University of Helsinki
Department of Digital Humanities
Language Technology

Master's thesis

Multilingual paraphrase grammars for machine translation evaluation

Tommi Nieminen
012364433

Supervisor: Jörg Tiedemann September 23, 2018

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Tiedekunta/Osasto – Fakultet/Sektion – Faculty	Laitos – Institution – Department	
Humanistinen tiedekunta	Nykykielten laitos	
Tekijä – Författare – Author		
Tommi Nieminen		
Työn nimi – Arbetets titel – Title		
Multilingual paraphrase grammars for machine translation evaluation		
Oppiaine – Läroämne – Subject		
Kieliteknologia		
Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä– Sidoantal – Number of pages
Pro gradu	09/2018	127

Tiivistelmä – Referat – Abstract

Konekäännösten laadun arviointiin on kehitetty erilaisia menetelmiä 1950-luvulta lähtien. Aluksi laadunarviointimenetelmät olivat lähes yksinomaan manuaalisia, eli ne perustuivat kohdekielen osaajien subjektiivisiin arvioihin konekäännöksen laadusta. 1990-luvulla otettiin käyttöön ensimmäiset automaattiset arviointimenetelmät.

Pitkäkestoisesta ja laajasta tutkimuksesta huolimatta sekä manuaaliset että automaattiset arviointimenetelmät ovat edelleen epäluotettavia. Manuaalisten menetelmien ongelmana on se, että eri arvioijien tekemät arviot eivät ole johdonmukaisia. Automaattiset menetelmät taas perustuvat yleensä konekäännöksen vertaamiseen ihmiskääntäjän tekemään yksittäiseen vertailukäännökseen. Lähes jokaiselle lähdelauseelle on olemassa suuri määrä mahdollisia käännöksiä, joten automaattiset menetelmät arvioivat hyvin usein käännökset väärin.

Tässä tutkielmassa kuvataan uudenlainen automaattinen menetelmä konekäännösten laadun arviointia varten. Menetelmän testiaineisto koostuu englanninkielisistä lähdelauseista, joista jokaiselle on käytettävissä erittäin laaja joukko suomenkielisiä vertailukäännöksiä. Testiaineisto perustuu manuaalisesti laadittuihin monikielisiin kielioppeihin, jotka ovat eräänlaisia semanttisia malleja, joilla on erilaisia ilmentymiä lähde- ja kohdekielessä. Lähdekielen ilmentymät muodostavat lähdelauseiden joukon ja kohdekielen ilmentymät vertailulauseiden joukon. Semanttiset mallit sisältävät semanttisia muuttujia, jotka lisäävät vaihtelevuutta testiaineistoon.

Lähdelauseiden konekäännöksiä verrataan vertailukäännöksiin käyttämällä äärellistilaisia menetelmiä, jotka mahdollistavat konekäännöstä eniten muistuttavan vertailukäännöksen tehokkaan etsimisen. Äärellistilaisten siirtymien avulla voidaan myös seurata, millaisia muutoksia konekäännökseen on tehtävä, jotta sen voi muuttaa sitä eniten muistuttavaksi vertailulauseeksi. Tämä mahdollistaa yksityiskohtaisten virheanalyysien laatimisen, joiden avulla voidaan analysoida konekäännösjärjestelmien vahvuuksia ja heikkouksia.

Tutkielman menetelmää arvioidaan kääntämällä testiaineisto kahdeksalla erilaisella konekäännösjärjestelmällä, jotka perustuvat erilaisiin konekäännösmenetelmiin. Konekäännökset käsitellään sen jälkeen menetelmällä. Menetelmän toimivuutta arvioidaan vertaamalla sen tuottamaa virheanalyysia kahden arvioijan tekemiin manuaalisiin virheannotaatioihin sekä testaamalla, pystyykö menetelmä erottamaan ihmiskääntäjien käännökset konekäännöksistä luotettavasti.

Menetelmän arviointi osoittaa, että se on riittävän luotettava antamaan yksityiskohtaisia tietoja konekäännösjärjestelmien ominaisuuksista. Menetelmän tulokset ovat myös yhdenmukaisia julkaistujen konekäännöksen virheanalyysia käsittelevien artikkelien tulosten kanssa. Menetelmä siis soveltuu ongelmien automaattiseen havaitsemiseen konekäännösjärjestelmien kehittämisen aikana, mikä on sen pääasiallinen käyttötarkoitus.

Avainsanat – Nyckelord – Keywords
konekäännös, arviointi, machine translation, evaluation

Säilytyspaikka – Förvaringställe – Where deposited
Keskustakampuksen kirjasto

Muita tietoja – Övriga uppgifter – Additional information

Contents

1 Introduction

The field of machine translation (MT) has advanced rapidly during the last two decades. The introduction of statistical machine translation methods (SMT) around the turn of the millennium inspired a wave of new research into MT. At the same time, the development of translation technology and workflows and the increasing use of Internet by the general public created fresh demand for MT products and services. In the past few years, the adoption of artificial neural networks for MT (NMT) has inspired yet another wave of intense MT research, and has almost certainly improved MT quality dramatically for many language pairs.

The quality improvement brought by NMT has to be qualified in many ways, since the concept of translation quality is vague and context-dependent. Machine translation evaluation (MTE) is a field that has developed in parallel with MT. It aims to describe different aspects of translation quality and offer methods of measuring them. However, even though research in MTE, as in MT, has been intense during the past two decades, there have been no breakthroughs comparable to NMT. The mainstays of MTE are still scale-based human evaluations and automated metrics utilizing reference translations, despite a myriad of known problems affecting both of these approaches.

The introduction of NMT has increased interest in novel automated methods of MTE, as traditional automated metrics have displayed a bias against NMT (Shterionov et al., 2018), and the higher quality level of NMT necessitates more fine-grained and informative MTE methods. One might say, that as the quality of MT approaches that of human translation (HT), it is necessary to adopt MTE methods that resemble more those used to evaluate HT. The main method of human translation evaluation (HTE) is recording the morphological, syntactical, semantical, stylistic and typographical errors that a translator makes, which is colloquailly known as proofreading, revising or editing and more formally as translation quality assessment (TQA). In MTE, a related method is known as error analysis (Vilar et al., 2006), and it is the method of choice in most of the current work that attempts to better analyze the strengths and weaknesses of NMT systems (see for example Klubicka et al. (2018), Bentivogli et al. (2016b)). New methods in automated error analysis have been recently explored in Burlot and Yvon (2017) and Sennrich (2016).

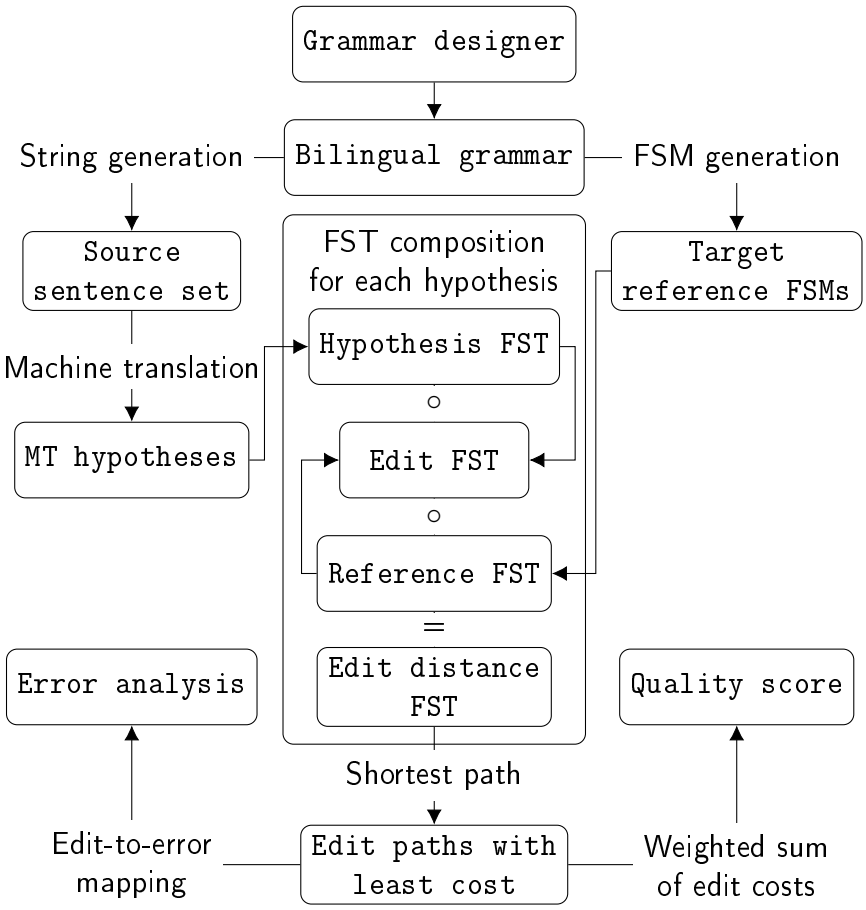
The main contribution of this thesis is a novel method of automated error analysis, which combines the automated multi-reference metric introduced in Dreyer and Marcu (2012) (from here on referred to as HyTER) with a

