaluators, and that are not observed in the training data. Conducting a Google search, with the terms enclosed in double quotation marks to search for the exact string, yields relatively few results: *konkurranseprogrammering* gives about 191 results, *beskyttelsesbyte* gives a single result, and *vitenskapelige programspråk* gives none at all. Searching for the decomposed output (without quotation marks) gives a magnitude of results: about 180,000 for "beskyttelses byte", 675,000 for "vitenskapelige program språk", and 1,160,000 for "konkurranse programmering". It can thus be argued that the system does output truly novel compounds, and that this is thanks to the EPOS tagset and the postprocessing it enables.

On the other hand, the EPOS tagset restricts the translation system to only outputting compounds with modifiers that have been observed in other compounds during training. This is the reason for why *grafikk biblioteker* is split and not translated into a single compound, like *lydbiblioteker* is. *[G]rafikk* has only been observed with the tag `subst` (noun), meaning it will never be concatenated to the following output token. A modifier may use this noun as its head, though.

8.2.4. The Effect of Preprocessing

The preprocessing step involves tokenising the English input before it is supplied to the decoder. For the purpose of the experiments presented in this thesis, this only involves terms, as opposed to fully punctuated sentences. The tokenisation therefore merely consists of lowercasing the input terms. As mentioned in Subsection 5.3.4, words with hyphens are also split. This is done because initial experiments showed that the same problem as Wang et al. (2014) point out was prominent: hyphenated compound words, like 'slow-growing' and 'easy-to-use' occur quite frequently. Wang et al. only observe this phenomenon for medical texts, but Vegnaduzzo (2009, p. 86) remarks that "from a linguistic point of view, the hyphen in potential productivity patterns seems to signal the awareness of the writer/speaker regarding the 'novelty' of the complex adjective". Being a research area in constant development, there is reason to believe that such novel complex

[2]The asterisks here are not indications of ungrammatical text, but a direct citation from the corpus.

compounds are likely to emerge in this domain as well. The effect of not splitting on hyphens is OOV words, which impedes the phrasal translation despite the compounds being composed entirely of known words. The input is therefore split on hyphens to allow the possibility of translating into similar structures.

This approach is partially successful: *data-strukturert programmeringsspråk* (translated from 'data - structured programming languages', ID 63 in tables B.1, B.3, and B.4), *rolle-orientert programmering* ('role - oriented programming', ID 76), and *prototype-basert programmeringsspråk* ('prototype-based programming languages', ID 68) are all translations that are rated as perfect by more than two-thirds of the evaluators. However, *nummer-av-nummeralgoritmer* ('digit - by - digit algorithms', ID 14) and *ikke-adusert kode* ('non-malleable code', ID 89) are not found to be good translations. Examples include 'two-thirds' (*to tredjedeler*), 'father-in-law' (*svigerfar*), and 'well-known' (*velkjent*). It is therefore not completely reliable to translate hyphen-by-hyphen and special processing of this type of compounded terms is probably needed in the training of the translation system.

8.2.5. The Effect of Postprocessing

To reassemble compounds after translation, words tagged `X-modif` are concatenated to the following word. Hyphens (if any) are also stripped of any preceding or trailing spaces. This approach is too simplistic. As mentioned above, the EPOS tagset only allows words tagged as modifiers to be concatenated to the following word and form compounds, and the postprocessing step reinforces this rule.

Stripping output spaces from hyphens does allow forming some compounds that would be illegal under the modifier tag. For instance, *matematikk-biblioteker* is tagged as `matematikk-|subst biblioteker|subst`, meaning it is only concatenated as a compound because the first noun ends with a hyphen. However, if coordinated compounds were encountered, this approach would fall through. For instance, translating 'mathematical and geometric libraries' needs a space before the conjunction *og*: *matematikk- og geometribiblioteker*; directly interposing the hyphen between the two first lexemes does not give the correct orthography.

Another major flaw of the postprocessing step is that it can concatenate a modifier to a head belonging to a different part-of-speech. This happens in 'data-programmot' ('data-directed programming', ID 71 in tables B.1, B.3, and B.4). This output is tagged as `data|subst-|strek program|subst-modif mot|subst-modif`. *Mot* is a preposition meaning 'against' or 'counter', interpreted as a translation for 'directed'. This translation is problematic because it creates a nonsensical expression (rated as incomprehensible by two-thirds of the evaluators). In addition, the POS combination of noun and preposition is very uncommon, and a modifier should not be the final constituent of a compound. It is possible to implement some additional rules preventing modifiers from being concatenated to words of other parts-of-speech, but it is more likely that this problem shows that this postprocessing approach is too simplistic.

Other approaches to postprocessing that could be explored are Stymne et al. (2013), who use sequence models, and Cap et al. (2014), who use underspecified representations.

8.2.6. The Effect of Contrasting Training and Test Data

The discrepancy between the form of the training data and the test data constitutes a challenge for the translation system; the system is trained on a full-text corpus, whereas the test set is composed of single lexemes or terms representing a single concept. As already pointed out, the system struggles with lexical ambiguity. Since the test items give no context, it is unreasonably hard to determine the domain and choose an appropriate translation. In contrast, the evaluators were instructed about which domain they were assessing and made ratings based on this information.

Furthermore, in English, word forms often coincide. This is for instance the case for the present tense of a verb and the plural noun form of the same word; e.g., strings ['to string', present tense] and strings [noun 'string', plural]. It may also happen that an adjective is derived from a verb. This is exactly the case for 'duplicate code' (*overlapper kode*, ID 83 in tables B.1, B.3, and B.4). Here, 'duplicate' is an adjective, but since the lexeme is identical to the infinitive and the imperative tense of the corresponding verb, the translation system mistakes it for a verb, and translates it accordingly: `overlapper|verb kode|subst`. This shows that the experiments were inadequately constructed, introducing context-based noise which could have been filtered out. However, in this particular case, one can determine that providing more context to clarify that 'duplicate' has to be an adjective and not a verb would not help because the training corpus does not contain the adjective form. The system would thus not be able to produce this derivation despite the models favouring this POS sequence.

The form of the test set also affects the evaluators: for the translation *blokk* ('block', ID 30), one evaluator commented that it was unclear to them whether 'block' referred to the verb or the noun form of the lexeme. They found the verb form to be the most likely usage and rated it accordingly, giving it a lower score, because the Norwegian translation has the form of a noun. The Wikipedia entry the translation is based on concerns lexical structures of grouped source code; i.e., a noun, but the evaluators were not informed of this. To minimise these issues, the evaluators should have been supplied with a definition of the translation they were rating. The test translations should also be translated in context. However, this opens up for other issues, for instance if the context contains other OOV words.

The English variety that is used in the training corpus and during testing is also an area of concern. The training corpus is based on British English, but the test items collected from Wikipedia often use American orthography; e.g. in 'program optimization' (ID 47) and 'data organization for low power' (ID 50). Without being trained on the latter standard or having other forms of morphological analysis that can relate the variants, the translation system will treat these lexemes as two different vocabulary items. This has the same effect as OOV words.

In short, training the translation system on full sentences but testing it on ambiguous terms with little context affects the translation performance negatively.

8.2.7. The Effect of Corpus Size

It is unclear whether the effect of a larger corpus is positive or negative in terms of translation quality. In initial experiments with the translation system, the term 'time travel debugging' was translated to *tidsreisedebugging*. *Tidsreise* is the conventional translation of 'time travel', and *debugging* is often employed as a loanword (although pronounced with a Norwegian accent). The epenthetic *s* between *tid* and *reise* is correct, and concatenating this compound to *debugging* without any further epentheses is also natural. However, adding more texts from the 'Translation memories from The Ministry of Foreign Affairs of Norway' (in particular, that of `nb_no_off_innkjop.tmx`, see Subsection 4.3.3), to the training corpus causes the translation of 'time travel debugging' to be translated as *tids reise debugging*. This formulation would not be accepted by native speakers due to the separating spaces.

On the other hand, 'hash tree' is improved: before adding all seven texts to the corpus, it was translated as *Hash frukt som vokser på trær* ('Hash fruit that grows on trees'). After adding the additional text it became translated as `Hash|subst trær|subst`. This translation was not rated by native speakers, but it is clear that fruit has nothing to do with the above type of trees, and that the latter translation probably is better[3]. *Hash* is a common loanword that is likely to be accepted by many native Norwegians, but analysis of the corpus shows that this translation is learnt by mistake. The letter sequence 'hash' occurs in three instances in the training corpus: once in *hash-ekvivalens* on line 57252, once in *hash-beregningen* 57263 and once as *Hash* on line 57266. The two first instances are not split, so *hash* can only be learnt from the last occurrence. The problem is that line 57266 is an extract from English legislation, meaning that the translation system is taught English-English translations. Evidently, this is undesirable. Despite producing an acceptable translation in this case, the approach is not reliable because it causes the same issues as discussed in Subsection 8.2.1.

An example of a translation that was truly improved is that of 'Symbolic programming' (ID 32 in tables B.1, B.3, and B.4). Prior to adding all seven texts, it was translated as `Symbolic|UNK|UNK|UNK program|subst-modif`. Increasing the corpus size made it a highly rated translation, viz., *symbolsk programmering*. All the assessments of this translation are either perfect or fluent (but not wanting to use it). Equivalently, meaning was rated as fully or mostly preserved.

This shows that there is no direct relationship between translation quality and the size of the training corpus. On the other hand, the content and its appropriateness for the domain has a much larger effect on the quality of the output translation.

8.2.8. The Effect of Compound Splitting

In this thesis, the basic principle that allows forming novel compounds is learning modifying constituents during the training. However, the fact that the compound splitter's lexicon contains a certain constituent does not guarantee that the decomposed translation model will learn this particular constituent. Prior to the compound splitter

[3] Although it would have to be concatenated to be completely correct.

analysing compounds, the Norwegian side of the corpus is tagged with OBT. The compound splitter only analyses compounds that are tagged as unknown compounds (`samset-analyse`) by OBT. If a compound is recognised as a compound, but is contained in OBT's lexicon, OBT will tag it with the `samset-leks` tag, which is not analysed by the splitter. This is the case for the singular and plural form of the word *programmeringsspråk* (programming language). In Norwegian, *programmeringsspråk* is a neuter noun, thus conjugated the same way in its singular indefinite and plural indefinite form. However, the singular 'programming language' does not occur in the English side of the corpus, so only the plural alignment is learnt. Consequently, in *vitenskapelige programspråk* (ID 77 in tables B.1, B.3, and B.4), *strenge programspråk* (ID 78), and *-programspråk generelle formål* (ID 93), *programspråk* is tagged is `program|subst-modif språk|subst`. In contrast, *programmeringsspråk* is always tagged as a single noun when it is output as a translation of the plural 'programming languages'.

Whereas this is consistent with the argumentation in Subsection 7.1.1, it is problematic that the same noun is translated differently based on its grammatical form. *[V]itenskapelige programspråk* and *strenge programspråk* are indeed highly rated translations, but the assessments are made of words in isolation; using different forms in the same text may be confusing to the reader.

Having the splitter analyse compounds tagged with the `samset-leks` tag in addition to the `samset-analyse` tag is not a viable approach. It would provide consistency for the plural and singular translations in this particular case, but many compounds that are lexicalised are exocentric; i.e., their meaning is not equal to the sum of their constituents (see Subsection 7.1.1). One such example is *derfor* (because, therefore, thus). Etymologically speaking, *derfor* is equivalent to 'therefore', being composed of *der* and *for*, but those constituents cannot freely be used as modifiers, like noun modifiers largely are. It is possible to adopt the approach of Stymne et al. (2013, p.1093) which requires that the constituents be content words, but this would eliminate the possibility of forming certain compound translations. *[P]å stedet-algoritme* ('in-place algorithm', ID 109 in tables B.2 and B.5) is an example of a translation that would be eliminated because *på* is a preposition. It is more likely that a grammar module that enables conjugation based on the compound head would be more effective at treating varying grammatical forms consistently.

8.2.9. The Effect of Domain Adaption

The main domain adaption technique used in this project is using specialised vocabulary from the domain the items of the test set belong to. It is evident that the training vocabulary affects what kind of compounds the translation system can output. The training data does indeed contain domain-specific vocabulary like *programmering*, *syntaks*, *programvare*, and *emulering*, but it struggles with lexical ambiguity (see Subsection 8.2.2) and lacks some essential vocabulary. 'Optimization'/'optimisation', 'error', 'computational', 'memory', 'recursion', 'dependency', and 'sequential' are some examples of relatively basic computer-related terms that the translation table does not contain. Note that these words are not necessarily in the test set, but were probed manually.

On the other hand, the high ratings of some of the unknown tokens show that the computer science domain is rather lenient in terms of accepting a relatively high number of untranslated English words. Again, as discussed in Subsection 8.2.1, this is not a generally reliable approach, and more computer-specific vocabulary needs to be introduced to achieve acceptable results.

On a broader level, one evaluator pointed out that despite having worked in the IT industry for several years, they did not recognise any of the English terms presented. The evaluator specified that they worked in the UX (user experience) field. This shows that 'computer science' may be too much of an open domain, and that the domain may benefit from being even more specialised; e.g., into subcategories like 'theoretical computer science', 'programming', 'low-level computer components', and similar.

8.2.10. Producing Known Compounds

Throughout this thesis, 'producing unseen, novel compounds' has been understood as creating compound translations for English terms that do not already have a Norwegian translation. However, it is fully possible for the translation system to produce a translation that was not observed during training, but that is identical to an already-established Norwegian translation. In particular, this is the case for *hardkoding* (hard coding, ID 17 in tables B.1, B.3, and B.4, tagged as `hard|subst-modif koding|subst`), *søke-algoritmer* (search algorithms, ID 16, tagged as `søke-|subst algoritmer|subst`), *lydbiblioteker* (audio libraries, ID 42, tagged as `lyd|subst-modif biblioteker|subst`), and *ressursforvaltning* (resource management, ID 29, tagged as `ressurs|subst-modif forvaltning|subst`). Whereas these translations do not fit the problem description, producing the same translation as humans strengthens the overall confidence in the translation system. On the other hand, these translations can be seen as "easy" translations, seeing as they have no epentheses, and are simple concatenations of the corresponding English constituents, in the same order.

8.3. Limitations of the Translation System

A compound can theoretically contain an arbitrary number of constituents. Disregarding translations that are concatenated due to hyphens, the translation with the most constituents is *direktebuffersamsvar* with three constituents (ID 19 in Table B.3). However, there is no guarantee that a closed compound is the best translation available.

One of the evaluators' comments regarding the translation *unntak håndteringssyntaks* (ID 58) pointed out that this translation is incorrectly split, but that the correct composition, *unntakshåndteringssyntaks*, would be excessively lengthy. This shows that despite the theoretical possibility that a compound can contain an arbitrary number of constituents, this will likely cause the naturalness to decrease. The implementation presented here does not have any restriction on the number of constituents or other functions that give negative weights to such unfavourable translations. Other studies, e.g., Bungum and Oepen (2009), have looked into how to generate templates from closed

compounds using function words and reordering (albeit in the opposite direction; i.e., from Norwegian to English). The methodology's ability to preserve naturalness in cumbersome translations could be explored.

Furthermore, the translation system is only capable of producing compounds whose constituents are translations of the parts of the English terms. The constituents are translated as far as known translations are contained in the translation system's phrase table, meaning that the translation system produces semantic calques. These are translations where some additional meaning from the source term is transferred to an already-existing translation. There is a multitude of English-Norwegian semantic calques in the technology domain; e.g., *harddisk* (hard drive), *hodetelefoner* (head phones), and *mus* ([computer] mouse), which could be an indication that this is a good translation approach.