bilingual natural language generation (NLG) system, which generates English and Finnish expressions that are semantically near-equivalent. The NLG system is based on Grammatical Framework (Ranta, 2011), which provides ready-made morphogical and syntactic implementations of English and Finnish, as well as NLG and parsing functionalities. Error analysis is implemented by extending HyTER, which is based on a finite-state implemention of edit distance, with different edit types corresponding to different errors. The method introduced in this thesis, which is called EvalGen (from "evaluation" and "generation"), can be used to automatically gain insights into the translation quality aspects of an MT system.

In addition to error analysis, the method can be used as an automated MTE metric by calculating and normalizing a weighted sum of the errors, in the same way as in conventional edit distance based MTE metrics with edit types of varying weights, such as TER-Plus (Snover et al., 2009a). When used as a metric, the strength of EvalGen is that it offers a partial solution to the basic problem of reference-based metrics, the problem of reference sparsity. For all but the simplest sentences there is a vast amount of possible translations, which may be very different from each other. If only a single reference is used, as is generally the case with evaluations performed with automated metrics, many correct translations will be misjudged. The use of a generated set of reference translations, which covers a much larger portion of the space of possible translations than traditional methods with single references, makes the analysis more accurate. The set of source sentences generated for a particular semantic form is also large (although not as large as the reference set), so the transmission of the same meaning can be evaluated in several different forms, which increases the amount of data points and the reliability of the evaluation.

The motivation for this work is to provide MT developers more versatile insight into the workings of their MT systems. The black-box evaluation methods that have dominated MTE since the 1990s have undoubtedly been useful, but they may have also blocked progress in some directions by being biased against certain MT methodologies, such as rule-based machine translation (RBMT), syntactic variants of SMT, and NMT. The dramatic leap in performance with NMT, which is not fully reflected by the automated metrics, has underlined this issue. With further NMT development, similar biases that block promising research directions may appear. For instance, the problem of distinguishing fluency and adequacy in manual evaluations may introduce a bias towards fluent NMT systems. In order to detect such biases, it is useful to have a wide range of different evaluation methods, based on different paradigms.

Figure 1: The structure of EvalGen. FST stands for finite-state transducer, FSM for finite-state machine



As this is a master's thesis, the scope of the work is limited. The methods and software introduced here should be seen more of a proof of concept than a production-ready tool. While the software is fully functional and freely available, the analysis it provides may be too narrow to indicate MT quality reliably. However, it is still perfectly usable for gaining linguistic insights into the workings of MT systems. For instance, EvalGen makes it easy to verify that NMT systems make mistakes even with simple source sentences, which allows the sanity-checking of excessive claims of MT performance.

The thesis is divided into two main parts. Chapters 1 and 2 cover the history, theoretical background, and state of the art of MT and MTE. The rest of the chapters describe the design of EvalGen and its components (see figure 1 for

an overview of the structure of EvalGen). Grammar design and implementation is described in chapters 4 and 5. Chapter 6 discusses the finite-state methods used to calculate edit distance and their use for translation error analysis. Chapter 7 describes experiments that were performed to validate the methods used.

2 Machine translation

2.1 Brief history of MT

The history of MT starts a few years after the end of the Second World War, when the development of computers first made MT possible in the first place. Several people developed ideas related to MT in those early years, but the most significant milestone was the publication of Warren Weaver's memorandum Translation (Weaver, 1949), which was the impetus for the start of actual MT research. Despite some interest in statistical methods of translation, the early work in MT was almost exclusively based on handcrafted rules. This is understandable, since the lack of bilingual corpora and the limited computational capabilities of the early computers would have made any corpus-based approach to MT infeasible.

The history of MT is characterized by recurring cycles of enthusiasm and disillusionment. The initial boom in MT came to an end in 1966 with the publication of the ALPAC report (Pierce and Carroll, 1966), which thoroughly inspected the state of MT research at the time. The report justifiably came to the conclusion that MT research had not delivered on its promises, and that it was unlikely to do so until basic research in linguistics, computational linguistics and other fields had advanced sufficiently. The report therefore recommended that the resources assigned to MT work should be allocated to basic research and less ambitious practical development, such as translation aids for human translators.

By the start of the 80s, computer technology and research in computational linguistics had advanced considerably, and large-scale MT research was started again on a more solid theoretical foundation. Perhaps the most prominent project of the time was EUROTRA (Maegaard, 1989), an ambitious attempt to build a state-of-the-art MT system for translation between the then nine official languages of the European Community.

EUROTRA was a representative of what where later characterized as rationalist MT systems. The rationalists conceived of translation as modeling of

linguistic rules and ways of mapping linguistic structures between languages. While practically all MT had of course been based on rules since the 50s, the rationalist approach differed from the first, often pragmatically designed MT systems in placing more emphasis on the theoretical motivations of the rules. 80s also saw the emergence of empiricist MT, in which the systems learn to model the translation process automatically from multilingual corpora instead of using manually designed rules.

The large-scale rationalist projects of the 80s failed to provide the results that were anticipated, and disillusionment with the rationalist approach coincided with the initial peak of enthusiasm for the empiricist methods in the early 90s. By the 1992 TMI confererence in Canada (Whitecomb, 1992), the empiricist methods, and especially the new SMT methods pioneered by IBM, were clearly dominant.

However, it is important to note that the empiricist approach triumphed over the rationalist approach, not the rule-based approach as a whole: pragmatically motivated rule-based MT was still clearly superior to the early empiricist systems, as exemplified by the SYSTRAN MT system, which had been developed since the 70s. MT pioneer Yorick Wilks described SYSTRAN and its pragmaticist approach in Wilks (2009):

> While SYSTRAN translates to a reasonably high degree of proficiency, it has no underlying theory that a theoretical linguist would acknowledge as such ... the power of SYSTRAN lies in its well-established and relentless system of lexicon modification and augmentation in the face of bad translation results.

However, developing SYSTRAN-like systems was extremely work-intensive ("a lot of drudgery for years to come" in Wilks' words), and not particularly interesting from a research perspective. As a result, empiricist methods swiftly took over MT research from the 90s onwards, and rule-based methods were consigned to a marginal role in the MT reseach community (while still being central in commercial systems).

Around the start of the millennium, empiricist methods finally developed sufficiently (mainly due to the introduction of phrase-based SMT) to genuinely rival the pragmaticist RBMT systems. Even so, the RBMT systems that remained in development were often judged superior to SMT systems in evaluation campaigns, especially for language pairs with a morphologically complex target language. For instance, the PROMT rule-based system dominated the English-Russian category in WMT evaluation campaigns until the ascendancy of NMT a few years ago (Bojar et al., 2016).

Artificial neural networks (ANNs) had been around since the early years of MT, and there had been sporadic attempts to utilize them for MT. During the past decade, advances in using ANNs in other subfields of natural language processing (NLP) led to more consistent efforts at harnessing ANNs for MT. Progress was unexpectedly rapid, and within a few years NMT methods clearly surpassed SMT in translation quality.

Unlike in the switchover from rationalism to empiricism, where disillusionment over the failure of the previous approach paved the way for new, unproven methods, NMT took over from SMT purely on the strength of its results. SMT was still a very active and creative field of research, with new methods being constantly developed (although there were some signs of stagnation, as more complex models rarely managed to improve on the results of more primitive systems). Despite some reluctance to abandon their previous work, most of the researchers in the field very quickly moved from SMT to NMT. Currently, SMT has been relegated to a similar marginal role as RBMT, and NMT is the uncontested primary MT methodology.

The next sections introduce the MT paradigms mentioned above in more detail.

2.2 Rule-based MT

In rule-based MT (RBMT), the translation is generated by applying a set of more or less hand-crafted rules on the source sentence. Traditionally RBMT systems have been subdivided into three types: direct, transfer-based and interlingual. The types differ in the intermediate representation that is used during translation.