On the other hand, calques fail to convey the meaning of a translation when the semantics are not explicitly contained in the original term. An article in *Termposten*[4] (a magazine published by *Språkrådet* "for everyone working with or having an interest in terminology") describes the process in which four researchers attempt to create new terminology in the field of cellular and molecular biology. The article describes the researchers' discussion of the English term 'locus' (the position on a chromosome where a particular gene or genetic marker is located), and how it leads to the suggested Norwegian translation *genplassering* (literally 'gene placement'). Whereas the researchers are satisfied with this suggestion, it is impossible for the translation system to produce this kind of output given its current configuration. *Genplassering* is an endocentric compound that identifies the properties that the English term conveys, consequently forming a novel compound independently of the original term's constituents. The translations made in this thesis are based on the English constituents, so unless the term 'locus' was known in advance (in which case the need for a translation disappears), no meaningful output could be generated from this term.

The Icelandic language policy uses a similar approach, in which modern terms that are often adapted as loanwords in other languages are given distinctive Icelandic names, see Iceland Ministry of Education, Science and Culture (2001). These terms are often in the form of compounds of already-existing Icelandic lexemes or portmanteau words (words composed of parts of multiple other words or their phonemes). One such example is the word for computer, 'tölva', composed of 'tala' (number) and 'völva' (prophetess).

Again, the translation system presented here is not able to produce such compounds and/or portmanteaux, and it is possible that a given term does not have an acceptable Norwegian translation that can be used as a calque of the English constituents.

[4] `https://www.sprakradet.no/Vi-og-vart/Publikasjoner/termposten/termposten-22019/visste-du-at/`

9. Conclusion and Future Work

This chapter is the final part of this thesis. Section 9.1 lists the contributions made, and in Section 9.2, the conclusions of this project are presented. Finally, suggestions for improvements and future work on the splitter and translation system are made in Section 9.3.

9.1. Contributions

In Chapter 1, three research questions were posed.

Research question 1 *How applicable are current approaches of domain-specific translation to Norwegian translations?*

In this thesis, domain-adaption is achieved by using domain-specific texts as the training corpus. As pointed out in Subsection 8.2.9, this does introduce some vocabulary that would not have been available otherwise, but the corpus also lacks some essential vocabulary which shows that the training corpus does not provide satisfactory amounts of domain-specific vocabulary. One solution to this could be to narrow down the 'computer science' domain and focus on finding more specialised vocabulary in some sub-domain. On the other hand, this requires that a collection of parallel text on this topic is available. Finding appropriate parallel corpora has been one of the main issues throughout this thesis, and is not guaranteed for a sub-domain, either.

Research question 2 *Does the split-and-merge approach give comparable results for Norwegian in terms of producing novel, unseen compounds?*

The split-and-merge strategy was used during training, in which compounds were split by the decompounder, and in the postprocessing step, where modifiers were joined to their presumed heads. As mentioned in Section 5.2, the chosen splitter struggles with explicitly splitting epentheses from preceding nouns, but this does not seem to affect the output translation; no evaluators commented that there were too many or too few interposed letter sequences in the closed compounds. The merge step has several issues due to being oversimplified, but even so, it is capable of producing unseen compounds. A more sophisticated postprocessing is necessary to increase the quality of the novel compounds, but the underlying principle is viable. Thus, there is nothing in this project that suggests that the split-and-merge approach cannot be used for Norwegian compounds, too.

Research question 3 *How natural do native speakers find the novel translations?*

Evaluating novel translations with automatic frameworks that compare against reference translations is impossible because there are no translations to compare against. For this reason, fluent Norwegian speakers were asked to rate the novel translations. As shown in Subsection 6.2.3 and Chapter 8, there is a large discrepancy between the assessments of different translations, but also among the evaluators for the same translation. Rating human translations as control translations showed that this discrepancy is not unique to machine translations, and that these assessments cannot be trusted blindly. Furthermore, a relatively low number of evaluators participated in the surveys, weakening the statistical significance of these results.

The research questions were introduced to provide evidence for the following goal:

Goal *Investigate the split-and-merge strategy's capability of producing unseen terms in the computer domain when translating from English to Norwegian, focusing on compound terms.*

The results from the previous chapters have shown that the split-and-merge strategy is capable of producing novel Norwegian compounds in the computer science domain. Several well-received, novel translations are produced, like *beskyttelsesbyte*, *leketøysprogram*, and *prototype-basert programmeringsspråk*. However, generally, the translation system struggles with issues like lexical ambiguity, out-of-vocabulary words, and a simplistic merge-strategy. More work is needed to address these issues, especially on the topic of domain adaption.

As for contributions, this project has first and foremost produced a custom Norwegian compound splitter that can assist Norwegian taggers where these lack decompounding capabilities. The splitter is implemented based on the part-of-speech method presented by Sjöbergh and Kann (2004). Despite struggling with the concept of epentheses, the splitter performs very well for compounds with lexicalised *s*. The splitter has an accuracy of 0.7389 on a test set of mixed types of compounds (ambiguous compounds, compounds with epentheses, and compounds without epentheses). Nine test sets designed to identify what type of errors the compound splitter makes are also attached to this project's source code.

Secondly, this splitter is integrated into the translation pipeline of a statistical machine translation system. This translation system is able to produce compounds that were not seen during training, but native speakers' assessments of these translations are mixed.

Thirdly, Chapter 4 provides a detailed review of available data sources that can be used in similar projects. Chapter 8 also discusses some subpar choices made during experimentation that should be avoided in similar projects.

9.2. Conclusion

As pointed out by other researchers (see Section 3.2), the splitting quality does not appear to play a large role in the resulting translation quality. Subsection 8.2.7 supports that there is no inherent relation between translation quality and corpus size. This is in agreement with other articles discussed.

It has been demonstrated that the split-and-merge approach can produce novel compounds in the computer science domain by learning and reassembling constituents. The translation system also produces some already-established translations despite these not being found in the training data, which shows that the system has the potential to output native-like translations. However, the approach used here limits what lexemes can be used as constituents, and despite being trained on a corpus with in-domain vocabulary, lexical ambiguity and out-of-vocabulary words are still prevalent problems. On the other hand, the high degree of loanwords used in the Norwegian computer science domain allows some translations containing out-of-vocabulary words to be accepted by native speakers. Still, the issues discussed above require more attention and prevent the system from outputting reliable translations.

9.3. Future Work

This section provides suggestions as for how the current setup can be improved.

In terms of implementation, one of the biggest challenges is the splitter's runtime; analysing compound parts takes a substantial amount of time, which grows with the length of the compound. Looking up potential constituents in the lexicon is the bottleneck of the splitter. The more constituents need to be looked up, the longer it takes to analyse the compound. One effort made to curb the search time was to use memoisation (see Section 5.1.1), but the runtime is still extensive. Lowering the scan time can be attempted by experimenting with holding the lexicon and the corresponding parts-of-speech in other data structures, or by reducing the number of look-ups.

The splitter has been demonstrated to suppress epentheses to a large degree. This has not been proven to affect the translation system, but to achieve correctness of compound splitting, the approach of Johannessen (2001) can be implemented. Here, constituents are not allowed to contain inflected forms. As mentioned in Section 8.1, the necessary resources to implement this approach are readily available: the NST database (Section 4.1.2) is equipped with additional fields that indicate whether an entry is an inflected form. Suggested splits containing such forms may thus be pruned.

In addition to avoiding epentheses, the splitter does not support proper names or hyphens. Hyphens can be treated similarly to epentheses (which are treated as valid constituents of length 1 or 2). Care must then be taken to ensure hyphens are treated the same way when tagged by The Oslo-Bergen Tagger (OBT). It is not known whether OBT automatically treats these as compound words or if the tagger contains some hyphenated lexemes in its corpus. Proper names can be handled using the suggestion of Johannessen and Hauglin (1998), where the split with the longest last member is chosen, and then assuming that the remaining part of the compound is a proper name.

The tagger should also be further developed to handle three identical consonants, both in training and output. Thorsen Ranang (2010, p. 49) presents a solution to handling this that could be explored.

As pointed out in Subsection 8.2.7, the corpus size does not affect the translation quality as much as the content of the corpus. One source of in-domain data that was

not explored in this thesis is technical user manuals. These are likely to contain large amounts of relevant vocabulary, but no collections of such parallel texts were identified. It is therefore likely that acquiring such a corpus will require a substantial amount of manual effort.

Appendix C.1 lists the directory `semesteroppgave/technicalTerms`. This folder contains some of the author's efforts to collect and compile technical English vocabulary and Norwegian translation from various online resources. These can be used in similar projects; e.g., by expand an existing lexicon with technical terms. Bear in mind that these collections have been manually annotated by a non-linguist, and are thus not guaranteed to be error-free. Details about sources and annotation schemes can be found in the `README` file in the same location.

As for the translation system, no efforts have been made to tune the weights of the different models. Tuning is an important step in all articifical intelligence implementations and this issue should therefore receive more attention.

Subsection 8.2.5 points out how the postprocessing step was cut short and suffers from oversimplification. Other postprocessing approaches that could be investigated are Stymne et al. (2013) and Cap et al. (2014).

The example of the singular and plural form of 'programming language' in Subsection 8.2.8 showed that the translation system will benefit from morphological analysis. The Moses documentation[1] suggests to model translation on the level of lemmata to handle morphologically rich languages. It could also be explored how well these morphological models apply to compounds, and whether or not it is sufficient to apply these to the compound head.

Finally, attempts should be made to strengthen the statistical significance of the ratings. Recruiting students from the Department of Computer Science at NTNU (Norwegian University of Science and Technology) as evaluators would have been a valuable data source, but other departments, like the Department of Language and Literature, could also be contacted. It is also possible to reach out to the equivalent departments of other Norwegian universities, like UiO (University of Oslo) or UiB (University of Bergen). Any institution of interest should be contacted early to ensure that the administration has ample time to respond to the request.

[1] `http://www.statmt.org/moses/?n=Moses.FactoredModels`

Bibliography

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. Decompounding query keywords from compounding languages. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 253–256, Stroudsburg, PA, USA, Jun 2008. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1557690.1557763`.

Gisle Andersen. Leksikalsk database for norsk [Lexical database for Norwegian]. `https://www.nb.no/sbfil/dok/nst_leksdat_no.pdf`, 2005. Extract from "Gjennomgang og evaluering av språkressurser fra NSTs konkursbo" [Evaluation of language resources from NST's assets]. In Norwegian.

Mihael Arcan, Paul Buitelaar, and Christian Federmann. Using Domain-specific and Collaborative Resources for Term Translation. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 86–94, Jeju, Republic of Korea, Jul 2012. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=2392936.2392950`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, International Conference on Learning Representations*, San Diego, CA, USA, May 2015. URL `http://arxiv.org/abs/1409.0473`.

Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

Bo Bjerke-Lindstrøm. Teaching NLTK Norwegian. MSc Thesis, Dept. of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway, 2017.

Sigrid Sørumgård Botheim. Norge blir digitalt—på norsk [Norway becomes digital—in Norwegian]. *Språknytt*, Aug 2019. URL `https://www.sprakradet.no/Vi-og-vart/Publikasjoner/Spraaknytt/spraknytt-32019/intervjuet-norge-blir-digitalt--pa-norsk/`. In Norwegian.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993. URL `https://www.aclweb.org/anthology/J93-2003`.

Lars Bungum and Stephan Oepen. Automatic Translation of Norwegian Noun Compounds. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, pages 136–143, Barcelona, Spain, May 2009. European Association for Machine Translation.

Fabienne Cap, Alexander Fraser, Marion Weller, and Aoife Cahill. How to Produce Unseen Teddy Bears: Improved Morphological Processing of Compounds in SMT. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 579–587, Gothenburg, Sweden, Apr 2014. Association for Computational Linguistics.

Vincent Claveau. Automatic Translation of Biomedical Terms by Supervised Machine Learning. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, pages 684–691, Marrakech, Morocco, May 2008. European Language Resources Association.

Elizaveta Loginova Clouet and Béatrice Daille. Multilingual compound splitting combining language dependent and independent features. *Dialogue*, pages 455–463, May 2013.

Jan Terje Faarlund, Svein Lie, and Kjell Ivar Vannebo. *Norsk referansegrammatikk [Norwegian reference grammar]*. Universitetsforlaget, Oslo, Norge, 1997. In Norwegian. Available online: `https://www.nb.no/items/URN:NBN:no-nb_digibok_2009120100107?page=123`.

Monica Gavrila and Cristina Vertan. Training Data in Statistical Machine Translation — the More, the Better? In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 551–556, Hissar, Bulgaria, Sep 2011. Association for Computational Linguistics.

Emiliano Guevara, Sergio Scalise, Antonietta Bisetto, and Chiara Melloni. MORBO/-COMP: a multilingual database of compound words. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 2160–2163, Genoa, Italy, May 2006. European Language Resources Association.

Emiliano Raul Guevara. NoWaC: a large web-based corpus for Norwegian. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Sixth Web as Corpus Workshop*, pages 1–7, Los Angeles, CA, USA, Jun 2010. Association for Computational Linguistics.

Biman Gujral, Huda Khayrallah, and Philipp Koehn. Translation of Unknown Words in Low Resource Languages. In *Proceedings of the Conference of the Association for Machine Translation in the Americas*, pages 163–176, Austin, TX, USA, Oct–Nov 2016. Association for Computational Linguistics.

Maria Holmqvist, Sara Stymne, and Lars Ahrenberg. Getting to know Moses: Initial Experiments on German-English Factored Translation. In *Proceedings of the Second*

Workshop on Statistical Machine Translation, pages 181–184, Prague, Czech Republic, Jun 2007. Association for Computational Linguistics.

John Hutchins. From first conception to first demonstration: the nascent years of machine translation, 1947-1954. A chronology. *Machine Translation*, 12:195–252, 1997. URL `https://link.springer.com/content/pdf/10.1023%2FA%3A1007969630568.pdf`.

Iceland Ministry of Education, Science and Culture. Icelandic: at once ancient and modern, 2001. URL `https://web.archive.org/web/20070705181515/http://bella.mrn.stjr.is/utgafur/enska.pdf`. English translation: Daniel Teague. Published in collaboration with the Icelandic Language Institute and the National Committee on the European Year of Languages 2001. The Ministry for Foreign Affairs supported this publication.

Janne Bondi Johannessen. Sammensatte ord [compound words]. *Norsk lingvistisk tidsskrift*, pages 59–92, 2001. In Norwegian.

Janne Bondi Johannessen and Helge Hauglin. An Automatic Analysis of Norwegian Compounds. In *Papers from the 16th Scandinavian Conference of Linguistics*, pages 209–220, Turku, Finland, Nov 1998. Department of Finnish and General Linguistics of the University of Turku.

Janne Bondi Johannessen, Kristin Hagen, Lynum André, and Anders Nøklestad. OBT+stat. A combined rule-based and statistical tagger. In Gisle Andersen, editor, *Exploring Newspaper Language: Using the web to create and investigate a large corpus of modern Norwegian*, pages 51–65. John Benjamins Publishing Company, Amsterdam, Netherlands, 2012.

Bjørghild Kjelsvik. *Bokmålsordboka, Web-edition*. Cambridge University Press, 2016.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86, 2005.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, Cambridge, UK, 2010.

Philipp Koehn and Hieu Hoang. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 868–876, Prague, Czech Republic, Jun 2007. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. Empirical methods for compound splitting. In *Proceedings of the 10th Conference on European Chapter of the Association for Computational Linguistics*, volume 1, pages 187–193, Budapest, Hungary, Apr 2003. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, BC, Canada, Aug 2017. Association for Computational Linguistics.

Philipp Koehn, Amittai Axelrod, Alexandra Birch-Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *International Workshop on Spoken Language Translation 2005*, pages 68–75, Pittsburgh, PA, USA, Oct 2005. International Speech Communication Association Archive.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, and Richard Zens. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics, demonstration session*, pages 177–180, Prague, Czech Republic, Jun 2007. Association for Computational Linguistics.