Direct RBMT does not use an intermediate representation at all, it directly converts the source language text into a target language text by using lexical and grammatical rules. Direct RBMT systems can be very simple (but also potentially very complex), so they were common in early MT research. The first show-cased MT system, the IBM-Georgetown system, is an example of a very rudimentary direct RBMT system (Hutchins, 2004). The main difficulty with direct RBMT systems is that wide-coverage systems need a very large amount of lexical and grammatical rules. Rule conflicts are inevitable, and the ruleset can easily become unmanageable. On the other hand, the absence of an intermediate representation removes one potential source of errors, namely parsing problems when converting the source text to a more abstract representation.

In transfer-based RBMT the translation is produced by first parsing the source text into a source language structure and then using rules to convert the source language structure to a target language structure. The structures used in transfer-based RBMT are usually specifically designed for the task of translating between the languages in question, so they do not need to be exhaustive analyses of all the linguistical aspects of the source text. It is enough that they describe the phenomena that significantly affect the translation between source and target languages. For instance, when translating from English to Finnish, the analysis of English present tense verbs does not have to include information on whether the verb refers to an event that is currently happening or that will happen later: both languages use present tense for both current and future events. If the target language has a tense that is strictly used with present events, this distinction would have to be captured in the analysis. Currently most transfer-based RBMT work is done with the Apertium open-source toolkit (Carme Armentano-Oller et al., 2005).

Interlingual RBMT involves a deeper intermediate representation, which attempts to capture the language-independent meaning of the source text. This representation is usually called an interlingua. What makes interlinguas especially attractive is that once a parsing component is created for a language, the language can theoretically be translated into any language for which an generation component exists. In contrast, direct and transfer-based RBMT usually require a separate system for each language pair. A wide-coverage interlingual RBMT system would require a comprehensive semantic language, which would probably be impossible to design, so interlingual systems are usually intended for constrained domains. The parsing task is also a lot more complex than for transfer-based systems, and errors in parsing have more effect on the quality of the translation. To alleviate these problems, an interlingual system may constrain the input, for instance by using a controlled language (for an example, see Mitamura (1999)).

2.3 Corpus-based MT

In the late 1980s, researchers started experimenting with corpus-based MT methods. The main distinguishing feature of these methods is that the core functionalities of the engines are learned or inferred from a bilingual corpus. The early corpus-based methods are generally divided into two categories: example-based MT (EBMT) and statistical MT (SMT). Both of these categories include a large variety of different approaches, and the categories also overlap in some respects. What separates the two is a different research tra-

dition (the starts of the both approaches can be traced to single articles) and a different conception of what translation is. During the last few decades, SMT became the dominant branch of research, and the work springing from the EBMT tradition was subsumed within the more general SMT approach.

2.3.1 Example-based MT

The basic concept of EBMT was introduced by Makoto Nagao in the early 80s (Nagao, 1984). At that time, the state-of-the-art MT systems were transfer-based, and they did not work well when translating between languages with large structural differences, such as English and Japanese. Nagao suggested that the quality problems were partially due to a faulty theory of translation that was implicit in transfer-based MT: the idea that deep linguistic analysis was necessary for translation. According to Nagao, no complex transfers were necessary, translation could be done just by using a database of examples and a method of adapting and combining those examples for new source texts.

Nagao's article was published years before actual work on EBMT started, and it only contained a very rough outline of a MT system. The systems the article later inspired took many different forms, and defining EBMT in a way that includes all the different EBMT systems (but excludes RBMT and SMT systems) is difficult (Hutchins, 2005). Despite the lack of a clear definition, it is commonly agreed that the key idea of EBMT is translation by analogy: the SL sentence is decomposed into fragments, which are matched with examples in the database, and the translation is produced by adapting and recombining matching TL fragments from the database.

The main advantage of EBMT is that it can produce very high-quality translations in the right circumstances. The quality can be comparable to matches from a translation memory (TM) used in computer-assisted translation tools (CAT). As a technology, TMs have many similarities with EBMT, such as the retrieval of similar translation examples. As with TMs, EBMT's usefulness is heavily dependent on whether the example database contains relevant examples. The size of the example database also needs to be restricted, since the search can be computationally expensive, especially if the matching uses complex methods (and generally the methods are more complex than the simple edit distance used by TMs). To have wider coverage, EBMT must produce translations by recombining smaller fragments from the database. However, as the size of the fragments decreases, the two main challenges of EBMT, alignment and recombination, become much more difficult.

Alignment is the process of connecting fragments of the SL text to their

translations in the TL text. At its simplest, alignment can be performed just by using a bilingual dictionary. However, the dictionary method works only in cases where the connection between the SL and TL fragments is obvious. For more complex alignments, such as word to phrase alignment, more sophisticated and less reliable methods are required.

Recombination is the process of combining translation fragments retrieved from the database. The challenge in recombination is that words are dependent on other words, which partly determine their surface form and place in the sentence. To produce grammatically correct sentences, a recombination method must be able to pick fragments that are compatible with each other. For instance, in many languages the object argument of a verb frame has a specific morphological form, which must be used when combining a verb frame with its case arguments. Morphological, syntactic and semantic errors caused by the recombination of incompatible fragments are called boundary friction in the EBMT literature (see for example Somers (1999)).

Even though EBMT research slowed down with the rising popularity of SMT, EBMT-like features have recently been incorporated into many CAT tools, such as Déjà vu, MemoQ and CafeTran. These features are part of a broader trend of including subsegmental matching into TM systems (see Flanagan (2015) for a typology of subsegment features in CAT tools).

2.3.2 Statistical MT

The first article on SMT (Brown et al., 1988) appeared roughly at the same time as the first EBMT projects. The motivation for SMT was the success of statistical methods in automatic speech recognition (ASR) in the preceding years. While the early SMT systems were crude and used what seemed like a very implausible method of translation, the results were comparable to the best RBMT and EBMT available at the time (White, 1994b). It should be noted that the success was partly due to a language direction (French to English) that was specifically selected to make the problem of SMT easier to solve (Jelinek, 2009).

As with EBMT, there are many varieties of SMT. The main subtypes are introduced in the following sections.

2.3.3 Word-based SMT

The first SMT systems are called word-based, because they subdivided the translation task into the tasks of translating each source word separately

and then combining the translations. Most word-based SMT models are refinements of the original IBM models developed in IBM's Candide project (Berger et al., 1994). Despite the inherent weaknesses of the word-based approach, it is important for historical reasons and for the major role it played in other, more advanced approaches.

The IBM models were directly based on the noisy channel model that IBM used in ASR:

$$\hat{E} = \arg\max_{e} P(F|E)P(E) \tag{1}$$

In this model, the translation ($\hat{E}$) for input string F is the string E that has the largest product of P(F|E) and P(E). P(F|E) is the probability that the TL string E (short for English, but used as shorthand for all TLs in IBM models) being evaluated is translated as the SL string F (short for French, but used for all SLs). Notice that the translation direction is reversed in the formula, since the noisy channel model makes the assumption that the SL string is a garbled version of the TL string that needs to be corrected, in the same way that the acoustic signal is assumed to be garbled form of the speaker's intended utterance in ASR. P(E) is the probability of the TL string occurring in the TL.

The formula of the noisy channel model makes explicit the two main components of most SMT systems. P(F|E) represents what is called a translation model in SMT, while P(E) represents a language model, a standard tool of statistical natural language processing.

The translation model is the primary component, it estimates how good a translation TL string E is for SL string F. In practise, the translation model is not able pick the best translation out of the available candidates. The reason is familiar from EBMT: boundary friction. It is especially bad with word-based models, since they compose the translation out of word-sized bits.

To overcome boundary friction, SMT systems use a language model, which estimates the fluency of TL strings. In statistical NLP, language models are usually n-gram models (see Chen and Goodman (1998) for overview), which assign probabilities to strings based on how often they occur in the training corpus. However, any method of assigning probabilities to possible TL strings can be used as a language model.

The main problem in training translation models is familiar from EBMT, where it was called the alignment problem: how can we identify which parts

of the SL and TL strings constitute source-translation pairs, and how can the identification be as fine-grained as possible (to increase coverage)? In word-based SMT, the granularity of the alignment is by necessity word-based, and the task is aligning each source word to zero or more target words. In IBM models, the word alignments are learned by using the expectation maximization (EM) algorithm, which is used for machine learning tasks in many fields.