Kulturdepartementet and Kunnskapsdepartementet. Norsk fagspråk i forskning og høyere utdanning [Norwegian academic language in reasearch and higher education]. *Regjeringen.no*, Nov 2018. URL `https://www.regjeringen.no/no/aktuelt/norsk-fagsprak-i-forskning-og-hoyere-utdanning/id2614784/`. In Norwegian.

Eivind Mikael Lindbråten. Automatisk orddeling – en regelbasert tilnærming til orddelingsproblemet [Automatic word breaking – a rule based approach to the word break problem]. MSc Thesis, Dept. of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway, 2015. In Norwegian.

Jan Tore Lønning, Stephan Oepen, Dorothee Beermann, Lars Hellan, John Carroll, Helge Dyvik, Dan Flickinger, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, Victoria Rosén, and Erik Velldal. LOGON. A Norwegian MT Effort. In *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden, 2004. Uppsala University, Department of Linguistics and Philology. URL `https://cl.lingfil.uu.se/RASMAT/extended_abstracts/LOGON.pdf`.

Klaus Macherey, Andrew M. Dai, David Talbot, Ashok C. Popat, and Franz Josef Och. Language-independent compound splitting with morphological operations. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404, Portland, OR, USA, Jun 2011. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013. URL `https://arxiv.org/abs/1301.3781`.

National Academy of Sciences. Languages and machines: computers in translation and linguistics. A report by the Automatic Language Processing Advisory Committee, Division of Behavioral Sciences, National Academy of Sciences, National Research Council. Washington, DC, USA, 1966.

Jan Niehues and Alexander H. Waibel. Using Wikipedia to translate domain-specific terms in SMT. In *Proceedings of the Eighth International Workshop on Spoken Language Translation*, pages 230–237, San Francisco, CA, USA, Dec 2011. Association for Computational Linguistics.

Sonja Nießen and Hermann Ney. Improving SMT Quality with Morpho-syntactic Analysis. In *Proceedings of the 18th Conference on Computational Linguistics*, pages 1081–1085, Saarbrücken, Germany, Jul–Aug 2000. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003.

Signe Oksefjell. A description of the English-Norwegian Parallel Corpus: Compilation and further developments. *International Journal of Corpus Linguistics*, 4(2):197–219, 1999.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA, July 2002. Association for Computational Linguistics.

Maja Popovic and Hermann Ney. Statistical Machine Translation with a Small Amount of Bilingual Training Data. In *5th International Language Resources and Evaluation Conference Speech and Language Technology for Minority Languages, Workshop on Minority Languages*, pages 25–29, Genoa, Italy, May 2006. International Language Resources and Evaluation Conference.

Martin Riedl and Chris Biemann. Unsupervised Compound Splitting with Distributional Semantics Rivals Supervised Methods. In *Proceedings of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 617–622, San Diego, CA, USA, Jun 2016. Association for Computational Linguistics.

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. Translating with Non-contiguous Phrases. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 755–762, Vancouver, BC, Canada, Oct 2005. Association for Computational Linguistics.

Jonas Sjöbergh and Viggo Kann. Finding the Correct Interpretation of Swedish Compounds a Statistical Approach. In *Proceedings of Language Resources Evaluation Conference 2004*, pages 899–902, Lisbon, Portugal, May 2004. European Language Resources Association.

Per Erik Solberg, Arne Skjærholt, Lilja Øvrelid, Kristin Hagen, and Janne Bondi Johannessen. The Norwegian Dependency Treebank. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph

Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 789–795, Reykjavik, Iceland, May 2014. European Language Resources Association.

Språkrådet and University of Bergen. Norsk ordbank — norwegian bokmål 2005, 2005. URL `https://www.nb.no/sprakbanken/show?serial=oai%3Anb.no%3Asbr-5&lang=en`. Accessed on 2019-10-03.

Andreas Stolcke. SRILM — An Extensible Language Modeling Toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904, Denver, CO, USA, Jul 2002. International Speech Communication Association Archive.

Sara Stymne, Nicola Cancedda, and Lars Ahrenberg. Generation of Compound Words in Statistical Machine Translation into Compounding Languages. *Computational Linguistics*, 39(4):1067–1108, 2013. URL `https://www.aclweb.org/anthology/J13-4009`.

Martin Thorsen Ranang. *Open-Domain Word-Level Interpretation of Norwegian: Towards a General Encyclopedic Question-Answering System for Norwegian.* PhD thesis, Dept. of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, Feb 2010.

Jörg Tiedemann and Lars Nygaard. The OPUS corpus - parallel & free. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 1183–1186, Lisbon, Portugal, May 2004. European Language Resources Association.

Stefano Vegnaduzzo. Morphological Productivity Rankings of Complex Adjectives. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 79–86, Boulder, CO, USA, Jun 2009. Association for Computational Linguistics.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-Based Word Alignment in Statistical Translation. In *The 16th International Conference on Computational Linguistics*, pages 836–841, Copenhagen, Denmark, Aug 1996. International Committee on Computational Linguistics.

Longyue Wang, Yi Lu, Derek F. Wong, Lidia S. Chao, Yiming Wang, and Francisco Oliveira. Combining Domain Adaptation Approaches for Medical Text Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 254–259, Baltimore, MD, USA, Jun 2014. Association for Computational Linguistics.

Warren Weaver. Translation. *Machine translation of languages*, 14:15–23, 1955.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg

Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, Oct 2016. URL `http://arxiv.org/abs/1609.08144`.

Eray Yıldız, Banu Diri, and A. Cüneyd Tantuğ. The Effect of Parallel Corpus Quality vs Size in English-to-Turkish SMT. In *Proceedings of the Fourth International Conference on Computer Science, Engineering and Applications*, pages 21–30, Chennai, India, Jul 2014. Academy & Industry Research Collaboration Center Computer Science Conference Proceedings.

Appendices

A. Complete List of Ambiguous Compounds

This appendix shows the complete list of the ambiguous Norwegian compounds that were used in evaluating the splitters in Section 6.1. Their descriptions and corresponding tables are given in the list below.

The column 'ID' identifies a test item so that it is possible to compare it across tables. 'Norwegian compound' shows the full compound. 'Correct split' shows the gold standard decompounding that the splitter is evaluated against. Note that *Johannessen* and *Representative* have their own IDs because these are treated differently than the other test sets.

- *Ambiguous*: 118 manually compiled ambiguous compounds. The ambiguity may be due to splitting into wrong constituents, mistaking epentheses for lexical compounding or vice versa. All compounds that appear in one of the other test sets are ambiguous, so they are also included in this set. This set is shown in Table A.1.
- *Epenthetic e*: 8 manually compiled ambiguous compounds, where the ambiguity should be interpreted as epenthetic *e*; e.g., *boks-e-kjøtt* (tinned meat), not **bokse-kjøtt* (boxing meat). This set is shown in Table A.2.
- *Epenthetic s*: 50 manually compiled ambiguous compounds, where the ambiguity should be interpreted as epenthetic *s*; e.g., *drap-s-alarmen* (attack alarm), not **draps-alarmen* (murder's alarm) or **drap-sal-armen* (murder saddle [the] arm). This set is shown in Table A.3.
- *Johannessen*: 37 ambiguous compounds, based on the examples given in Johannessen and Hauglin (1998). This set is shown in Table A.4.
- *Lexicalised e*: 17 manually compiled ambiguous compounds, where the ambiguity should be interpreted as part of one of the constituents; e.g., *bokse-talent* (boxing talent), not **boks-e-talent* (box talent). This set is shown in Table A.5.
- *Lexicalised s*: 16 manually compiled ambiguous compounds, where the ambiguity should be interpreted as part of one of the constituents; e.g., *bok-selger* (book seller), not **bok-s-elger* (book moose [plural]) or **boks-elger* (box moose [plural]). This set is shown in Table A.6.
- *No epenthesis*: 56 manually compiled ambiguous compounds, where the ambiguity should be interpreted as two constituents with no epenthesis; e.g., *banekapasiteten* ([the] capacity [of a] [sports] field), not **ban-e-kapasiteten* (clear [imperative] capacity). This set is shown in Table A.7.
- *No split*: 16 manually compiled lexemes that can be interpreted as compounds

but are single lexemes; e.g., *morsommere* (funnier), not **mor-sommere* (mother summers). This set is shown in Table A.8.

- *Representative*: 203 compounds manually compiled by extracting the first 500 compounds returned from Oslo-korpuset's web interface[1]. The quality of being a compound is determined by the Text Laboratory's multitagger[2], and obvious errors, like *euroen* (the Euro), were left out. Coordinated compounds, e.g., *hotell- og feriekonge* (the king of hotels and holidays), were shortened to the last constituent. Furthermore, compounds containing hyphens, abbreviations and proper nouns were omitted because these are not handled by the splitters. Finally, all entries were transformed to lowercase and duplicates were removed. This left a test set where 53 of the compounds had already been observed in the corpus and 150 were unknown. This set is shown in Table A.9.

Table A.1.: Full list of ambiguous compounds.

ID	Norwegian compound	Correct split
1	villahus	villa-hus
2	kulturforskeren	kultur-forskeren
3	melkeforretning	melk-e-forretning
4	slottsvinduene	slott-s-vinduene
5	vinduskitt	vindu-s-kitt
6	takk-for-maten-taler	takk-for-maten-taler
7	lysmaskinen	lys-maskinen
8	krigsmaske	krig-s-maske
9	aluminiumsnakke	aluminium-s-nakke
10	lesesalsturer	lesesal-s-turer
11	storhavstang	storhav-s-tang
12	buskspilling	busk-spilling
13	bildekor	bil-dekor
14	flyskam	fly-skam
15	brannslukker	brann-slukker
16	fisketanker	fisk-e-tanker
17	plantesting	plan-testing
18	plantestasjon	plante-stasjon
19	skolemur	skole-mur
20	haremdanserinne	harem-danserinne
21	stolkarmen	stol-karmen
22	norgescupseire	norge-s-cup-seire
23	stemmeklar	stemme-klar

Continued on next page

[1] https://tekstlab.uio.no/norsk/korpus/bokmaal/netscape/treord/oktntb.shtml
[2] http://www.tekstlab.uio.no/norsk/bokmaal/english.html#tagger

Table A.1 – *Continued from previous page*

ID	Norwegian compound	Correct split
24	drømmeide	drømme-ide
25	robotarmen	robot-armen
26	rockenatten	rock-e-natten
27	innenifra	innen-ifra
28	arbeidstreningen	arbeid-s-treningen
29	bokserier	bok-serier
30	boksekjøtt	boks-e-kjøtt
31	boksetalent	bokse-talent
32	ullsåler	ull-såler
33	strikkenåler	strikke-nåler
34	begynnerjobb	begynner-jobb
35	mannerøst	mann-e-røst
36	insektarter	insekt-arter
37	rosenblad	rose-n-blad
38	sønnabris	sønn-a-bris
39	bilalarm	bil-alarm
40	drapsalarmen	drap-s-alarmen
41	vinnerbidrag	vinner-bidrag
42	tidsur	tid-s-ur
43	rugbyspiller	rugby-spiller
44	bukselengder	bukse-lengder
45	dykkertropp	dykker-tropp
46	bokselger	bok-selger
47	overgangspapirene	overgang-s-papirene
48	samfunnsfunksjon	samfunn-s-funksjon
49	samferdselsløsninger	samferdsel-s-løsninger
50	bjørnekreftene	bjørn-e-kreftene
51	frelsestilbud	frelse-s-tilbud
52	krigsforbryterfengsel	krig-s-forbryter-fengsel
53	tiltalelisten	tiltale-listen
54	tilhørertribunen	tilhører-tribunen
55	koppevaksine	koppe-vaksine
56	treningsklær	trening-s-klær
57	skoletilhørigheten	skole-tilhørigheten
58	delingstimer	deling-s-timer
59	barneutstyrsforretningen	barn-e-utstyr-s-forretningen
60	vintersportutstyr	vinter-sport-utstyr
61	bispestemme	bispe-stemme
62	europacupfotball	europa-cup-fotball

Continued on next page

Table A.1 – *Continued from previous page*

ID	Norwegian compound	Correct split
63	videomillionæren	video-millionæren
64	kunnskapsbasen	kunnskap-s-basen
65	utdanningsansvar	utdanning-s-ansvar
66	doktorgradsstipender	doktorgrad-s-stipender
67	bildelindustri	bil-del-industri
68	skipsdesign	skip-s-design
69	risikoavlastning	risiko-avlastning
70	suksesskonsert	suksess-konsert
71	slankekontraktene	slanke-kontraktene
72	vektpiningen	vekt-piningen
73	enkeltkonkurranser	enkelt-konkurranser
74	landslagskamerater	landslag-s-kamerater
75	slankekontrakter	slanke-kontrakter
76	nasjonspåmelding	nasjon-s-påmelding
77	toppidrettssenter	toppidrett-s-senter
78	toppidrettsbyen	toppidrett-s-byen
79	toppidrettsenter	toppidrett-senter
80	toppidrettsjef	toppidrett-sjef
81	breddeidrettssjef	bredde-idrett-s-sjef
82	idrettsvirksomhet	idrett-s-virksomhet
83	toppidrettsbyen	toppidrett-s-byen
84	toppidrettsbyer	toppidrett-s-byer
85	idrettsrevolusjon	idrett-s-revolusjon
86	jubileumsakevitt	jubileum-s-akevitt
87	jubileumsakevitten	jubileum-s-akevitten
88	slankekontrakt	slanke-kontrakt
89	slankepress	slanke-press
90	slankekontrakter	slanke-kontrakter
91	onsdagstrimmen	onsdag-s-trimmen
92	planleggingsministerens	planlegging-s-ministerens
93	banekapasiteten	bane-kapasiteten
94	banestandarden	bane-standarden
95	museumsnybygg	museum-s-nybygg
96	publikumsbygget	publikum-s-bygget
97	viseutenriksministeren	vise-utenriksministeren
98	fiskeforedlingsbedriften	fisk-e-foredling-s-bedriften
99	kroppslengden	kropp-s-lengden
100	tilleggsjobb	tillegg-s-jobb
101	utendørsbanen	utendørs-banen

Continued on next page

Table A.1 – *Continued from previous page*

ID	Norwegian compound	Correct split
102	landslagsutøvere	landslag-s-utøvere
103	hjelpekorridorer	hjelpe-korridorer
104	stålselskapet	stål-selskapet
105	anbudspapirene	anbud-s-papirene
106	etterretningsmetodikk	etterretning-s-metodikk
107	dyrkajorda	dyrk-a-jorda
108	dyrkajord	dyrk-a-jord
109	mesterligaspill	mesterliga-spill
110	fusjonsmotstandere	fusjon-s-motstandere
111	regnskapsframleggelse	regnskap-s-framleggelse
112	plankegater	planke-gater
113	narkotikaproblematikken	narkotika-problematikken
114	elevtimetallet	elev-time-tallet
115	samrådsbesøk	samråd-s-besøk
116	sammenslåingsprogrammet	sammenslåing-s-programmet
117	ukestarten	uke-starten
118	stortapene	stor-tapene

Table A.2.: Full list of the *Epenthetic e* test set.

ID	Norwegian compound	Correct split
3	melkeforretning	melk-e-forretning
16	fisketanker	fisk-e-tanker
26	rockenatten	rock-e-natten
30	boksekjøtt	boks-e-kjøtt
35	mannerøst	mann-e-røst
50	bjørnekreftene	bjørn-e-kreftene
59	barneutstyrsforretningen	barn-e-utstyr-s-forretningen
98	fiskeforedlingsbedriften	fisk-e-foredling-s-bedriften

Table A.3.: Full list of the *Epenthetic s* test set.

ID	Norwegian compound	Correct split
4	slottsvinduene	slott-s-vinduene
5	vinduskitt	vindu-s-kitt
8	krigsmaske	krig-s-maske
9	aluminiumsnakke	aluminium-s-nakke
10	lesesalsturer	lesesal-s-turer
11	storhavstang	storhav-s-tang
22	norgescupseire	norge-s-cup-seire
28	arbeidstreningen	arbeid-s-treningen
40	drapsalarmen	drap-s-alarmen
42	tidsur	tid-s-ur
47	overgangspapirene	overgang-s-papirene
48	samfunnsfunksjon	samfunn-s-funksjon
49	samferdselsløsninger	samferdsel-s-løsninger
51	frelsestilbud	frelse-s-tilbud
52	krigsforbryterfengsel	krig-s-forbryter-fengsel
56	treningsklær	trening-s-klær
58	delingstimer	deling-s-timer
59	barneutstyrsforretningen	barn-e-utstyr-s-forretningen
64	kunnskapsbasen	kunnskap-s-basen
65	utdanningsansvar	utdanning-s-ansvar
66	doktorgradsstipender	doktorgrad-s-stipender
68	skipsdesign	skip-s-design
74	landslagskamerater	landslag-s-kamerater
76	nasjonspåmelding	nasjon-s-påmelding
77	toppidrettssenter	toppidrett-s-senter
78	toppidrettsbyen	toppidrett-s-byen
81	breddeidrettssjef	bredde-idrett-s-sjef
82	idrettsvirksomhet	idrett-s-virksomhet
83	toppidrettsbyen	toppidrett-s-byen
84	toppidrettsbyer	toppidrett-s-byer
85	idrettsrevolusjon	idrett-s-revolusjon
86	jubileumsakevitt	jubileum-s-akevitt
87	jubileumsakevitten	jubileum-s-akevitten
91	onsdagstrimmen	onsdag-s-trimmen
92	planleggingsministerens	planlegging-s-ministerens
95	museumsnybygg	museum-s-nybygg
96	publikumsbygget	publikum-s-bygget
98	fiskeforedlingsbedriften	fisk-e-foredling-s-bedriften

Continued on next page

Table A.3 – *Continued from previous page*

ID	Norwegian compound	Correct split
99	kroppslengden	kropp-s-lengden
100	tilleggsjobb	tillegg-s-jobb
102	landslagsutøvere	landslag-s-utøver
105	anbudspapirene	anbud-s-papirene
106	etterretningsmetodikk	etterretning-s-metodikk
110	fusjonsmotstandere	fusjon-s-motstandere
111	regnskapsframleggelse	regnskap-s-framleggelse
115	samrådsbesøk	samråd-s-besøk
116	sammenslåingsprogrammet	sammenslåing-s-programmet
R121	forsvarsskriv	forsvar-s-skriv
R154	veiarbeidsmerkingen	vei-arbeid-s-merkingen

Table A.4.: Full list of the *Johannessen* test set.

ID	Norwegian compound	Correct split
J1	bilsyk	bil-syk
J2	telefonsvarer	telefon-svarer
J3	klesskap	klær-skap
J4	billedspråk	bilde-språk
J5	morsbinding	mors-binding
J6	aluminiumsfabrikk	aluminiums-fabrikk
J7	barnetrygd	barne-trygd
J8	hesteekvipasje	heste-ekvipasje
J9	kulturforskeren	kultur-forskeren
J10	melkeforretning	melk-e-forretning
J11	slottsvinduene	slott-s-vinduene
J12	verdenscupseire	verdenscup-seire
J13	vinduskitt	vindu-s-kitt
J14	heleide	hel-eide
J15	tilsynsorgan	tilsyn-s-organ
J16	ikkespredningsavtale	ikkespredning-s-avtale
J17	takk-for-maten-taler	takk-for-maten-taler
J18	lysmaskinen	lys-maskinen
J19	løvemanke	løve-manke
J20	ølskum	øl-skum
J21	krigsmaske	krig-s-maske
J22	aluminiumsnakke	aluminium-s-nakke
J23	oppslag	opp-slag
J24	lesesalsturer	lesesal-s-turer
J25	storhavstang	storhav-s-tang
J26	barneskje	barn-e-skje
J27	blomsterholder	blomster-holder
J28	hundyr	hun-dyr
J29	spisestueur	spisestue-ur
J30	fagplanarbeid	fagplan-arbeid
J31	hesteekvipasje	hest-e-ekvipasje
J32	trehestekvipasje	tre-hest-ekvipasje
J33	kongehushesteekvipasje	konge-hus-hest-e-ekvipasje
J34	buskspilling	busk-spilling
J35	couturevisningen	couture-visningen
J36	alkoroboten	alko-roboten
J37	lavastøvet	lava-støvet

Table A.5.: Full list of the *Lexicalised e* test set.

ID	Norwegian compound	Correct split
18	plantestasjon	plante-stasjon
31	boksetalent	bokse-talent
33	strikkenåler	strikke-nåler
44	bukselengder	bukse-lengder
53	tiltalelisten	tiltale-listen
55	koppevaksine	koppe-vaksine
57	skoletilhørigheten	skole-tilhørigheten
71	slankekontrakter	slanke-kontrakter
88	slankekontrakt	slanke-kontrakt
89	slankepress	slanke-press
90	slankekontrakter	slanke-kontrakter
91	banekapasiteten	bane-kapasiteten
94	banestandarden	bane-standarden
97	viseutenriksministeren	vise-utenriksministeren
103	hjelpekorridorer	hjelpe-korridorer
112	plankegater	planke-gater
117	ukestarten	uke-starten

Table A.6.: Full list of the *Lexicalised s* test set .

ID	Norwegian compound	Correct split
12	buskspilling	busk-spilling
14	flyskam	fly-skam
15	brannslukker	brann-slukker
32	ullsåler	ull-såler
43	rugbyspiller	rugby-spiller
46	bokselger	bok-selger
61	bispestemme	bispe-stemme
101	utendørsbanen	utendørs-banen
117	ukestarten	uke-starten
R98	tanksenter	tank-senter

Table A.7.: Full list of the *No epenthesis* test set.

ID	Norwegian compound	Correct split
1	villahus	villa-hus
2	kulturforskeren	kultur-forskeren
12	buskspilling	busk-spilling
13	bildekor	bil-dekor
14	flyskam	fly-skam
15	brannslukker	brann-slukker
17	plantesting	plan-testing
18	plantestasjon	plante-stasjon
19	skolemur	skole-mur
20	haremdanserinne	harem-danserinne
21	stolkarmen	stol-karmen
23	stemmeklar	stemme-klar
24	drømmeide	drømme-ide
25	robotarmen	robot-armen
27	innenifra	innen-ifra
29	bokserier	bok-serier
32	ullsåler	ull-såler
33	strikkenåler	strikke-nåler
34	begynnerjobb	begynner-jobb
36	insektarter	insekt-arter
39	bilalarm	bil-alarm
41	vinnerbidrag	vinner-bidrag
43	rugbyspiller	rugby-spiller
44	bukselengder	bukse-lengder
45	dykkertropp	dykker-tropp
46	bokselger	bok-selger
53	tiltalelisten	tiltale-listen
54	tilhørertribunen	tilhører-tribunen
57	skoletilhørigheten	skole-tilhørigheten
60	vintersportutstyr	vinter-sport-utstyr
61	bispestemme	bispe-stemme
62	europacupfotball	europa-cup-fotball
63	videomillionæren	video-millionæren
67	bildelindustri	bil-del-industri
69	risikoavlastning	risiko-avlastning
70	suksesskonsert	suksess-konsert
71	slankekontraktene	slanke-kontraktene
72	vektpiningen	vekt-piningen

Continued on next page

Table A.7 – *Continued from previous page*

ID	Norwegian compound	Correct split
73	enkeltkonkurranser	enkelt-konkurranser
75	slankekontrakter	slanke-kontrakter
79	toppidrettsenter	toppidrett-senter
80	toppidrettsjef	toppidrett-sjef
75	slankekontrakt	slanke-kontrakt
89	slankepress	slanke-press
90	slankekontrakter	slanke-kontrakter
91	banekapasiteten	bane-kapasiteten
94	banestandarden	bane-standarden
101	utendørsbanen	utendørs-banen
103	hjelpekorridorer	hjelpe-korridorer
104	stålselskapet	stål-selskapet
112	plankegater	planke-gater
113	narkotikaproblematikken	narkotika-problematikken
114	elevtimetallet	elev-time-tallet
117	ukestarten	uke-starten
118	stortapene	stor-tapene
R94	tingvedtak	ting-vedtak

Table A.8.: Full list of the *No split* test set.

ID	Norwegian compound	Correct split
N1	frister	frister
N2	fortjenesten	fortjenesten
N3	markedet	markedet
N4	bergenser	bergenser
N5	spaserende	spaserende
N6	sangerinne	sangerinne
N7	hermetikk	hermetikk
N8	flerring	flerring
N9	flammete	flammete
N10	ligament	ligament
N11	gravide	gravide
N12	kultur	kultur
N13	registrering	registrering
N14	strukturering	strukturering
N15	formål	formål
N16	morsommere	morsommere

Table A.9.: Full list of the *Representative* test set.

ID	Norwegian compound	Correct split
R1	overgangspapirene	overgang-s-papirene
R2	serieforeningen	serieforeningen
R3	samfunnsfunksjon	samfunn-s-funksjon
R4	samferdselsløsninger	samferdsel-s-løsninger
R5	dekkvalg	dekk-valg
R6	medtrafikkanter	medtrafikkanter
R7	finaleplass	finaleplass
R8	rankingpoeng	ranking-poeng
R9	rankingstatus	ranking-status
R10	pistolskyting	pistolskyting
R11	målskiven	mål-skiven
R12	finpistol	fin-pistol
R13	duellskyting	duell-skyting
R14	grovpistol	grov-pistol
R15	fripistol	fripistol
R16	silhuettpistol	silhuett-pistol
R17	feltpistol	felt-pistol
R18	finfelt	fin-felt
R19	enkeltapparater	enkelt-apparater
R20	favorittøvelsen	favoritt-øvelsen
R21	basketjenter	basket-jenter
R22	trønderjentene	trønder-jentene
R23	førsteomgangen	førsteomgangen
R24	vektløfterforbundet	vektløfterforbundet
R25	bjørnekreftene	bjørn-e-kreftene
R26	frelsestilbud	frelse-s-tilbud
R27	krigforbrytere	krig-forbrytere
R28	krigsforbryterfengsel	krigsforbryter-fengsel
R29	fengselkost	fengsel-kost
R30	klientantallet	klient-antallet
R31	krigsforbryterdomstolen	krigsforbryterdomstolen
R32	sluttstadiet	slutt-stadiet
R33	embedsmennene	embedsmennene
R34	tiltalelisten	tiltale-listen
R35	serberlederne	serberlederne
R36	mediaoppstyret	media-oppstyret
R37	tilhørertribunen	tilhører-tribunen
R38	stålglasset	stål-glasset

Continued on next page

Table A.9 – *Continued from previous page*

ID	Norwegian compound	Correct split
R39	barnefond	barnefond
R40	serviceorganisasjon	serviceorganisasjon
R41	vaksinedag	vaksine-dag
R42	vaksinedager	vaksine-dager
R43	vaksinedagene	vaksine-dagene
R44	koppevaksine	koppe-vaksine
R45	supertravere	super-travere
R46	fiberskiva	fiber-skiva
R47	basketmisjonærer	basket-misjonærer
R48	basketmisjonæren	basket-misjonæren
R49	baseballuene	baseball-luene
R50	treningsklær	trening-s-klær
R51	misjonærvirksomhet	misjonær-virksomhet
R52	skoletilhørigheten	skole-tilhørigheten
R53	avisselgeren	avis-selgeren
R54	rekkekamerat	rekkekamerat
R55	måltørke	måltørke
R56	spiontokt	spion-tokt
R57	vidvinkelsyn	vidvinkel-syn
R58	tribunebygg	tribune-bygg
R59	gigantprosjektene	gigant-prosjektene
R60	millionstøtte	millionstøtte
R61	superminister	super-minister
R62	delingstimer	deling-s-timer
R63	barneutstyrsforretningen	barn-e-utstyr-s-forretningen
R64	forsvarspill	forsvar-spill
R65	vintersportutstyr	vintersport-utstyr
R66	bispestemme	bispe-stemme
R67	europacupfotball	europa-cup-fotball
R68	videomillionæren	video-millionæren
R69	kunnskapsbasen	kunnskap-s-basen
R70	utdanningsansvar	utdanning-s-ansvar
R71	doktorgradsstipender	doktorgrad-s-stipender
R72	materialforskning	materialforskning
R73	bildelindustri	bil-del-industri
R74	skipsdesign	skip-s-design
R75	risikoavlastning	risiko-avlastning
R76	suksesskonsert	suksess-konsert
R77	sportsdirektør	sportsdirektør

Continued on next page

Table A.9 – *Continued from previous page*

ID	Norwegian compound	Correct split
R78	slankekontraktene	slanke-kontraktene
R79	vektpiningen	vekt-piningen
R80	enkeltkonkurranser	enkelt-konkurranser
R81	landslagskamerater	landslag-s-kamerater
R82	slankekontrakter	slanke-kontrakter
R83	nasjonspåmelding	nasjon-s-påmelding
R84	toppidrettssenter	toppidrett-s-senter
R85	toppidrettsby	toppidrett-s-by
R86	toppidrettsenter	toppidrett-senter
R87	toppidrettssenteret	toppidrettssenteret
R88	toppidrettsjef	toppidrett-sjef
R89	breddeidrettssjef	breddeidrett-s-sjef
R90	idrettsvirksomhet	idrett-s-virksomhet
R91	toppidrettsbyen	toppidrett-s-byen
R92	toppidrettsbyer	toppidrett-s-byer
R93	idrettsrevolusjon	idrett-s-revolusjon
R94	tingvedtak	ting-vedtak
R95	idrettskrets	idrettskrets
R96	jubileumsakevitt	jubileum-s-akevitt
R97	jubileumsakevitten	jubileum-s-akevitten
R98	tanksenter	tank-senter
R99	slankekontrakt	slanke-kontrakt
R100	slankepress	slanke-press
R101	slankekontrakter	slanke-kontrakter
R102	sportssjef	sportssjef
R103	sportsjef	sport-sjef
R104	onsdagstrimmen	onsdag-s-trimmen
R105	flyktningestrømmen	flyktningestrømmen
R106	militærstyrke	militærstyrke
R107	militærstyrken	militær-styrken
R108	minnecupen	minne-cupen
R109	miljørapport	miljørapport
R110	miljøtilstanden	miljøtilstanden
R111	reaktorhavari	reaktor-havari
R112	miljørapporten	miljø-rapporten
R113	supertroppen	super-troppen
R114	planleggingsminister	planleggingsminister
R115	millionstøtten	millionstøtten
R116	superministeren	super-ministeren