Recombination in SMT (called decoding) is a difficult problem because of two factors: word order selection and overlapping translations in the translation model. These issues have been shown to make even the simplest form of SMT decoding intractable (Knight, 1999). Because of this, heuristic search methods must be used in decoding for all but shortest of source sentences. IBM's original system used a variant of a stack decoding algorithm that was also used in IBM's ASR work (Berger et al., 1996). Many other decoding methods were proposed for word-based SMT, and beam search emerged as the most popular option.

2.3.4 Phrase-based SMT

Word-based SMT was initially a compromise, a deliberately naive approach chosen for its simplicity, which made the early development of SMT easier. In fact, the first article on SMT (Brown et al., 1988) proposes to expand the word-to-word method to handle longer "fixed locutions", such as "one which reflects on"/"pour mettre en doute". The motivation of this expansion is clear: by translating larger chunks of text at a time, boundary friction is alleviated, as phrase translations can be guaranteed to be internally grammatical and correct at least in some context.

In the late 90s articles were published on various implementations of phrase-based SMT, but most of the phrase-based SMT research can be traced back to alignment template models first proposed in Och and Weber (1998). The key innovation in alignment template models is the phrase-extract algorithm. The algorithm is used to learn bilingual phrases from a word alignment matrix, which can be fairly straight-forwardly generated from the word alignments of IBM models. Word alignments are generated for both possible translation directions, and the two alignments are then symmetrized to create the word aligment matrix. The two-way generation and symmetrization is required, because in word-based alignments each target word can only be aligned with exactly one source word, which causes problems with e.g. compound words.

The alignment template models introduced the phrase-extract that is basis of modern phrase-based SMT, but they differ significantly from later PBSMT systems. In alignment template models the phrase translations (called templates) are mappings from sequences of source word classes to target word classes. The word classes are learned using unsupervised methods on the bilingual corpus, and their purpose is to allow the use of the templates in wider contexts than the exact lexical context from which they are learned from. The generalization provided by the word classes is especially important, when the training corpus is relatively small, which was the case for the corpora used to test the alignment template models.

Another innovation introduced in alignment template models was the replacement of the hidden channel method of the IBM models with a log-linear model. The main advantage of the log-linear model is that features can be easily added to the model to complement the translation and language models. The weights of the features are tuned using a dataset that is not included in the training set.

Och (2002) found alignment template models with word classes superior to models where purely lexical phrases are used. However, lexical phrases eventually emerged as a standard in phrase-based SMT. Koehn et al. (2003) described the simplified approach that became dominant in the field, especially with the introduction of the popular Moses SMT toolkit. The main finding of the article was that a relatively simple and easily implementable model based on the phrase-extract algorithm and lexical phrases provided a very strong baseline that outperformed more complex methods. The baseline implementation of this original method remained competitive right to the end of the SMT era.

Pure PBSMT models are not well-suited for translation into morphologically complex languages due to boundary friction. Several approaches have been developed to solve this problem. Segmentation of the TL may be used to make it possible to align SL tokens with individual morphemes of the TL (Virpioja et al., 2007). SL and TL tokens may also be factored into several different aspects, which can be individually processed (Koehn and Hoang, 2007). For instance, the SL surface form may be first translated into a TL surface form, from which a morphological tag is also generated. This allows the use of morphological sequence models, which can accommodate much longer contexts than language models. It is also possible to use a separate tool to post-process the SMT output to increase fluency (Rosa et al., 2012).

2.3.5 Tree-based SMT

The striking thing about word- and phrase-based SMT is that they completely ignore compositional aspects, which many consider essential to natural language. Language consists of expressions, which can be combined with other expressions to form new expressions, although views on the nature of the expressions and the ways of combining them differ. The words and phrases in WBSMT and PBSMT generally do not correspond to any conception of linguistic expressions, and the only method of combination used is simple concatenation.

Alternative SMT methods that allowed for more complex structure were introduced already in the nineties, a few years after the introduction of IBM models. Alshawi et al. developed an SMT system that learned a set of weighted dependency transduction rules from an aligned bilingual corpus, and then created the translation by using the rules to create the target dependency tree that was most probably in the model defined by the rules (Alshawi et al., 2000). Wu replaced the translation model in a word-based noisy channel model with a translation model that allows only word alignments that can be generated with a simple bilingual grammar (Wu, 1996).

There are many different approaches to tree-based SMT, with varying parsing methods and tree formalisms. One popular way of categorizing tree-based SMT systems is to classify them according to whether they make use of linguistically motivated annotation for SL or TL or for both.

Systems that use no linguistic annotation at all are called hierarchical (Chiang, 2005), while systems with linguistic annotations are called syntactic. The rules for hierarchical systems can contain only lexical items and a single type of generic non-terminal. The main motivation for using hierarchical models is to capture a wider context than is possible with PBSMT. The non-terminals in the translation rules of hierarchical systems should make them better at handling medium- and long-range dependencies in source texts, which are not present in the training corpus. Coincidentally hierarchical systems should also be better at reordering, since the non-terminals can be reordered in the translation.

Models where only SL non-terminals contain linguistic annotations are called tree-to-string models. The motivation for these models is that syntactic or semantic parses of the source sentence can be used as a source of information in the translation process. For instance, word class information may be used for word disambiguation, or syntactic role information can be used to trigger phrase reordering in the TL. In tree-to-string models the decoding is also

considerably simpler, as it is constrained by the SL parse.

Correspondingly, models where only the TL non-terminals are annotated are called string-to-tree models. The function of the annotations is different from the one in tree-to-string models: the TL annotations are used to ensure that TL fragments are combined grammatically, i.e. to reduce boundary friction. Their function is similar to that of language models.

In tree-to-tree models, the SL and TL annotations are used in tandem, which potentially combines the benefits of both approaches. However, all syntactic models have a problem with coverage, and its generally more severe with tree-to-tree models (Ambati and Lavie, 2008). For this reason, the constraints of syntactic models are often relaxed to increase coverage (Hoang and Koehn, 2010).

While syntactic SMT appears on the surface to have much greater capability than PBSMT, the results achieved with it were roughly on par with those of PBSMT. It should be remembered, though, that PBSMT was a much more popular method, and that syntactic SMT might have surpassed it with further research. One considerable obstacle in the way of syntactic SMT may have been that automatic MTE metrics were biased against it (Bojar et al., 2015).

2.4 Neural MT

Artificial neural networks (ANNs) have been used in machine translation since the 90s, and the first pure NMT system was described in Castaño et al. (1997). This system was trained for experimental purposes on extremely constrained generated data. Training a production system on the same principles would have been impossible due to the computational demands of training the network, so the work was not pursued further.

In the 2000s, ANNs showed good results in other NLP domains, and slowly started to be adopted for MT work. A pioneer in this field was Holger Schwenk, who demonstrated that ANNs could be utilized as language and translation models in SMT systems (Schwenk, 2012). ANNs were also used to rerank translations produced by SMT systems (Kalchbrenner and Blunsom, 2013). MT systems based entirely on ANNs, which can properly be called NMT, appeared in 2014 (Sutskever et al., 2014).

Most current NMT systems use an encoder-decoder architecture (Cho et al., 2014). In an encoder-decoder system, an encoder component first encodes the source sentence into a vector representation, which is then converted into

a probability distribution over TL words by a decoder component. A search algorithm (usually beam search) is then used to generate the likeliest TL sentence. The encoder and decoder components are trained jointly.

The first encoder-decoder systems had a serious deficiency: they encoded the SL sentence into a fixed-length vector for any length of input. Predictably performance was good with short sentences but considerably worse with longer sentences, whose meaning could not be compressed into the fixed-width vector. Bahdanau et al. (2014) enhanced the encoder-decoder model with an attention mechanism, which allows the decoder to utilize information about the SL tokens during decoding instead of relying on a fixed-width encoding.

These attention-based encoder-decoder models were the first ones to consistently demonstrate better performance than SMT systems. Initially the performance of the models was hindered by the limited size of the TL vocabulary, which is inherent to NMT, but efficient segmentation methods were soon developed (Sennrich et al., 2015a). Another important development was the use of backtranslation, i.e. translating TL monolingual data with another MT system and using the translations and monolingual data as supplementary training data (Sennrich et al., 2015b).

In the WMT16 news translation task (Bojar et al., 2016), NMT systems were clearly superior for many language pairs, and by WMT17 (Bojar et al., 2017), nearly all entries were attention-based encoder-decoder models, which decisively overperformed the few remaining SMT entries.