Continued on next page

Table A.9 – *Continued from previous page*

ID	Norwegian compound	Correct split
R117	planleggingsministerens	planlegging-s-ministerens
R118	planavdelingen	planavdelingen
R119	planleggingsministeren	planleggingsministeren
R120	pengeoverføringer	pengeoverføringer
R121	forsvarsskriv	forsvar-s-skriv
R122	fotballkrets	fotballkrets
R123	banekapasiteten	bane-kapasiteten
R124	fotballfolket	fotballfolket
R125	fotballkretsen	fotballkretsen
R126	banekapasiteten	banekapasiteten
R127	banestandarden	bane-standarden
R128	museumsnybygg	museum-s-nybygg
R129	publikumsbygget	publikum-s-bygget
R130	planbehandling	plan-behandling
R131	fotballkretsen	fotballkretsen
R132	svingtribunen	sving-tribunen
R133	planarbeidet	planarbeidet
R134	viseutenriksministeren	viseutenriksministeren
R135	fiskeforedlingsbedriften	fiskeforedling-s-bedriften
R136	hygienekravene	hygiene-kravene
R137	membrananlegg	membran-anlegg
R138	fiskeindustribedriftene	fiskeindustri-bedriftene
R139	hoppregler	hopp-regler
R140	kroppslengden	kropp-s-lengden
R141	skilengden	ski-lengden
R142	tilleggsjobb	tillegg-s-jobb
R143	pistolskyting	pistolskyting
R144	utendørsbanen	utendørs-banen
R145	medlemskontigent	medlemskontigent
R146	våpenkjøp	våpenkjøp
R147	våpenhold	våpen-hold
R148	landslagsutøvere	landslag-s-utøvere
R149	fiskeforedlingsbedriften	fiskeforedling-s-bedriften
R150	markedsdirektør	markedsdirektør
R151	plasthyllen	plast-hyllen
R152	hjelpekorridorer	hjelpe-korridorer
R153	flyktningeleir	flyktningeleir
R154	veiarbeidsmerkingen	veiarbeid-s-merkingen
R155	gravmaskinen	grav-maskinen

Continued on next page

Table A.9 – *Continued from previous page*

ID	Norwegian compound	Correct split
R156	overflatekontrakt	overflate-kontrakt
R157	brannbeskyttelse	brann-beskyttelse
R158	stålselskapet	stål-selskapet
R159	møbellager	møbel-lager
R160	anbudspapirene	anbud-s-papirene
R161	etterretningsmetodikk	etterretning-s-metodikk
R162	lakseprodusenter	laks-e-produsenter
R163	egenkapitaldivisjonen	egenkapitaldivisjonen
R164	granskningslisten	granskning-s-listen
R165	dumpingpåstanden	dumping-påstanden
R166	maksimumkvoter	maksimum-kvoter
R167	dyrkajorda	dyrk-a-jorda
R168	dyrkajord	dyrk-a-jord
R169	dyrefôr	dyrefôr
R170	mesterligaspill	mesterliga-spill
R171	fiskeridivisjonen	fiskeri-divisjonen
R172	hovedaksjonær	hovedaksjonær
R173	fusjonsmotstandere	fusjon-s-motstandere
R174	regnskapsframleggelse	regnskap-s-framleggelse
R175	kommunetall	kommune-tall
R176	besøksrunden	besøk-s-runden
R177	vetoduell	veto-duell
R178	plankegater	planke-gater
R179	elevforsamling	elev-forsamling
R180	narkotikaproblematikken	narkotika-problematikken
R181	delingstimeressurs	deling-s-time-ressurs
R182	elevtimetallet	elev-timetallet
R183	delingstimeressursen	deling-s-time-ressursen
R184	skolestruktur	skolestruktur
R185	delingsressurs	deling-s-ressurs
R186	delingstimetall	deling-s-timetall
R187	førsteelleveren	førsteelleveren
R188	netthandel	netthandel
R189	flyktningestrømmen	flyktningestrømmen
R190	nødhjelparbeidere	nødhjelp-arbeidere
R191	protestunderskrifter	protest-underskrifter
R192	samrådsbesøk	samråd-s-besøk
R193	skolestrukturen	skolestrukturen
R194	skolestruktursaken	skolestruktur-saken

Continued on next page

Table A.9 – *Continued from previous page*

ID	Norwegian compound	Correct split
R195	sammenslåingsprogrammet	sammenslåing-s-programmet
R196	skilengde	ski-lengde
R197	plasthopprennet	plast-hopprennet
R198	finaleomgangen	finaleomgangen
R199	ukestarten	uke-starten
R200	sportssjef	sportssjef
R201	hoppteknikk	hopp-teknikk
R202	landslagsledelsen	landslagsledelsen
R203	regelforslag	regel-forslag

B. Full Test Results

This appendix shows the full results of the Norwegian suggestions made by the translation system, as well as the native speakers' ratings of these translations and the control translations.

Table B.1 shows the Norwegian translations made by the translation system. The column 'ID' represents the translation's ID and is used to identify that translation throughout the discussions made in this thesis. The column 'Survey ID' indicates which survey the particular translation belonged to. The column 'English Term' represents the English term that was translated by the translation system. The column 'Norwegian Translation' shows the output translation in addition to applied post-processing.

Table B.1.: English input terms and Norwegian translations made by the translation system.

ID	Survey ID	English Term	Norwegian Translation
1	11	conditional operator	eroperatør
2	11	directive	direktivet
3	8	event	tilfelle
4	11	program comprehension	programcomprehension
5	11	self-hosting	self-datalagring
6	1	self-relocation	self-prisfastsettelse
7	10	side effect	sidevirkning
8	11	toy program	leketøysprogram
9	11	transient	flyktige
10	10	anti-patterns	anti-mønstre
11	10	program derivation	programderivation
12	10	broadcast	kringkasting
13	10	reduce	redusere
14	10	digit-by-digit algorithms	nummer-av-nummeralgoritmer
15	11	reduction	reduksjon
16	10	search algorithms	søke-algoritmer
17	10	hard coding	hardkoding
18	7	interface bloat	grensesnittbloat
19	9	cache coherence	direktebuffersamsvar
20	9	process	prosessen

Continued on next page

Table B.1 – *Continued from previous page*

ID	Survey ID	English Term	Norwegian Translation
21	3	thread	tråd
22	3	yield	tjæreinnholdet
23	9	scheduling	tidsplanlegging
24	9	threads	tråder
25	9	business object	foretaksgjenstand
26	9	collective operation	fellesdrift
27	9	constant	konstant
28	8	operator	operatør
29	8	resource management	ressursforvaltning
30	8	block	blokk
31	8	symbolic language	symbolsk språk
32	7	symbolic programming	symbolsk programmering
33	8	in-circuit emulation	i-overbelastningemulering
34	11	in-target probe	i-målprobe
35	8	program animation	programanimation
36	8	program slicing	programsnitting
37	3	minimal working example	enkleste arbeid .
38	7	code reuse	kode ombruk
39	7	dynamic linker	dynamisk linker
40	10	dynamic loading	dynamisk lasting
41	7	object file	gjenstand data
42	7	audio libraries	lydbiblioteker
43	7	graphics libraries	grafikk biblioteker
44	8	mathematical libraries	matematikk-biblioteker
45	7	software development kits	programvareutvikling-sett
46	6	numerical libraries	Numeriske biblioteker
47	6	program optimization	programoptimization
48	6	live coding	levende koding
49	6	chief programmer team	sjefsprogrammer lag
50	6	data organization for low power	organisasjonen for laveffektopplysninger
51	6	data-oriented design	data-orientert konstruksjon
52	6	input enhancement	inndata-utvidelser
53	6	operation reduction for low power	drift reduksjon for laveffekt
54	6	optimum programming	best programmering
55	5	competitive programming	konkurranseprogrammering
56	5	guard byte	beskyttelsesbyte
57	5	software design patterns	programvare relieffkonstruksjon
58	5	exception handling syntax	unntak håndteringssyntaks

Continued on next page

Table B.1 – *Continued from previous page*

ID	Survey ID	English Term	Norwegian Translation
59	5	processing	behandling
60	5	programming languages created by women	programmeringsspråk laget av kvinner
61	5	academic programming languages	akademiske programmeringsspråk
62	5	agent-oriented programming languages	ansatt-orientert programmeringsspråk
63	5	data-structured programming languages	data-strukturert programmeringsspråk
64	4	experimental programming languages	forsøksprogrammeringsspråk
65	4	function-level languages	funksjon-nivå språk
66	4	functional languages	funksjonelle språk
67	4	object-based programming languages	gjenstand-basert programmeringsspråk
68	4	prototype-based programming languages	prototype-basert programmeringsspråk
69	4	secure programming languages	sikre programmeringsspråk
70	4	data-directed programming	data-programmot
71	4	flow-based programming	gjennomstrømning-basert programmering
72	4	imperative programming	ufravikelig programmering
73	3	intentional programming	forsettlig programmering
74	3	multi-stage programming	flertrinns programmering
75	3	organic computing	organiske datamaskiner
76	3	role-oriented programming	rolle-orientert programmering
77	3	scientific programming language	vitenskapelige programspråk
78	7	strict programming language	strenge programspråk
79	3	black box	svart alternativ
80	9	offensive programming	krenke programmering
81	9	software architectural model	programvare arkitektmodell
82	2	control structure diagram	kontroll struktur diagram
83	2	dead code	encellede kode
84	2	duplicate code	overlapper kode
85	2	data transfer object	dataoverføring gjenstand
86	2	global offset table	globale offsetbord
87	2	input mask	inndata-maske
88	2	remote evaluation	fjernvurdering
89	2	non-malleable code	ikke-adusert kode
90	2	focus stealing	samle stealing

Continued on next page

Table B.1 – *Continued from previous page*

ID	Survey ID	English Term	Norwegian Translation
91	1	functional data structures	funksjonelle opplysninger konstruksjoner
92	1	cohesion	utjevning
93	1	general-purpose programming language	-programspråk generelle formål
94	1	god object	god gjenstand
95	1	differentiated service	differensierte tjenester
96	1	structured concurrency	strukturert concurrency
97	1	programming game	programmering vilt
98	1	digital signal processing	digitalt signal behandling
99	11	sequence point	sekvens bokstav

Table B.2 shows the Norwegian translations supplied by human Wikipedia contributors. The columns have the same functions as in Table B.1.

Table B.2.: English control translations and human-supplied Norwegian translations.

ID	Survey ID	English Term	Norwegian Translation
100	1	divide-and-conquer algorithm	splitt og hersk-algoritme
101	2	syntax highlighting	syntaksfremheving
102	3	source-code editor	kildekode-editor
103	4	procedural programming	prosedyrisk programmering
104	5	cooperative multitasking	kooperativ fleroppgavekjøring
105	6	concurrent computing	parallellprogrammering
106	7	scheduling	skedulering
107	8	software design pattern	designmønster
108	9	runtime library	runtimebibliotek
109	10	in-place algorithm	på stedet-algoritme
110	11	coupling	programvareavhengigheter

Table B.3 shows the Norwegian translations made by the translation system and their tagged output. Columns 'ID', 'Survey ID' have the same function as in Table B.1. The column 'Norwegian tagged output' shows the raw output from Moses; i.e., surface forms and part-of-speech tags.

Table B.3.: Translations and tagged output made by the translation system.

ID	Survey ID	Norwegian Translation	Norwegian tagged output
1	11	eroperatør	er|subst-modif operatør|subst
2	11	direktivet	direktivet|subst
3	8	tilfelle	tilfelle|prep+subst
4	11	programcomprehension	program|subst-modif comprehension|UNK
5	11	self-datalagring	self|UNK -|strek datalagring|subst
6	1	self-prisfastsettelse	self|UNK -|strek prisfastsettelse|subst
7	10	sidevirkning	side|subst-modif virkning|subst
8	11	leketøysproram	leketøys|subst-modif program|subst-modif
9	11	flyktige	flyktige|adj
10	10	anti-mønstre	anti|UNK -|strek mønstre|subst
11	10	programderivation	program|subst-modif derivation|UNK
12	10	kringkasting	kringkasting|subst
13	10	redusere	redusere|verb
14	10	nummer- av- nummeralgoritmer	nummer|subst-modif -|strek av|prep -|strek nummer|subst-modif algoritmer|subst
15	11	reduksjon	reduksjon|subst
16	10	søke-algoritmer	søke-|subst algoritmer|subst
17	10	hardkoding	hard|subst-modif koding|subst
18	7	grensesnittbloat	grensesnitt|subst-modif bloat|UNK
19	9	direktebuffersamsvar	direkte|subst-modif buffer|subst-modif samsvar|subst
20	9	prosessen	prosessen|subst

Continued on next page

Table B.3 – *Continued from previous page*

ID	Survey ID	Norwegian Translation	Norwegian tagged output
21	3	tråd	tråd|subst
22	3	tjæreinnholdet	tjæreinnholdet|subst
23	9	tidsplanlegging	tidsplanlegging|subst
24	9	tråder	tråder|verb
25	9	foretaksgjenstand	foretaks|subst-modif gjenstand|subst
26	9	fellesdrift	felles|subst-modif drift|subst
27	9	konstant	konstant|adj
28	8	operatør	operatør|subst
29	8	ressursforvaltning	ressurs|subst-modif forvaltning|subst
30	8	blokk	blokk|subst
31	8	symbolsk språk	symbolsk|adj språk|subst
32	7	symbolsk programmering	symbolsk|adj programmering|subst
33	8	i-overbelastningemulering	i|prep -|strek overbelastning|subst- modif emulering|subst
34	11	i-målprobe	i|prep -|strek mål|subst-modif probe|UNK
35	8	programanimation	program|subst-modif animation|UNK
36	8	programsnitting	program|subst-modif snitting|subst
37	3	enkleste arbeid .	enkleste|adj arbeid|subst .|clb
38	7	kode ombruk	kode|subst ombruk|subst
39	7	dynamisk linker	dynamisk|adj linker|UNK
40	10	dynamisk lasting	dynamisk|adj lasting|subst
41	7	gjenstand data	gjenstand|subst data|subst
42	7	lydbiblioteker	lyd|subst-modif biblioteker|subst

Continued on next page

Table B.3 – *Continued from previous page*

ID	Survey ID	Norwegian Translation	Norwegian tagged output
43	7	grafikk biblioteker	grafikk|subst biblioteker|subst
44	8	matematikk-biblioteker	matematikk-|subst biblioteker|subst
45	7	programvareutvikling- sett	programvareutvikling|subst -sett|verb
46	6	Numeriske biblioteker	Numeriske|adj biblioteker|subst
47	6	programoptimization	program|subst-modif optimization|UNK
48	6	levende koding	levende|adj koding|subst
49	6	sjefsprogrammer lag	sjefs|subst-modif programmer|UNK lag|adv
50	6	organisasjonen for laveffektopplysninger	organisasjonen|subst for|prep lav|subst-modif effekt|subst-modif opplysninger|subst
51	6	data-orientert konstruksjon	data-oriented|UNK konstruksjon|subst
52	6	inndata-utvidelser	inndata-|subst utvidelser|subst
53	6	drift reduksjon for laveffekt	drift|subst reduksjon|subst for|prep lav|subst-modif effekt|subst-modif
54	6	best programmering	best|adj programmering|subst
55	5	konkurranseprogrammering	konkurranse|subst-modif programmering|subst
56	5	beskyttelsesbyte	beskyttelses|subst-modif byte|UNK
57	5	programvare relieffkonstruksjon	programvare|subst relieff|subst-modif konstruksjon|subst
58	5	unntak håndteringssyntaks	unntak|subst håndterings|subst-modif syntaks|subst
59	5	behandling	behandling|subst

Continued on next page

Table B.3 – *Continued from previous page*

ID	Survey ID	Norwegian Translation	Norwegian tagged output
60	5	programmeringsspråk laget av kvinner	programmeringsspråk|subst laget|subst av|prep kvinner|subst
61	5	akademiske programmeringsspråk	akademiske|adj programmeringsspråk|subst
62	5	ansatt-orientert programmeringsspråk	ansatt|adj -|strek orientert|adj programmeringsspråk|subst
63	5	data-strukturert programmeringsspråk	data|subst -|strek strukturert|verb programmeringsspråk|subst
64	4	forsøksprogrammeringsspråk	forsøks|subst-modif programmeringsspråk|subst
65	4	funksjon-nivå språk	funksjon|subst -|strek nivå|subst språk|subst
66	4	funksjonelle språk	funksjonelle|adj språk|subst
67	4	gjenstand-basert programmeringsspråk	gjenstand|subst -|strek basert|verb programmeringsspråk|subst
68	4	prototype-basert programmeringsspråk	prototype|UNK -|strek basert|verb programmeringsspråk|subst
69	4	sikre programmeringsspråk	sikre|adj programmeringsspråk|subst
70	4	data-programmot	data|subst -|strek program|subst-modif mot|subst-modif
71	4	gjennomstrømning-basert programmering	gjennomstrømning|subst -|strek basert|verb programmering|subst
72	4	ufravikelig programmering	ufravikelig|adj programmering|subst
73	3	forsettlig programmering	abc|tag
74	3	flertrinns programmering	forsettlig|adj programmering|subst
75	3	organiske datamaskiner	organiske|adj datamaskiner|subst

Continued on next page

Table B.3 – *Continued from previous page*

ID	Survey ID	Norwegian Translation	Norwegian tagged output
76	3	rolle-orientert programmering	rolle|subst -|strek orientert|adj programmering|subst
77	3	vitenskapelige programspråk	vitenskapelige|adj program|subst-modif språk|subst
78	7	strenge programspråk	strenge|adj program|subst-modif språk|subst
79	3	svart alternativ	svart|adj alternativ|adj
80	9	krenke programmering	abc|tag
81	9	programvare arkitektmodell	krenke|verb programmering|subst
82	2	kontroll struktur diagram	kontroll|subst struktur|subst diagram|UNK
83	2	encellede kode	encellede|adj kode|subst
84	2	overlapper kode	overlapper|verb kode|subst
85	2	dataoverføring gjenstand	dataoverføring|subst gjenstand|subst
86	2	globale offsetbord	globale|adj offset|subst-modif bord|subst
87	2	inndata-maske	inndata-|subst maske|subst
88	2	fjernvurdering	fjern|subst-modif vurdering|subst
89	2	ikke-adusert kode	ikke|adv -|strek adusert|adj kode|subst
90	2	samle stealing	samle|verb stealing|UNK|UNK|UNK
91	1	funksjonelle opplysninger konstruksjoner	funksjonelle|adj opplysninger|subst konstruksjoner|subst
92	1	utjevning	utjevning|subst

Continued on next page

Table B.3 – *Continued from previous page*

ID	Survey ID	Norwegian Translation	Norwegian tagged output
93	1	-programspråk generelle formål	-|strek program|subst-modif språk|subst generelle|adj formål|subst
94	1	god gjenstand	god|UNK gjenstand|subst
95	1	differensierte tjenester	differensierte|adj tjenester|subst
96	1	strukturert concurrency	strukturert|verb concurrency|UNK
97	1	programmering vilt	programmering|subst vilt|subst
98	1	digitalt signal behandling	digitalt|adj signal|subst behandling|subst
99	11	sekvens bokstav	sekvens|subst bokstav|subst

Table B.4 shows the ratings made by native speakers for each translation. Columns 'ID' and 'Norwegian Translation' have the same function as in Table B.1. The column 'Ratings' indicates how many ratings this particular translation received. The first number shows how many evaluators saw the translation, and the number in parentheses shows how many knew or understood the term and subsequently rated the translation. The columns 'Natural transl.' and 'Mean. cons.' represent the degree to which evaluators rated the translation to be natural and how much of the meaning was conserved. 'Perfect', 'Natural', 'Unnatural', 'Incorrect', and 'Incompr.' represent the five alternatives for naturalness in Subsection 6.2.2. These are: (1) the translation is perfect, (2) the translation is natural, but I would not use it myself, (3) the translation is fluent, but it seems unnatural; e.g., it is not suitable for the domain, (4) the translation is incorrect, but is possible to understand; e.g., it contains Norwegian words, but has orthographic faults or has too many/too few spaces, or (5) the translation is incomprehensible; e.g., it contains Norwegian words but the translation as a whole is nonsensical, or it contains English words that cannot be used as loanwords.

Equivalently, 'All', 'Most', 'Some', 'Little', and 'None' represent the five alternatives for meaning conserved in Subsection 6.2.2. These are: (1) all of the original meaning, (2) most of the original meaning, (3) some of the original meaning, (4) little of the original meaning, or (5) no original meaning.

Table B.4.: Ratings of the machine translations.

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
1	eroperatør	5 (3)	Perfect	0%	All	0%
			Natural	0%	Most	0%
			Unnatural	0%	Some	0%
			Incorrect	0%	Little	33.33%
			Incompr.	100%	None	66.67%
2	direktivet	5 (3)	Perfect	33.33%	All	66.67%
			Natural	33.33%	Most	33.33%
			Unnatural	0.00%	Some	0.00%
			Incorrect	33.33%	Little	0.00%
			Incompr.	0.00%	None	0.00%
3	tilfelle	4 (4)	Perfect	0.00%	All	50.00%
			Natural	25.00%	Most	25.00%
			Unnatural	50.00%	Some	25.00%
			Incorrect	25.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
4	programcomprehension	5 (3)	Perfect	0.00%	All	66.67%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	33.33%
			Incorrect	33.33%	Little	0.00%
			Incompr.	66.67%	None	0.00%
5	self-datalagring	5 (3)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	33.33%
			Incorrect	66.67%	Little	66.67%
			Incompr.	33.33%	None	0.00%
6	self-prisfastsettelse	4 (3)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	33.33%
			Incompr.	100.00%	None	66.67%
7	sidevirkning	4 (4)	Perfect	0.00%	All	0.00%
			Natural	50.00%	Most	75.00%
			Unnatural	25.00%	Some	0.00%
			Incorrect	25.00%	Little	25.00%
			Incompr.	0.00%	None	0.00%
8	leketøysprogram	5 (3)	Perfect	33.33%	All	66.67%
			Natural	66.67%	Most	33.33%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
9	flyktige	5 (3)	Perfect	0.00%	All	33.33%
			Natural	0.00%	Most	66.67%
			Unnatural	33.33%	Some	0.00%
			Incorrect	33.33%	Little	0.00%
			Incompr.	33.33%	None	0.00%
10	anti-mønstre	4 (4)	Perfect	50.00%	All	75.00%
			Natural	0.00%	Most	25.00%
			Unnatural	50.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
11	programderivation	4 (4)	Perfect	25.00%	All	25.00%
			Natural	0.00%	Most	50.00%
			Unnatural	25.00%	Some	25.00%
			Incorrect	50.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
12	kringkasting	4 (4)	Perfect	50.00%	All	75.00%
			Natural	25.00%	Most	25.00%
			Unnatural	25.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
13	redusere	4 (4)	Perfect	50.00%	All	50.00%
			Natural	25.00%	Most	50.00%
			Unnatural	25.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
14	nummer-av-nummeralgoritmer	4 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	50.00%
			Unnatural	50.00%	Some	0.00%
			Incorrect	0.00%	Little	50.00%
			Incompr.	50.00%	None	0.00%
15	reduksjon	5 (3)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
16	søke-algoritmer	4 (4)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
17	hardkoding	4 (4)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
18	grensesnittbloat	5 (5)	Perfect	20.00%	All	20.00%
			Natural	0.00%	Most	20.00%
			Unnatural	0.00%	Some	20.00%
			Incorrect	0.00%	Little	40.00%
			Incompr.	80.00%	None	0.00%
19	direktebuffersamsvar	5 (5)	Perfect	0.00%	All	40.00%
			Natural	80.00%	Most	40.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	20.00%	None	20.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
20	prosessen	5 (5)	Perfect	40.00%	All	40.00%
			Natural	0.00%	Most	40.00%
			Unnatural	0.00%	Some	20.00%
			Incorrect	60.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
21	tråd	2 (2)	Perfect	100.00%	All	50.00%
			Natural	0.00%	Most	50.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
22	tjæreinnholdet	2 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	100.00%	None	100.00%
23	tidsplanlegging	5 (5)	Perfect	60.00%	All	40.00%
			Natural	0.00%	Most	40.00%
			Unnatural	40.00%	Some	0.00%
			Incorrect	0.00%	Little	20.00%
			Incompr.	0.00%	None	0.00%
24	tråder	5 (5)	Perfect	100.00%	All	80.00%
			Natural	0.00%	Most	20.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
25	foretaksgjenstand	5 (4)	Perfect	0.00%	All	25.00%
			Natural	25.00%	Most	25.00%
			Unnatural	75.00%	Some	25.00%
			Incorrect	0.00%	Little	25.00%
			Incompr.	0.00%	None	0.00%
26	fellesdrift	5 (3)	Perfect	66.67%	All	33.33%
			Natural	33.33%	Most	33.33%
			Unnatural	0.00%	Some	33.33%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
27	konstant	5 (5)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
28	operatør	4 (3)	Perfect	0.00%	All	66.67%
			Natural	66.67%	Most	0.00%
			Unnatural	33.33%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	33.33%
29	ressursforvaltning	4 (4)	Perfect	75.00%	All	50.00%
			Natural	25.00%	Most	50.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
30	blokk	4 (4)	Perfect	100.00%	All	75.00%
			Natural	0.00%	Most	25.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
31	symbolsk språk	4 (3)	Perfect	66.67%	All	100.00%
			Natural	33.33%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
32	symbolsk programmering	5 (4)	Perfect	75.00%	All	75.00%
			Natural	25.00%	Most	25.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
33	i-overbelastningemulering	4 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	50.00%
			Incorrect	50.00%	Little	50.00%
			Incompr.	50.00%	None	0.00%
34	i-målprobe	5 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	50.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	50.00%	Little	0.00%
			Incompr.	50.00%	None	50.00%
35	programanimation	4 (3)	Perfect	66.67%	All	66.67%
			Natural	0.00%	Most	33.33%
			Unnatural	0.00%	Some	0.00%
			Incorrect	33.33%	Little	0.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
36	programsnitting	4 (3)	Perfect	33.33%	All	100.00%
			Natural	33.33%	Most	0.00%
			Unnatural	33.33%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
37	enkleste arbeid .	2 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	50.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	50.00%	Little	50.00%
			Incompr.	50.00%	None	0.00%
38	kode ombruk	5 (5)	Perfect	20.00%	All	60.00%
			Natural	40.00%	Most	20.00%
			Unnatural	0.00%	Some	20.00%
			Incorrect	40.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
39	dynamisk linker	5 (5)	Perfect	60.00%	All	60.00%
			Natural	20.00%	Most	40.00%
			Unnatural	20.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
40	dynamisk lasting	4 (4)	Perfect	75.00%	All	100.00%
			Natural	25.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
41	gjenstand data	5 (5)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	40.00%	Some	20.00%
			Incorrect	20.00%	Little	40.00%
			Incompr.	40.00%	None	40.00%
42	lydbiblioteker	5 (5)	Perfect	80.00%	All	80.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
43	grafikk biblioteker	5 (5)	Perfect	40.00%	All	60.00%
			Natural	20.00%	Most	40.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	40.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
44	matematikk-biblioteker	4 (4)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
45	programvareutvikling-sett	5 (5)	Perfect	0.00%	All	20.00%
			Natural	20.00%	Most	80.00%
			Unnatural	40.00%	Some	0.00%
			Incorrect	20.00%	Little	0.00%
			Incompr.	20.00%	None	0.00%
46	Numeriske biblioteker	6 (6)	Perfect	83.33%	All	83.33%
			Natural	16.67%	Most	16.67%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
47	programoptimization	6 (6)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	100.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	66.67%	Little	0.00%
			Incompr.	33.33%	None	0.00%
48	levende koding	6 (6)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	16.67%
			Incorrect	50.00%	Little	50.00%
			Incompr.	50.00%	None	33.33%
49	sjefsprogrammer lag	6 (5)	Perfect	60.00%	All	20.00%
			Natural	20.00%	Most	20.00%
			Unnatural	20.00%	Some	60.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
50	organisasjonen for laveffektopplysninger	6 (5)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	60.00%	Some	60.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	40.00%	None	40.00%
51	data-orientert konstruksjon	6 (5)	Perfect	20.00%	All	0.00%
			Natural	20.00%	Most	20.00%
			Unnatural	60.00%	Some	60.00%
			Incorrect	0.00%	Little	20.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
52	inndata-utvidelser	6 (3)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	66.67%
			Unnatural	66.67%	Some	0.00%
			Incorrect	33.33%	Little	33.33%
			Incompr.	0.00%	None	0.00%
53	drift reduksjon for laveffekt	6 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	50.00%
			Unnatural	0.00%	Some	50.00%
			Incorrect	100.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
54	best programmering	6 (2)	Perfect	0.00%	All	0.00%
			Natural	50.00%	Most	50.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	50.00%	None	50.00%
55	konkurranse programmering	4 (3)	Perfect	66.67%	All	66.67%
			Natural	33.33%	Most	33.33%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
56	beskyttelsesbyte	4 (2)	Perfect	50.00%	All	50.00%
			Natural	50.00%	Most	50.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
57	programvare relieffkonstruksjon	4 (4)	Perfect	25.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	50.00%
			Incompr.	75.00%	None	25.00%
58	unntak håndteringssyntaks	4 (4)	Perfect	25.00%	All	50.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	50.00%
			Incorrect	75.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
59	behandling	4 (4)	Perfect	25.00%	All	50.00%
			Natural	75.00%	Most	25.00%
			Unnatural	0.00%	Some	25.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
60	programmeringsspråk laget av kvinner	4 (4)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
61	akademiske programmeringsspråk	4 (4)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
62	ansatt-orientert programmeringsspråk	4 (4)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	25.00%
			Unnatural	50.00%	Some	25.00%
			Incorrect	0.00%	Little	25.00%
			Incompr.	50.00%	None	25.00%
63	data-strukturert programmeringsspråk	4 (3)	Perfect	66.67%	All	33.33%
			Natural	33.33%	Most	66.67%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
64	forsøksprogrammerings språk	4 (4)	Perfect	0.00%	All	25.00%
			Natural	75.00%	Most	25.00%
			Unnatural	25.00%	Some	50.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
65	funksjon-nivå språk	4 (3)	Perfect	66.67%	All	33.33%
			Natural	0.00%	Most	66.67%
			Unnatural	0.00%	Some	0.00%
			Incorrect	33.33%	Little	0.00%
			Incompr.	0.00%	None	0.00%
66	funksjonelle språk	4 (4)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
67	gjenstand-basert programmeringsspråk	4 (4)	Perfect	0.00%	All	50.00%
			Natural	25.00%	Most	25.00%
			Unnatural	50.00%	Some	0.00%
			Incorrect	25.00%	Little	25.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
68	prototype-basert programmeringsspråk	4 (4)	Perfect	75.00%	All	75.00%
			Natural	0.00%	Most	25.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	25.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
69	sikre programmeringsspråk	4 (4)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
70	data-programmot	4 (3)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	33.33%	Some	33.33%
			Incorrect	0.00%	Little	33.33%
			Incompr.	66.67%	None	33.33%
71	gjennomstrømning-basert programmering	4 (3)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	66.67%
			Unnatural	66.67%	Some	33.33%
			Incorrect	33.33%	Little	0.00%
			Incompr.	0.00%	None	0.00%
72	ufravikelig programmering	4 (4)	Perfect	0.00%	All	25.00%
			Natural	25.00%	Most	0.00%
			Unnatural	25.00%	Some	0.00%
			Incorrect	25.00%	Little	75.00%
			Incompr.	25.00%	None	0.00%
73	forsettlig programmering	2 (2)	Perfect	0.00%	All	0.00%
			Natural	50.00%	Most	50.00%
			Unnatural	0.00%	Some	50.00%
			Incorrect	50.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
74	flertrinns programmering	2 (2)	Perfect	50.00%	All	50.00%
			Natural	0.00%	Most	50.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	50.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
75	organiske datamaskiner	2 (2)	Perfect	50.00%	All	50.00%
			Natural	50.00%	Most	50.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
76	rolle-orientert programmering	2 (2)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
77	vitenskapelige programspråk	2 (2)	Perfect	100.00%	All	100.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
78	strenge programspråk	5 (5)	Perfect	20.00%	All	40.00%
			Natural	40.00%	Most	40.00%
			Unnatural	20.00%	Some	20.00%
			Incorrect	20.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
79	svart alternativ	2 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	50.00%	Some	100.00%
			Incorrect	50.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
80	krenke programmering	5 (4)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	25.00%	Some	25.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	75.00%	None	75.00%
81	programvare arkitektmodell	5 (5)	Perfect	0.00%	All	20.00%
			Natural	20.00%	Most	20.00%
			Unnatural	0.00%	Some	60.00%
			Incorrect	40.00%	Little	0.00%
			Incompr.	40.00%	None	0.00%
82	kontroll struktur diagram	7 (5)	Perfect	20.00%	All	60.00%
			Natural	20.00%	Most	40.00%
			Unnatural	20.00%	Some	0.00%
			Incorrect	40.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
83	encellede kode	7 (7)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	28.57%
			Incorrect	28.57%	Little	42.86%
			Incompr.	71.43%	None	28.57%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
84	overlapper kode	7 (7)	Perfect	14.29%	All	28.57%
			Natural	14.29%	Most	42.86%
			Unnatural	0.00%	Some	14.29%
			Incorrect	57.14%	Little	14.29%
			Incompr.	14.29%	None	0.00%
85	dataoverføring gjenstand	7 (3)	Perfect	33.33%	All	33.33%
			Natural	0.00%	Most	33.33%
			Unnatural	33.33%	Some	33.33%
			Incorrect	33.33%	Little	0.00%
			Incompr.	0.00%	None	0.00%
86	globale offsetbord	7 (5)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	20.00%
			Unnatural	20.00%	Some	20.00%
			Incorrect	20.00%	Little	20.00%
			Incompr.	60.00%	None	40.00%
87	inndata-maske	7 (5)	Perfect	80.00%	All	80.00%
			Natural	20.00%	Most	0.00%
			Unnatural	0.00%	Some	20.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
88	fjernvurdering	7 (5)	Perfect	40.00%	All	20.00%
			Natural	40.00%	Most	60.00%
			Unnatural	20.00%	Some	20.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
89	ikke-adusert kode	7 (1)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	100.00%	None	100.00%
90	samle stealing	7 (4)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	25.00%	Some	25.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	75.00%	None	75.00%
91	funksjonelle opplysninger konstruksjoner	4 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	50.00%	Some	0.00%
			Incorrect	0.00%	Little	200.00%
			Incompr.	50.00%	None	0.00%