Vaswani et al. (2017) introduced the transformer model, which further improved NMT performance by abandoning the recursive neural networks (RNN) used in previous attention-based models and relying entirely on the attention mechanism. This allows easier parallelization of training, which makes system creation faster, and also improves translation quality. Currently most competitive NMT systems are based on transformer models.

Although NMT is corpus-based like EBMT and SMT, it has one decisive difference when compared the other two approaches: the core of NMT models is not a database of aligned SL and TL segments and structures, but an ANN that has been optimized to produce output that resembles the examples in the training data. As the size of the model is limited, the ANN cannot learn anything similar to the massive databases of SMT but has to learn generalizations and structures, which approximate components of natural language, such as morphology and syntax. One consequence of NMT not storing its models as aligned segments is that there is very little boundary friction in NMT output.

3 MT evaluation

Evaluation of MT output is one of the most important and challenging aspects of MT development. Evaluation has several roles and forms in MT development, and some MT approaches are more dependent on evaluation than others. At the bare minimum, every MT system is periodically tested in some way during development and eventually evaluated on how it fulfills the goals that have been set for it.

Historically, RBMT systems could be developed without systematic evaluation, since the development was guided by the theoretical underpinnings of the MT system. Ad-hoc evaluation of the output was still probably always used for debugging rules and as a source of inspiration for improving the ruleset. When RBMT systems were systematically evaluated during development, it was usually done by using glass-box evaluation, where components of the system (such as the parser or the transfer component) were tested individually to ensure that they operated according to specifications.

For modern MT systems based on machine learning, automatic evaluation metrics are used during training to directly optimize parameters of the model or to pick a stopping point that minimizes overfitting. This kind of evaluation is almost always of the black box variety, i.e. the system is evaluated as a whole based on what output it provides for a given input, with no reference to the system's internal structure or functioning.

3.1 History of MT evaluation

Translations in general are difficult to evaluate. This is caused partly by the inherent complexity of the task and partly by the high cost of human resources which many forms of evaluation require. Many of the problems that affect the evaluation of original text also affect evaluation of translations: fluent users of the standard variants of languages often disagree even on what is grammatical, and different preferences regarding structure and style cause more disagreement. For translations, a further source of disagreement is how and to what extent the translation should reflect the literal meaning and form of the source text.

3.1.1 Traditional translation evaluation

Traditionally, evaluation of translations has been performed by proofreaders and teachers as part of editing and language instruction. This traditional form of evaluation is based on informal and highly subjective methods, and relies on the evaluator's intuition and conception of the rules of the standard language in question. In any case, the function of this sort of evaluation is not to produce a consistent measurement of translation quality, but only to provide pedagogical feedback to the translator and to enforce quality standards (however they may be defined). Translation tests are another traditional method of evaluation, and they have been used to rate translator competency or language skill in general since the 19th century or even earlier (Madras et al., 1866, p. 485). While these tests should ideally be consistent and objective, they still heavily rely on the grader's intuition and preferences.

Traditional translation tests also measure a very narrow part of the quality scale, and the tests are generally designed only to identify candidates who can produce almost error-free translations. For instance, in the translation tests administered by the European Personnel Selection Office, marks are deducted for omissions, difficulties in conveying the sense of the passage, lacking the vocabulary required by the passage for translation, incomplete translation, and mistakes in meaning, grammar, terminology and register (EPSO, 2018).

This kind of testing is only informative and feasible, if the amount of errors remains relatively low. As the number of errors increases, grading becomes more and more cumbersome and uninformative, since there's no clear way of categorizing the competence of a translator who makes tens or hundreds of mistakes, except to say that the translator is not good enough. Also, when there are numerous errors in a sentence, as is often the case with MT, the interrelations between different errors make it difficult to judge the overall amount of errors accurately. For instance, if a translated sentence is completely ungrammatical, the grader would in theory need to convert the translation into a grammatical version and track in some consistent way all the changes that are required to achieve that. It would extremely difficult to create consistent guidelines for performing this sort of error tracking manually. In practise, the grader would not bother to even try, since fair scoring of low quality translations is irrelevant in selection tests.

The unsuitability of traditional translation tests for MTE was confirmed, when a standard US government metric for grading translations was tried as part of the ARPA MT program evaluation:

> . . . it was not possible to maintain the exact structure of the metric: the nature and proliferation of MT errors necessitated alterations of the grading limits in the original method, thus introducing the potential for the very bias that its use was intended to avoid. (White et al., 1994)

So while there were existing methods for translation evaluation at the time when the need for MT evaluation arose, they were neither clearly formulated nor suitable for the task. This is also apparent from the most influential early article on machine translation evaluation Carroll (1966), which refers to only two prior articles on translation evaluation, both of which concern MT evaluation specifically. Carroll (1966) was part of the work of ALPAC, which was a multidisciplinary committee and interviewed experts on human translation, so this lack of references to earlier work on evaluation of human translation was probably not due to omission, but simply because relevant work had not been done previously. Carroll seems to suggest this in the article:

> It would be desirable, in studies of the merits of machine translation attempts, to have available a relatively simple yet accurate and valid technique for scaling the quality of translations. It has also become apparent that such a technique would be useful in assessing human translations.

3.1.2 Evaluation within translation studies

The lack of an existing theoretical framework for translation evaluation can be explained by the fact that intense, specialized academic study of translation began roughly contemporaneously with the start of MT research. In fact, some early pioneers of translation studies were evidently influenced by considerations related to MT research (Chesterman, 2012). Translation research was eventually consolidated under the discipline of translation studies, which James S. Holmes attempted to define in his seminal paper The Name and Nature of Translation Studies (Holmes, 1975). In Holmes's classification, translation evaluation occupies a place in the second main branch of the discipline (applied translation studies), under the broader category of translation criticism. Holmes's brief comments on this subdiscipline seem to confirm that translation evaluation was not a mature field:

> The level of such criticism is still today frequently very low, and

> in many countries still quite uninfluenced by developments within the field of translation studies.

Since then, a large body of academic work on translation evaluation has been produced within translation studies. While scientific models of evaluation have been criticised as difficult to apply in practice (Lauscher, 2000), translation evaluation systems used in the translation industry and universities have also been actively investigated (cf. Secară, Zehnalová (2013)). Both the theoretical and practical approaches to evaluation within translation studies are mainly concerned with human translation, which makes them impractical to use in MT evaluation. This is due to the fact that MT quality is generally still so low that many of the important aspects of human translation evaluation, such as fitness for audience and correct cultural adaptation of source texts, are simply not relevant. However, as the quality of MT improves and implementing these aspects becomes feasible, the issues discussed in evaluation research within translation studies must eventually be taken into account also in MT evaluation.

3.1.3 Evaluation as part of MT research

For the reasons mentioned above, evaluation methods suitable for MT have had to be developed within the MT research community, in parallel with the actual MT development. The first paper on MT evaluation was published in 1958 (Miller and Beebe-Center, 1958). Even though MT development had started only a few years earlier, and the paper had to rely on simulated MT output, it correctly identified many of the crucial concepts of MT evaluation, including the use of subjective assessment by bilingual readers as a reference metric, distinction between measuring general quality and fitness for a purpose, and comparing translations to a reference translation with objective measures. After this pioneering work, evaluation methods were studied and refined gradually, often in connection with large-scale MT initiatives, such as Eurotra and DARPA programs. Another driving factor behind MT evaluation development has been the arrival of new types of corpus-based MT. The rise of SMT led to many new developments in evaluation, and NMT has already stimulated new work that approaches evaluation from a different viewpoint.

3.2 Task-based MT evaluation

One of the main distinctions in MT evaluation is the separation between intrinsic methods, which attempt to measure general quality of translations independent of any actual use case, and extrinsic or task-based methods. Much work has been done over the decades on both of these branches of evaluation, and the literature on different methods is too extensive to be covered completely. Since the focus of this thesis is on general quality evaluation, task-based evaluation will covered briefly in this part of the chapter, and the rest of the chapter will cover general quality evaluation in more detail.

In general, translations are produced for some purpose, such as to allow people to use a product manual or read a work of literature even if they do not understand the language of the original text. The key point is that the characteristics of the optimal translation depend on the purpose of the translation: a translation of a product manual must mostly preserve the meaning and form of the original, but its aesthetic qualities are irrelevant, while the opposite holds for a work of literature. Therefore it would seem reasonable to evaluate translations based how well they fulfill their intended purpose.

Task-based evaluation was explicitly recognized as a separate type of evaluation during the nineties, but the basic concept was identified and implemented earlier in many forms. Miller and Beebe-Center (1958) already contained this observation:

> The purpose [of translation], of course, is communication. The translation should be judged successful if this purpose is achieved. . . . One way to apply this technique is in the form of commands that must be carried out by some gross, bodily behavior. A more convenient way is to ask questions that can be answered verbally.

One early example of task-based evaluation is Sinaiko and Brislin (1973), where translation quality is assessed by counting how many errors helicopter technicians made when performing maintenance tasks based on translated instructions. This test indicated that performance is sensitive to translation quality, even though all assessed translations were produced by professional translators (the test did not include MT, even though Sinaiko had previously tested the Logos MT system with similar helicopter maintenance instructions). Another interesting finding was that the test subjects' opinions of

the usefulness of different translations were opposite to the measured performance levels, which shows the importance of task-based evaluation as a tool for validating other evaluation methods.

Since most of the tasks that task-based metrics can be based on are generally performed by humans, task-based evaluation usually requires the assistance of human evaluators. However, automatic task-based metrics can be devised, if MT is part of a larger automated NLP workflow (Popescu-Belis, 2007) where the output of the MT is used as input to a component for which better evaluation metrics are available. For instance in Babych and Hartley (2004), MT is used as input for named entity recognition software and evaluated on the performance on the named entity recognition task.

3.2.1 MT-specific applications and task tolerance

For MT there are more applications than for human translation, since it is usually considerably faster and cheaper. This means that even if MT quality is very low by general human standards, it may be useful in some capacity which cannot be feasible performed by a human (Church and Hovy, 1993). For instance, it would not be possible to have a human translator produce instant, low-quality translations of web pages to get an overview of their contents, but MT can perform that task, and has done it very successfully for the past decade or so. In other words, each task has a certain threshold of translation quality, task tolerance, which an MT system must meet to be useful in that task (White and B. Taylor, 1998), and task tolerances may be much lower for some tasks than for traditional translation.

Task-based metrics are especially important for these MT-specific applications, since the metrics of general MT quality do not necessarily correlate with the usefulness of a MT system in these tasks. For instance, Tatsumi (2009) investigated the correlations between post-editing speed and various automatic MTE metrics, and found that several factors such as sentence length affected how well the metrics correlated with the post-editing speed.

One of the benefits of task-based evaluation is that it provides meaningful results, which researchers and users of MT can easily interpret. While setting up one-off task-based tests may be expensive, task-based evaluation data can in some cases be collected unobstrusively in production applications (John Moran, 2014), which means that data can be accumulated continuously and cost-free except for the initial investment.

3.3 Intrinsic MT evaluation

While task-based methods try to assess the suitability of an MT system for a particular task, intrinsic methods attempt to measure a much more abstract and harder to define feature of MT, general translation quality. As was discussed, translations are generally made for a purpose, and their quality is, in a practical sense, always ultimately judged on how well they fulfill that purpose.

There are also other context-sensitive aspects to translation quality. Translations are usually part of a larger collection of text, and their quality is partially judged on how consistent they are with other parts of the same collection: sentences are parts of paragraphs, paragraphs of documents, documents of document collections, and so on. There is often an expectation of terminological, phraseological, structural, and stylistic consistency betweeen all of the parts.

Languages also differ on what types of information are obligatory, and if the SL omits information that is obligatory in the TL, the obligatory information will have to be inferred from the context. This is very often the case in translation from Finnish to English because of the structural difference between the languages. For instance, Finnish only has a gender-neutral third-person pronoun, while English generally requires the use of gendered third-person pronouns for referents of determinate gender.

Even though translation quality is inherently context-sensitive, it is certainly possible to assess the quality of even a single sentence translation which is disconnected from any larger context. In this scenario, the evaluator considers whether there exists any feasible context in which the translation is valid, rather than considering whether the translation is valid in a specific context, as would be the case in normal evaluation. The evaluator will still use their real-world knowledge to exclude certain contexts as unfeasible, which means that quality estimates will inevitably vary between evaluators. Some correct translations may also seem incorrect out of context: the translation may be non-literal or it may have been necessary to add information to a translation to clarify the meaning.

The Finnish WMT18 reference translations contain many examples of translations that appear incorrect because context information has been added to the translation:

Source	It is the first such warning that the organization has issued for a state in the US.
Reference	**Missouria koskeva matkustustiedote** on järjestön ensimmäinen Yhdysvaltojen osavaltiota koskeva varoitus.

The text in bold contains information that is not included in the source. The translator has translated the word "it" with its referent, which is only available in the larger context of the document. Judged independently, many evaluators would consider this an incorrect translation, even though it is perfectly good in the context it was created in.

Despite these concerns, intrinsic MT evaluation is almost exclusively performed on out-of-context single sentence translations (Gimenez et al., 2010). The reasons are practical: evaluation based on larger contexts would be considerably more expensive and complex, and there would not be much point to it: current MT systems are overwhelmingly designed for translation of independent sentences, even though interest in discourse-level MT has rekindled in recent years (Webber et al., 2015). It is an open question how much ignoring context affects the validity of MT evaluation, although the effect probably is not great at the current state of MT technology. Evaluation of independent sentences is a justifiable simplification for now, although this will change when MT methods mature further. Since task-based evaluation does measure quality in a manner that includes context effects, it can eventually be used assess the validity of evaluating independent sentences.

3.4 Manual evaluation

For most of the history of MT, intrinsic quality evaluation has relied exclusively on human judges, who evaluate translations based on their own linguistic competence. Humans are ultimately the arbiters of translation quality, so it seems reasonable to base MT evaluation on their opinions, even if they are subjective.

3.4.1 Early work

The main problem with human evaluation is providing the judges evaluation metrics that are easy to interpret, consistent across judges and evaluations, and sensitive enough. Miller and Beebe-Center (1958) tested a quality scale from 0 to 100 that was specifically designed to be unidimensional so that it could provide 'a single figure of merit' for a translation. The results were

not promising, since the judges made "bitter comments about the difficulty of their task", and the variance of ratings was large. The unidimensional translation quality metric also appears have been difficult to grasp for the judges, as grammatical correctness seemed to have had a much larger impact on their ratings than correct information content.

Carroll (1966) made a distinction that has become central in manual translation evaluation. The article identified two separate dimensions, which it deemed essential to representing translation quality: intelligibility and fidelity or accuracy. Intelligibility of a translation means the degree to which the translation resembles 'normal, well-edited prose'. Fidelity or accuracy means the degree to which the translation conveys the core meaning of the source sentence. 'Core meaning' here refers to the meaning of the source sentence without any context, and it allows justified omission or addition of information in the translation.

The results of experiments in Carroll (1966) confirmed that separation of translation quality into two dimensions is justified: inconsistency between intelligibility and fidelity ratings[1] for the same sentences usually indicated inaccuracy or disfluency in translation, which might have not been adequately captured by an unidimensional scale. The rating variance was also smaller on the intelligibility scale, which indicates that intelligibility is easier to rate than fidelity. This may explain why intelligibility appeared overrepresented in the results of Miller's experiment with an unidimensional scale. Somewhat surprisingly, intelligibility and fidelity were found to correlate strongly, and Carroll even suggests that evaluating machine translations solely on the intelligibility scale might be sufficient for most purposes.

Intelligibility and fidelity or accuracy formed a part of a larger toolkit of MT evaluation methods, which accumulated over the initial decades of MT research, as Slype (1979) summarized in a survey of the state of the art of evaluation in 1979. The survey, which was produced as part of preparations for the EUROTRA project, recognizes intelligibility and fidelity as some of the most important evaluation criteria and lists several different methods of measuring them.