Continued on next page

Table B.4 – *Continued from previous page*

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
92	utjevning	4 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	50.00%
			Unnatural	50.00%	Some	50.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	50.00%	None	0.00%
93	-programspråk generelle formål	4 (4)	Perfect	0.00%	All	25.00%
			Natural	0.00%	Most	25.00%
			Unnatural	25.00%	Some	0.00%
			Incorrect	50.00%	Little	25.00%
			Incompr.	25.00%	None	25.00%
94	god gjenstand	4 (2)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	100.00%	None	100.00%
95	differensierte tjenester	4 (3)	Perfect	66.67%	All	66.67%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
96	strukturert concurrency	4 (3)	Perfect	0.00%	All	33.33%
			Natural	33.33%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	33.33%	None	33.33%
97	programmering vilt	4 (3)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	33.33%
			Incorrect	33.33%	Little	0.00%
			Incompr.	66.67%	None	66.67%
98	digitalt signal behandling	4 (4)	Perfect	50.00%	All	75.00%
			Natural	0.00%	Most	25.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	50.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
99	sekvens bokstav	5 (3)	Perfect	0.00%	All	0.00%
			Natural	0.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	33.33%	Little	66.67%
			Incompr.	66.67%	None	33.33%

Table B.5 shows the ratings made by native speakers for the control translations. The columns have the same representations as in Table B.4.

Table B.5.: Ratings of the control translations.

ID	Norwegian translation	Ratings	Natural transl.		Mean. cons.	
100	splitt og hersk-algoritme	4 (3)	Perfect	100%	All	100%
			Natural	0%	Most	0%
			Unnatural	0%	Some	0%
			Incorrect	0%	Little	0%
			Incompr.	0%	None	0%
101	syntaksfremheving	7 (7)	Perfect	100%	All	85.71%
			Natural	0%	Most	14.29%
			Unnatural	0%	Some	0%
			Incorrect	0%	Little	0%
			Incompr.	0%	None	0%
102	kildekode-editor	2 (2)	Perfect	0.00%	All	50.00%
			Natural	50.00%	Most	0.00%
			Unnatural	50.00%	Some	5.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
103	prosedyrisk programmering	4 (3)	Perfect	33.33%	All	66.67%
			Natural	66.67%	Most	33.33%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
104	kooperativ fleroppgavekjøring	4 (3)	Perfect	0.00%	All	0.00%
			Natural	66.67%	Most	66.67%
			Unnatural	33.33%	Some	33.33%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
105	parallellprogrammering	6 (5)	Perfect	60.00%	All	20.00%
			Natural	20.00%	Most	20.00%
			Unnatural	20.00%	Some	60.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
106	skedulering	5 (5)	Perfect	40.00%	All	80.00%
			Natural	40.00%	Most	0.00%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	20.00%
			Incompr.	20.00%	None	0.00%

Continued on next page

Table B.5 – *Continued from previous page*

ID	Norwegian Translation	Ratings	Natural transl.		Mean. cons.	
107	designmønster	4 (4)	Perfect	50.00%	All	0.00%
			Natural	25.00%	Most	50.00%
			Unnatural	25.00%	Some	50.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%
108	runtimebibliotek	5 (5)	Perfect	20.00%	All	20.00%
			Natural	0.00%	Most	60.00%
			Unnatural	0.00%	Some	20.00%
			Incorrect	40.00%	Little	0.00%
			Incompr.	40.00%	None	0.00%
109	på stedet-algoritme	4 (3)	Perfect	0.00%	All	0.00%
			Natural	66.67%	Most	66.67%
			Unnatural	0.00%	Some	0.00%
			Incorrect	0.00%	Little	33.33%
			Incompr.	33.33%	None	0.00%
110	programvareavhengigheter	5 (3)	Perfect	33.33%	All	33.33%
			Natural	33.33%	Most	66.67%
			Unnatural	33.33%	Some	0.00%
			Incorrect	0.00%	Little	0.00%
			Incompr.	0.00%	None	0.00%

C. Installation Guide

This appendix describes how to setup the translation system as needed to reproduce the experiments. This includes installing OBT and Moses and their respective required packages. These are the commands the author used on a particular Ubuntu 18.04 system, and the commands are not guaranteed to work neither for this OS or any other.

C.1. Contents of the Attached Code

```
root folder
├── OBT_scripts
│   ├── tagged_obt
│   ├── untagged
│   └── tag_all_untagged
├── semesteroppgave
│   ├── nor_comp_clean
│   ├── secos_extra
│   ├── technicalTerms
│   ├── test_set/
│   ├── automatic_test_sets.py
│   ├── baseline_sjobergh.py
│   ├── exception.py
│   ├── extr_words_and_pos_from_NKL.py
│   ├── globals.py
│   ├── hybrid_sjobergh.py
│   ├── measurements.py
│   ├── NDTCorpusReader.py
│   ├── ngrams_sjobergh.py
│   ├── nor030224NST.txt
│   ├── NorKompLeksCorpusReader.py
│   ├── norLexDBExtr.py
│   ├── pos_sjobergh.py
│   └── prob_pos_combos.py
└── ud_corpus
    ├── 20150601_ud.nb
    │   ├── nb_no
    │   └── 20150601_lesmeg
    └── clean
```

```
        ├── en
        └── nb
    ├── check_no_tags_in_clean
    ├── clean_ud_corpus.py
├── unmarkedEpos
    ├── clean_ud
    ├── clean_ud_en
    ├── data
    ├── nor_comp_clean
    ├── tagged_comp
    ├── tagged_obt
    ├── concat_files
    ├── exception
    ├── globals
    ├── hybrid_sjobergh
    ├── measurements
    ├── NDTCorpusReader
    ├── ngrams_sjobergh
    ├── NorKompLeksCorpusReader
    ├── nst_database
    ├── pos_sjobergh
    ├── prob_pos_combos
    ├── same_num_of_lines
    ├── tag_obt_comp
    ├── tagUnmarkedEpos.py
    └── concat_files
├── utenriks_dep_corp
    ├── corpus
    │   └── training
    │       ├── concat.en
    │       ├── concat.nb
    │       ├── concat_tagged.en
    │       └── concat_tagged.nb
    ├── test_sets
    ├── auto_test.py
    ├── auto_test
    └── lowercase_and_split_dash.py
└── Survey_answers
```

The directories have the following functions

- `OBT_scripts`: Scripts for automatic tagging using OBT
  - `tagged_obt`: Files that have been tagged by OBT
  - `untagged`: Files that are to be tagged by OBT

- `semesteroppgave`: Holds code related to the compound splitters
  - `nor_comp_clean`: Compounds and lexemes extracted from the NST database.
  - `secos_extra`: Files used for testing the performance of SECOS.
  - `technicalTerms`: Unused tagged compounds. See the `README` file in this location for details.
  - `test_sets`: The test sets described in Subsection 4.4.1. Test Sets and Evaluation for the Compound Splitters.
  - `automatic_test_sets.py`: A script used to automatically test the four implemented splitters.
  - `baseline_sjobergh.py`: The implemented version of the baseline splitter.
  - `exception.py`: Helper script used by the splitters.
  - `extr_words_and_pos_from_NKL.py`: Script to extract words and corresponding parts-of-speech from the NKL database.
  - `globals.py`: Helper script used by the splitters.
  - `hybrid_sjobergh.py`: The implemented version of the hybrid splitter.
  - `measurements.py`: Helper script used by the splitters.
  - `NDTCorpusReader.py`: An implemented corpus reader for the NDT corpus.
  - `ngrams_sjobergh.py`: The implemented version of the n-gram splitter.
  - `nor030224NST.txt`: The raw contents of the NST database, in txt format.
  - `NorKompLeksCorpusReader.py`: An alternative corpus reader for the NKL corpus.
  - `norLexDBExtr.py`: Script to extract compounds and corresponding parts-of-speech from the NKL database.
  - `pos_sjobergh.py`: The implemented version of the POS splitter.
  - `prob_pos_combos.py`: Helper script used by the splitters.
- `ud_corpus`: Corpus files constructed from the "Translation memories from The Ministry of Foreign Affairs of Norway" (see Subsection 4.4.1).
  - `20150601_ud.nb`: The download of the "Translation memories from The Ministry of Foreign Affairs of Norway".
    - `nb_no`: The 18 English-Norwegian translation memories.
    - `20150601_lesmeg`: Info supplied by *Utenriksdepartementet* about the translation memories.
  - `clean`: The cleaned corpus text, split into an English and a Norwegian directory.
    - `en`: The English side of the cleaned corpus text.
    - `nb`: The Norwegian side of the cleaned corpus text.
  - `check_no_tags_in_clean`: Helper script ensuring that all translation memory tags have been removed.
  - `clean_ud_corpus.py`: Script for cleaning the translation memory.
- `unmarkedEpos`: The modified POS compound splitter used to tag compounds.

  - `clean_ud`: Copy of `ud_corpus/clean/nb`.
  - `clean_ud_en`: Copy of `ud_corpus/clean/en`.
  - `data`: Various source files used in corpus extraction
  - `nor_comp_clean`: Copy of `semesteroppgave/nor_comp_clean`.
  - `tagged_comp`: Output of compound tagging.
  - `tagged_obt`: Input for compound tagging. Norwegian text tagged by OBT.
  - `concat_files`: Helper script for concatenating the tagged files in `/tagged_comp`.
  - `exception`: Copy of `semesteroppgave/exception`.
  - `globals`: Copy of `semesteroppgave/globals`.
  - `hybrid_sjobergh`: Copy of `semesteroppgave/hybrid_sjobergh`.
  - `measurements`: Copy of `semesteroppgave/measurements`.
  - `NDTCorpusReader`: Copy of `semesteroppgave/NDTCorpusReader`.
  - `ngrams_sjobergh`: Copy of `semesteroppgave/ngrams_sjobergh`.
  - `NorKompLeksCorpusReader`: Copy of `semesteroppgave/NorKompLeksCorpusReader`.
  - `nst_database`: Copy of `nor030224NST.txt`.
  - `pos_sjobergh`: Modified version of `semesteroppgave/pos_sjobergh` that is used to tag compounds.
  - `prob_pos_combos`: Copy of `semesteroppgave/prob_pos_combos`.
  - `same_num_of_lines`: Helper script for ensuring that the concatenated training files have the same number of lines and can be sentence aligned.
  - `tag_obt_comp`: Wrapper script for tagging all compounds in the text files in `tagged_obt`.
  - `tagUnmarkedEpos.py`: Modified version of `pos_sjobergh` that performs the compound splitting.
  - `concat_files`: Helper script for concatenating all files in `tagged_obt`.

- `utenriks_dep_corp`: The directory where training and testing takes place.
  - `corpus`: Location of training corpus and models created during training.
    - `training`: The location of the training corpus.
      - `concat.en`: Clean, untagged English corpus text.
      - `concat.nb`: Norwegian counterpart of concat.en.
      - `concat_tagged.en`: Copy of `concat_en.txt`. Needed for factored training,where Moses uses two files with the same name, only distinguished by language extension.
      - `concat_tagged.nb`: Tagged Norwegian corpus text.
  - `test_sets`: Holds English terms to be translated.
  - `auto_test.py`: Helper script for extracting the translation and tags from the Moses output.
  - `auto_tets`: Wrapper script for translating all English terms located in test_sets/test_set.txt.
  - `lowercase_and_split_dash.py`: Helper script for doing preprocessing on English input terms.

- `Survey_answers`: Holds raw data from the survey in the form of `.csv` files as well as a human-readable overview in `Evaluations.xslx`.

C.2. Requirements for the Compound Splitters

The splitters are implemented using Python3, and *SECOS* uses Python2.7. The different versions can be found on Python's Download page[1].

The splitters use some of the built-in features of the NLTK toolkit. Follow the installation instructions on the official page[2].

C.3. Installing Moses and Requirements

This guide is largely based on that of the sntllaventhiran blog[3].

1. First, install required Ubuntu packages:
   ```
   sudo apt-get install build-essential git-core pkg-config automake
   libtool wget zlib1g-dev python-dev libbz2-dev
   ```

2. Clone the Moses decoder from its repository and move into the directory to build.
   ```
   git clone https://github.com/moses-smt/mosesdecoder.git
   cd mosesdecoder
   ```

3. Install remaining packages needed
   ```
   sudo apt-get install g++ git subversion automake libtool zlib1g-dev
   libboost-all-dev libbz2-dev liblzma-dev python-dev graphviz
   imagemagick make cmake libgoogle-perftools-dev (for tcmalloc)
   ```

4. Download and build SRILM:
   Follow the instructions from the Speech Technology and Research (STAR) Laboratory[4], and install it to the home directory; e.g., ∼`/srilm-1.7.3`.

5. Download Boost from Sourceforge[5] and extract it to the home directory. Then move into the directory and build:
   ```
   cd boost_1_66_0/
   ./bootstrap.sh
   ./b2 -j3 -prefix=$PWD -libdir=$PWD/lib64 -layout=tagged link=static
   threading=multi,single install || echo FAILURE
   ```

[1] `https://www.python.org/downloads/`
[2] `https://www.nltk.org/install.html`
[3] `http://sntllaventhiran.blogspot.com/2016/06/how-to-install-moses-on-ubuntu-1604-x64.html`
[4] `http://www.speech.sri.com/projects/srilm/docs/INSTALL`
[5] `https://sourceforge.net/projects/boost/files/`

This will install the library file in the `lib64` directory, and not in the system directory, which is important when installing Moses.