3.4.2 DARPA: fluency and adequacy

The DARPA MT initiative, which started in 1992 and ran for four years, permanently established intelligibility and fidelity as the most important MT

[1] Note that fidelity was measured indirectly by rating the additional information gained from source sentences after first reading the translation, a measure called informativeness.

evaluation metrics. The initiative sponsored research on three MT systems, which all used different approaches to MT: the first SMT system Candide, interlingua-based Pangloss, and the hybrid system Lingstat. The heterogeneity of the systems necessitated a black-box approach to evaluation rather than the glass-box approach that was more common at the time. John White, who led the evaluation portion of the DARPA initiative, later recounted his conversion from a glass-box approach to a black-box approach during the initiative:

> Until the ARPA MT Evaluation took shape in the last couple of years, I would have considered any attempt to evaluate machine translation to be purely driven by the design of the system or by its end-application. . . . The profoundly black-box evaluation approach in the ARPA initiative is necessitated by the well known differences among the systems being evaluated. . . . I have come to believe that the functional perspective driven by black-box sets a clearer path for improvement in terms of the ultimate use of MT systems. (White, 1994a)

It appears that the DARPA evaluation methods were developed entirely from ground up with no reference to earlier evaluation work. The initial methods were a comprehension test and a quality panel (White, 1994b). For the comprehension test, newspaper articles originally written in English were first translated by professional translators into the source languages of the systems, and then translated back to English with the MT systems under evaluation. These round-trip translations were then read by English speakers, who answered questions about the content of the articles. In the quality panel method, professional translators graded the MT based on a standard US goverment metric for human translator competence.

Neither of the initial methods worked well: the comprehension test failed, because it was difficult to differentiate the effect of the human translation from that of the MT, while the quality panel grading was logistically very demanding and the quality of the MT output was too low to be graded with a metric designed for human translators. In the subsequent DARPA evaluations, these methods were superseded by three new metrics: the two primary metrics fluency and adequacy, and the supplementary informativeness metric[2]. Although these metrics were apparently developed independently in the

[2]This is not the same metric as the informativeness metric in Carroll (1966), but a modified version of the DARPA comprehension test.

DARPA initiative, fluency and adequacy are in practice identical to intelligibility and fidelity, and a metric similar to informativeness was first proposed in Miller and Beebe-Center (1958) and tested in Pfafflin (1965).

It may seem surpising that a large-scale MT effort like the DARPA initiative did not utilize prior evaluation research, and instead chose to initially develop methods which seem needlessly complex and even somewhat naive compared to the work performed in the earlier decades, and then unknowingly replicated almost exactly the methods developed earlier. One of the reasons for this was probably the prominence of glass-box evaluation during the time when the initiative started, which is apparent in the account by John White that was quoted above. Further explanation is provided in King et al. (2003), which outlined a proposal to consolidate prior research related to MT evaluation under the FEMTI project:

> . . .there are features of MT evaluation which make evaluators feel that each evaluation is special, tempting them to design the evaluation from scratch each time. An immediate consequence of course is that work is wasted: a deal of literature is generated, but since it starts from different presuppositions and from different pre-conditions, its utility is not immediately obvious. Furthermore, much of this literature is not easily available . . . The evaluator charged with designing a new evaluation thus quite naturally feels that he has neither the time nor the inclination to carry out a systematic search of the literature . . .

White (1994b), which describes the evaluation methods of the original DARPA MT initiative, became the standard reference on intrinsic manual evaluation. The use of fluency and adequacy as primary manual evaluation metrics was continued in DARPA's later TIDES and GALE projects, and the NIST OpenMT evaluation campaign related to those projects (Przybocki et al., 2008; NIST, 2002). The terms "fluency" and "adequacy" were also used in the extremely influential Papineni et al. (2002), which further established the terminology of the original DARPA MT initiative. Fluency and adequacy were later chosen as the manual metrics in the ACL (Koehn and Monz, 2006) and IWSLT (Akiba et al., 2004) evaluation campaigns, which are still regularly organized.

After Callison-Burch et al. (2007) found that inter-annotator agreement was only fair for fluency and adequacy, they were mostly superseded as metrics in the ACL and IWSLT evaluation campaigns by a ranking metric. However, Graham et al. (2013) demonstrated that inter-annotator agreement could be

improved by using continuous scales, and adequacy evaluation returned to ACL evaluations in 2016 and 2017.

3.5 Automatic evaluation

While human evaluation is ultimately the only way to verify translation quality, it is expensive, difficult to organize, and often not consistent. For purposes of MT development, it is desirable to have access to an evaluation method that can be applied inexpensively and quickly whenever necessary. This is only possible if the method is automatic or if the human contribution to the method can be performed separately and reused. Completely automatic evaluation methods are problematic, since in order to be accurate, they would have to solve the same problems that MT itself attempts to solve. While recognizing good translations automatically is a slightly easier problem than generating them, it is still beyond what is currently possible. That is why practically all of the current metrics that are called automatic rely on some sort of human contribution[3].

3.5.1 History of automatic MTE

An automatic metric was first proposed and tentatively tested in Miller and Beebe-Center (1958). The metric was based on comparing a candidate translation to a previously produced translation that is known to be correct (called a criterion translation in the article, and now usually called a reference translation). This is still the most common type of automatic metric, since it is conceptually straight-forward and reference translations are easily available. The metric proposed in the article would in modern terms be described as a recall metric based on a bag-of-words model, i.e. it measures the proportion of the words in the reference translation that are also found in the candidate translation, without considering word order. Bigram and trigram variations of this metric were also tested, along with a variant where the matched words were also scored for being in the same order as in the reference translation.

The article also recognized the obvious problem with the use of reference translations, i.e. the fact that there are many valid translations for any given source sentence, and comparing a translation to an arbitrary single reference

[3]Fully automatic MT quality estimation is an active field of research, but its main purpose is not to evaluate MT systems but to facilitate postediting by predicting whether it is cost-effective to post-edit a MT sentence.

may produce misleading results. This problem, sometimes called reference sparsity, affects all automatic metrics based on reference translations.

While the work on automatic metrics in Miller and Beebe-Center (1958) was visionary and predicted many features of future automatic metrics, it was only tested preliminarily, and it had essentially no impact on any future research. Carroll (1966) dismissed word-counting and other objective methods, because the same methods could be used for producing translations, and that might introduce circularity into the evaluation procedure (it could be argued that this did in fact happen once automatic metrics finally became established, as ngram-based evaluation metrics have been shown to be biased towards ngram-based MT methods).

No work seems to have been done on automatic evaluation until S. Thompson (1991) first introduced MT evaluation with edit distance metrics. This work was continued within the TextEval project (Thompson and Brew, 1996), where various statistical evaluation methods were also tested and proposed, including counting parts of speech, word classes and n-grams. The TextEval project pioneered many methods which were later widely adopted in MT evaluation, such as the use of multiple reference translations, weighted edit distance scoring, and using parts of speech and word classes as partial match categories. However, neither the statistical nor the edit distance metrics could be shown to be effective in measuring translation quality and the implementation was too cumbersome. Although slightly better known than Miller and Beebe-Center (1958), TextEval project seems to have had as little impact on further automatic evaluation research. Intensive research on automatic MT evaluation started only after SMT work picked up pace in the nineties.

3.5.2 Edit distance and precision/recall

Most automatic MT evaluation metrics are based on two distinct approaches to measuring similarity, which were popular in other NLP fields before being adopted for MT use. Edit distance metrics have been used since the 80s to evaluate speech recognition performance. Since SMT grew out of speech recognition, it was natural to adopt edit distance also to the task of MT evaluation. Edit distance metrics evaluate a hypothesis against a reference by calculating how many changes (edits) are required to transform the reference into the hypothesis. The earliest such metric is the Levenshtein distance, where the possible edit types are insertions, deletions and substitutions of single characters in a string:

$$\text{Levenshtein distance} = \textit{insertions} + \textit{deletions} + \textit{substitutions} \tag{2}$$

Precision and recall originate from information retrieval. The main difference between the approaches is that while edit distance metrics evaluate translations as a sequence, precision and recall metrics see them as bags of separate objects, which have no general structure. Precision penalizes mistakes and recall omissions:

$$\textit{precision} = \frac{|\text{tokens both in hypothesis and reference}|}{|\text{hypothesis tokens}|} \tag{3}$$

$$\textit{recall} = \frac{|\text{tokens both in hypothesis and reference}|}{|\text{reference tokens}|} \tag{4}$$

Using precision or recall exclusively for evaluation may result in evaluation errors with certain types of input. Precision favors hypotheses that are short relative to the reference and contain tokens with high likelihood of occurring in the reference. For instance, using the single token "the" as a hypothesis will result in perfect precision for the majority of English sentences. At the other extreme, a list of all possible tokens in English will achieve perfect recall in most cases. Because of this potential for error, precision and recall are often combined. The most common combination technique is the F-measure, the harmonic mean of precision and recall:

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5}$$

It is worth noting that edit distance metrics inherently balance precision and recall. The deletion cost implements a feature similar to precision by penalizing extraneous tokens, while the insertion cost is functionally similar to recall, as it penalizes the hypothesis based on how few of the reference words it contains. It is also possible to emphasize the similarity to precision or recall in edit distance metrics by adjusting the edit costs (zero insertion cost results in a precision-like metric, and zero deletion cost in a recall-like metric).