6. Install Moses with `bjam`
   ```
   cd mosesdecoder/
   ./bjam -j3
   ```

7. Compile Moses. Once boost is installed, tell Moses where `boost` is located using the `-with-boost` flag.
   ```
   ./bjam -with-boost=~/boost_1_66_0 -with-srilm=~/srilm-1.7.3 -j8
   ```

8. Install `GIZA++`:
   ```
   git clone https://github.com/moses-smt/giza-pp
   cd giza-pp
   make
   ```

   This should create the three binaries `~/giza-pp/GIZA++-v2/GIZA++`, `~/giza-pp/GIZA++-v2/snt2cooc.out` and `~/giza-pp/mkcls-v2/mkcls`. Copied these compiled executables to a `bin/` directory, so that Moses can find them:
   ```
   cd
   mkdir bin
   cp giza-pp/GIZA++-v2/GIZA++ bin/
   cp giza-pp/mkcls-v2/mkcls bin/
   cp giza-pp/GIZA++-v2/snt2cooc.out bin/
   ```

   The `bin` directory will be the location used by the training script and is specified by the `-external-bin-dir` switch (see Subsection 5.3.4).

C.4. Installing OBT

After having cloned that The Oslo-Bergen Tagger, it requires some additional packages for the individual parts to function. The Multitagger with lexicon for Norwegian Bokmål and Nynorsk[6], and the statistical module OBT-stat[7] have to be installed in the root folder of OBT. These modules are written in Python and Ruby, so these softwares need to be available on the host as well. The CG3-compiler (VISL-CG3)[8], supplied by The University of Southern Denmark, is also needed, but is installed centrally in `/usr/local/bin`. The part-of-speech tagger `HunPos` may pose problems; for Linux, it is only available as a 32-bit version. All the training and

[6] `https://github.com/textlab/mtag`
[7] `https://github.com/andrely/OBT-Stat`
[8] `https://visl.sdu.dk/cg3ide.html`

tagging presented in this thesis was conducted on a 64-bit Ubuntu system, meaning that the 32-bit GNU C Libraries had to be installed. If error messages of the type `/hunpos/hunpos-1.0-linux/hunpos-tag: not found` are encountered, the necessary libraries can be installed with `sudo apt-get install gcc-multilib`.

When the above steps have been completed, copy the files in `OBT_scripts` into the root folder of `The-Oslo-Bergen-Tagger`.

C.5. Final File Hierarchy

When the above installation instructions have been completed, the file hierarchy should resemble the structure below and contain at least the following files:

```
home directory
├── bin
├── giza-pp
├── mosesdecoder
├── semesteroppgave
│   ├── nor_comp_clean
│   ├── secos_extra
│   └── test_sets
├── srilm-1.7.3
├── The-Oslo-Bergen-Tagger
│   ├── tagged_obt
│   └── untagged
├── ud_corpus
│   ├── 20150601_ud.nb
│   └── clean
│       ├── en
│       └── no
├── unmarkedEpos
│   ├── clean_ud
│   ├── clean_ud_en
│   ├── data
│   ├── nor_comp_clean
│   ├── tagged_comp
│   └── tagged_obt
└── utenriks_dep_corp
    ├── corpus
    │   └── training
    └── test_sets
```

The directories have the following functions

- `bin` holds binary files used by Moses.
- `giza-pp` holds the code for the implementation of Giza++.
- `mosesdecoder` holds the scripts and configuration files for the Moses decoder. Details of the contents will not be discussed in more detail here, as mainly the scripts are utilised.
- `semesteroppgave` is the location in which all the implemented splitters are contained. `nor_comp_clean` holds the compound training data, `secos_extra` holds test sets and an automatic testing script for *SECOS*, and `test_sets` contains the test sets for the other splitters.
- `srilm-1.7.3` holds the configuration files used by the SRILM tool. Details of this

these files will not be further discussed here.

- `The-Oslo-Bergen-Tagger` holds tagger tools used by OBT as well as an automatic tagger script, `tag_all_untagged`. This script tags all the texts contained in `untagged`, placing the tagged output in `tagged_obt`.
- `ud_corpus` holds the original translation memory files and a script for separating the Norwegian and English texts as described in Subsection 5.3.1.
- `unmarkedEpos` holds the implementation of the modified POS method used for compound splitting in the training phase. `clean_ud` and `clean_ud_en` hold copies of the Norwegian and English translation memories. `data` and `nor_comp_clean` hold the same data as in `semesteroppgave`. `tagged_comp` and `tagged_obt` contain the output and input files from the tagger. The input files are texts tagged by OBT, and the compound tagger only treats words tagged as `samset-leks`.
- `utenriks_dep_corp` holds the data used by the translation system. `corpus/training holds` four files, tagged Norwegian forms, Norwegian surface forms and two copies of English surface forms. Two copies are necessary, because Moses requires two files with the same name (separated by different language extensions) for each factor; i.e., two for the surface form and two for the tagged form. `test_sets` hold the English terms that are to be translated.

D. Reproducing Experiments

This appendix describes how to reproduce the experiments presented in Chapter 6, and how run the translation system with the same configuration as was used in the experiments.

D.1. Reproducing the Splitter Experiments

To reproduce the results listed in Table 6.1a and Table 6.1b, run the following command from the root folder of the splitter, `~/semesteroppgave`:

```
python3 automatic_test_sets.py -f compound_file
```

This will run all the methods of Sjöbergh and Kann (2004) and output the correct and incorrect splits, along with performance of the test set specified by `compound_file`. The performance metrics, as well as details on their calculations, are described in Subsection 4.4.2. The file `compound_file` is a tab separated text file that contains a compound in the first column, and the correct split, separated by hyphens, in the second column.

Individual methods can also be applied to the test set of one's choosing. For example, to test the *Ambiguous* test set on the baseline method, simply execute the following command:

```
python3 baseline_sjobergh.py -f test_sets/ambiguous.txt
```

To reproduce the results in Table 6.1c, download *SECOS* from its GitHub repo[1], as well as `wikipedia_no_tokenized_clean_w2v_skip_w5_n5_s500.dt.cand.gz` from the Word2Vec models[2], and `wikipedia_no_tokenized_trigram__WordCount.gz` from the JoBimText models[3]. Extract them, and place them in the *SECOS* root directory. Next, download and copy the test sets to be tested from `semesteroppgave/secos_extra/` to the *SECOS* root directory. Here, run the following script:

```
python2.7 test_secos_comp_corr_splits.py compound_file
```

As above, `compound_file` is a tab separated text file with the compound in the first column and the correct split in the second column. The names of the test sets are listed in Section 4.4. One can also test individual compounds by running

[1] `https://github.com/riedlma/SECOS`
[2] `http://ltdata1.informatik.uni-hamburg.de/SECOS/models_word2vec/`
[3] `http://ltdata1.informatik.uni-hamburg.de/SECOS/models_jobimtext/`

```
python2.7 decompound_secos.py
    secos_data/wikipedia_no_tokenized_clean_w2v_skip_w5_n5_s500.dt.cand
    secos_data/wikipedia_no_tokenized_trigram__WordCount 50 compound_file
    0 3 3 5 3 lower 0.01
```

This will output the method *SECOS* considered best for the compounding, possible splits using all available methods and some other details. To see the performance, run the following:

```
cat compound_file | python2.7 eval_decompounding.py 0 1 debug
```

See the *SECOS* documentation for details on parameters.

D.2. Generating the Training Data

The fully tagged training data are given in the attached zip-file, but for completeness, the entire tagging process is described here.

1. Start by separating the English and Norwegian translation variants in `ud_corpus` by running `python3 clean_ud_corpus.py` from that location.
2. Copy the desired Norwegian texts to `The-Oslo-Bergen-Tagger/untagged` and run `tag_all_untagged` from `The-Oslo-Bergen-Tagger`.
3. The resulting tagged files are contained in `tagged_obt`. Copy the files from this folder into `unmarkedEpos/tagged_obt`.
4. Also copy the cleaned Norwegian and English translation texts from `ud_corpus` into `unmarkedEpos/clean_ud` and `unmarkedEpos/clean_ud_en`, respectively.
5. Run `tag_obt_comp` from `unmarkedEpos` to decompose the unknown compounds. This step can take a substantial amount of time depending on the number of unknown compounds in the corpus and the length of these.
6. Once the compound analysis is complete, run the script `concat_files.sh` from `unmarkedEpos` to concatenate Norwegian and English texts into the parallel training corpus. This outputs the four files `concat.en`, `concat_tagged.en`, `concat.nb`, and `concat_tagged.nb`.
7. Copy the above four files to `utenriks_dep_corp/corpus/training`. The training corpus is now complete and can be used to train the decoder.

D.3. Training the Decoder

To train the translation system, ensure that Moses the packages in Section C.3 have been installed, and that the training data have been obtained, either through the instructions in Section D.2 or by using the attached files. There should now be four files in `utenriks_dep_corp/corpus/training`. These are:

- `concat.en`: clean, untagged English corpus text.
- `concat_tagged.en`: copy of `concat.en`. Needed for factored training, where Moses uses two files with the same name, only distinguished by language extension.

- `concat.nb`: Norwegian counterpart of `concat.en`
- `concat_tagged.nb`: tagged Norwegian corpus text.

The training process largely consists of creating a standard and a factorised language model that the decoder can be trained on. The different scripts used have different requirements for absolute paths. Whenever the tilde (`~`) is used, relative paths are allowed, and whenever `/home/maren/` is used, absolute paths are required. The reader should exchange the `maren` for the appropriate user on their own system in the below commands.

1. First create the surface form language model. Tokenise the English and Norwegian raw text. Note that the Swedish tokeniser is used here for practical reasons, since Moses does not already have a Norwegian one. This step may be performed in another way, and copied to `concat.tok.nb` if an external Norwegian tokeniser is available.

```
~/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en <
    corpus/training/concat.en > corpus/concat.tok.en
~/mosesdecoder/scripts/tokenizer/tokenizer.perl -l sv <
    corpus/training/concat.nb > corpus/concat.tok.nb
```

2. Train the truecaser to extract some statistics about the corpus:

```
~/mosesdecoder/scripts/recaser/train-truecaser.perl --model
    corpus/truecase-model.en --corpus corpus/concat.tok.en
~/mosesdecoder/scripts/recaser/train-truecaser.perl --model
    corpus/truecase-model.nb --corpus corpus/concat.tok.nb
```

3. Perform the actual truecasing on the tokenised corpus:

```
~/mosesdecoder/scripts/recaser/truecase.perl --model
    corpus/truecase-model.en < corpus/concat.tok.en >
    corpus/concat.true.en
~/mosesdecoder/scripts/recaser/truecase.perl --model
    corpus/truecase-model.nb < corpus/concat.tok.nb >
    corpus/concat.true.nb
```

4. Clean the corpus and limit the sentence length to 80:

```
~/mosesdecoder/scripts/training/clean-corpus-n.perl
    corpus/concat.true en nb corpus/concat.clean 1 80
```

5. Create the surface form language model using SRILM's tools:

```
cd
srilm-1.7.3/bin/i686-m64/ngram-count -order 3 -interpolate
    -kndiscount -unk -text
    ~/utenriks_dep_corp/corpus/concat.clean.nb -lm
    /home/maren/utenriks_dep_corp/surface.lm
```

6. Now, make the factored language model. First, move back to the corpus location and delete the tokenised and truecase model files created previously:

```
cd ~/utenriks_dep_corp/
rm corpus/concat.* corpus/truecase-model.*
```

7. Tokenise the English corpus file:

```
cd ~/utenriks_dep_corp/
~/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en <
    corpus/training/concat_tagged.en > corpus/concat.tok.en
```

   The tagging made by OBT effectively works as tokenisation, so `concat_tagged.nb` does not need to be tokenised. Copy it to the corpus directory and rename it to `concat.tok.nb`:

```
cp corpus/training/concat_tagged.nb corpus/concat.tok.nb
```

8. Train the truecase model:

```
~/mosesdecoder/scripts/recaser/train-truecaser.perl --model
    corpus/truecase-model.en --corpus corpus/concat.tok.en
~/mosesdecoder/scripts/recaser/train-truecaser.perl --model
    corpus/truecase-model.nb --corpus corpus/concat.tok.nb
```

9. Truecase the tokenised corpus:

```
~/mosesdecoder/scripts/recaser/truecase.perl --model
    corpus/truecase-model.en < corpus/concat.tok.en >
    corpus/concat.true.en
~/mosesdecoder/scripts/recaser/truecase.perl --model
    corpus/truecase-model.nb < corpus/concat.tok.nb >
    corpus/concat.true.nb
```

10. Clean the corpus and limit the sentence length to 80:

```
~/mosesdecoder/scripts/training/clean-corpus-n.perl
    corpus/concat.true en nb corpus/concat.clean 1 80
```

11. Create the factored language model using SRILM's tools:

```
cd
srilm-1.7.3/bin/i686-m64/ngram-count -order 3 -interpolate
    -kndiscount -unk -text
    ~/utenriks_dep_corp/corpus/concat.clean.nb -lm
    /home/maren/utenriks_dep_corp/pos.lm
```

Verify that `surface.lm` and `pos.lm` are located in `utenriks_dep_corp/`.

12. Train the decoder:

```
cd ~/mosesdecoder
nohup nice scripts/training/train-model.perl --root-dir
    /home/maren/utenriks_dep_corp --external-bin-dir /home/maren/bin
    --corpus /home/maren/utenriks_dep_corp/corpus/concat.clean -f en
    -e nb --lm 0:3:/home/maren/utenriks_dep_corp/surface.lm --lm
    1:3:/home/maren/utenriks_dep_corp/pos.lm --translation-factors
    0-0,1 &> /home/maren/utenriks_dep_corp/training.out &
```

This script runs the Moses training script with the configuration specified in Subsection 5.3.4. The script is run as a background process, and will not produce any output in the terminal. The progress of the training script can be followed with the command `watch tail -f work/training.out`. The last step should look similar to: `(9) create moses.ini @ Tue Dec 12 12:43:38 CET 2019`.

When the training is done, the translation system is operational. Sanity test it with the following command:

```
echo "list of algorithms"| ~/mosesdecoder/bin/moses -f
    ~/utenriks_dep_corp/model/moses.ini
```

This should produce an output similar to the following:

```
...
liste over algoritmer
BEST TRANSLATION: liste|subst over|prep algoritmer|subst [111]
    [total=-131.951]
```

The translation process may be slow, and Moses recommends that the phrase-table and reordering models be binarised. This was not done in this thesis, but instructions on how to do this are covered by the Moses documentation[4].

As mentioned in Section 2.2 and Section 9.3, no tuning was done in this project. The Moses documentation page on tuning[5] describes tuning algorithms used in statistical machine translation.

D.4. Reproducing Novel Compounds

To reproduce the translations listed in Subsection 6.2.3 and Appendix B, ensure that the four training files are contained in `utenriks_dep_corp/corpus/training`, and that the above sanity test works as intended. Then, run the script `auto_test.sh` from the folder `utenriks_dep_corp`.

```
cd ~/utenriks_dep_corp
./ auto_test.sh
```

[4] `http://www.statmt.org/moses/?n=Moses.Baseline`
[5] `http://www.statmt.org/moses/?n=FactoredTraining.Tuning`

This will treat each line in `test_sets/test_set.txt` by lowercasing and splitting any words in the line containing hyphens. This preprocessed English term is then passed to the decoder for translation. Subsequently, the surface form and the tagged form of the decomposed translation is extracted from the output of the decoder. Finally, the postprocessing rules discussed in Subsection 5.3.5 are applied to the surface forms and the results are stored in `transl_test_set.txt`. The tagged decomposed output is stored in `transl_test_set_tags.txt`.

Maren Helene Orvedal

Generating Novel Compounds in Technical Translations

NTNU
Norwegian University of
Science and Technology