3.5.3 Automatic MTE metrics

The first automatic metric utilized in MTE research was word error rate (WER), which had previously been used in ASR research. WER is standard

Levenshtein distance applied to words instead of individual characters, and Tillmann et al. (1997a) was the first article to report WER scores for MT. WER is not well suited for MT evaluation, since it cannot fairly score translations where tokens are not in the same order as in the reference translation: moving a token requires a combination of deletion and insertion edits, which is too costly compared to the relatively low cognitive cost of token movement.

Position-independent word error rate (PER) was introduced in Tillmann et al. (1997b) to compensate for the word order problems affecting WER. In PER, the candidate and reference sentences are treated as bags of words instead of sequences, so order becomes irrelevant. However, entirely disregarding order brings its own problems, as even completely incomprehensible token orders become equal to perfect ones.

The most prominent and still dominant MT metric BLEU was introduced some years after WER and PER (Papineni et al., 2002). BLEU is a precision-based metric, which penalizes short translation by applying a brevity penalty. In BLEU, precision is calculated over ngrams, so BLEU takes token order into account (unlike PER) while not penalizing token movement too harshly (as WER does). BLEU was designed as a precision metric in order to allow the use of multiple references (recall is difficult to calculate if there multiple variable length reference translations), but in practice multiple references are rarely used, although there are indications that single-reference BLEU is unreliable Lommel (2016).

After BLEU, several metrics based on both precision and recall were introduced, the most notable of which are GTM and METEOR. GTM (Melamed et al., 2003) is conceptually simple, and calculates an F-score over aligned ngrams of different sizes. METEOR calculates an F-score with heavy emphasis on recall over tokens (Lavie et al., 2004). The emphasis on recall was due to results that indicated that manual annotation results correlated strongly with recall but not with precision. To account for token order, METEOR includes a fragmentation penalty that rewards matching spans of tokens between hypothesis and reference. Later versions of METEOR (Denkowski and Lavie, 2014) include tunable weights for different types of matches and functionalities for matching hypothesis tokens if they have the same stem as a reference token or are synonyms or paraphrases of the reference token.

Several metrics adapt edit distance to MTE by adding a block movement functionality that eliminates the main shortcoming of WER. The most popular of these is TER (Snover et al., 2005). TER uses a heuristic to reduce the complexity of block movements, which is otherwise NP-complete (Shapira and Storer, 2002). TERp Snover et al. (2009b) is an extended version of

TER, which includes two features familiar from METEOR: edit types based on stem, synonym and phrase matching and tunable edit weights.

Recent automatic MTE metrics allow character-level variations in matching (Popovic (2015), Virpioja and Grönroos (2015)), which improves performance with morphologically complex languages. Another recent trend is using a large amount features to alleviate the brittleness of conventional metrics (Stanojevic and Sima'an, 2014).

3.6 Error analysis and classification

Another approach to MTE is to investigate the errors that an MT system makes. By classifying and counting these errors, the evaluator can collect more fine-grained information about the quality of the translation than is provided by intrinsic evaluation with a single figure of merit. Error analysis (EA) is the form of MTE that is most closely related to HTE, and error taxonomies in recent HTE standardization efforts are designed to be also usable for MT error analysis (Lommel et al., 2014a).

Error analysis has undoubtedly been used in an adhoc manner since the early days of MT, and early approaches to structured error analysis have been described e.g. in Flanagan (1994). Vilar et al. (2006) introduced a general error analysis framework, which has influenced most of the current work on EA. A good summary of the state of the art in EA is provided by Popovic (2018), which also specifies a list of questions that EA can theoretically provide answers to:

> What are the most prominent problems of a translation system? What are particular strengths and weaknesses of the system? Does a particular modification improve some aspect of the system, although perhaps it does not improve the overall score? Does a worse-ranked system outperform a best-ranked one in some aspect? Are some types of errors more difficult to post-edit than others?

These are questions for which MTE metrics with a single figure of merit, be they manual or automatic, cannot provide answers.

3.6.1 Error taxonomies

A central concept in EA is the error taxonomy, which specifies the types and subtypes of possible errors. Many translation quality standards specify error taxonomies of varying complexity, including LISA QA, SAE J2450, MQM, and TAUS DQF, and localization tools often introduce their own error taxonomies. MT research literature likewise contains a large selection of different error taxonomies (see chapter 2.1. in Popovic (2018)).

The proliferation of error taxonomies is partly caused by the difficulty of defining error types exactly, and partly by differences in the intended use of the error taxonomy. For instance, the error taxonomy in the MQM framework is intended as a superset of error categories, which can be used as a template for more restricted taxonomies, so it contains over hundred error categories. Error taxonomies used for MT error analysis are generally much simpler, since complex taxonomies make annotation more difficult and less consistent. Usually error categories are defined concretely in terms of the corrections that they require, as in Vilar et al. (2006):

Main category	**Subcategory 1**	**Subcategory 2**
Missing words	Content words Filler words	
Word order	Word level	Local range Long range
	Phrase level	Local range Long range
Incorrect words	Sense	Wrong lexical choice Incorrect disambiguation
	Incorrect form Extra words Style Idioms	
Unknown words	Unknown stem Unseen forms	
	Incorrect form	
Punctuation		

Concrete error types are clearly easier to grasp for the error annotators, who are generally non-linguists and unfamiliar with the meaning of the linguistically motivated categories.

3.6.2 Manual error analysis

Most error analysis is based on manual error annotation by bilinguals or monolinguals. Manual error annotation is a more demanding task than manual adequacy or fluency evaluation, as the annotator needs to be able to recognize specific errors and also to learn how to classify them into the categories of the taxonomy. While most native speakers are capable of providing fluency and adequacy judgements, error annotation ideally requires people with analytical skills and a good command of the standard written language variety, such as professional writers and translators.

When error annotation is performed by bilinguals, the source text can be used to identify errors in the translation (in HTE, this is the only context in which error annotation is performed). When annotators are monolingual, a reference translation is required. Popović and Ney (2011) describes three ways of carrying out manual error analysis with the help of a reference translation:

1. Strict: The hypothesis translation is compared directly to the reference translation, and all deviations from the reference are considered errors.
2. Flexible: Substitution of words and phrases with meaning equivalents and different syntactically correct word orders are not considered errors.
3. Free: Any deviation from the reference that preserves the meaning of the reference is acceptable.

It is obvious that the method where a bilingual annotator checks the hypothesis against the source text is theoretically capable of yielding the most accurate error annotation: even with the free reference comparison method, omissions, additions and different ways of presenting source text information may mean that the meaning of the reference translation differs materially from the source text, which in turn means that some correct hypothesis translations are judged to contain errors. This effect can be reduced by using multiple reference translations, but it cannot be eliminated.

However, even if error annotations based on the source text are potentially the most accurate, there are practical problems with relying solely on the source text as a knowledge source. In the first place, it is difficult to find bilingual annotators with sufficient language skills in both the source and target languages to perform the error annotation task. Besides the resourcing problems, an experiment on manual evaluation with a single figure of merit

(Guzmán et al., 2015) seems to indicate that using bilingual annotators and the source text alone lowers the consistency of annotations. A similar effect may exist for error annotation based strictly on the source text.

In practice, error annotation seems to be mostly performed with both the source text and a reference translation as sources of information, and the hypothesis is compared to the reference translation freely. This is the setup used for instance in Klubicka et al. (2018), which contains an error analysis using an MQM-based taxonomy. Despite the importance of the details of the error annotation setup, it seems to be fairly rare for this information to be exactly specified in experiment descriptions.

Klubicka et al. (2018) also reports relatively low IAA, which is consistent with other studies looking at IAA for error annotation. Lommel et al. (2014b) found three main causes for this low IAA:

1. Disagreement about the span of text in the hypothesis that contains the error
2. Unclear or ambiguous error categorization
3. The annotator's personal interpretations of what constitutes an error

These causes can in principle be addressed by providing more training to the annotators and designing taxonomies better, although intensive training is usually precluded by the lack of resources. However, all three of these causes are partly caused by the variability of natural language, which is much more difficult to control. If errors are annotated against a free semantic interpretation of the reference sentence, there often exists multiple ways to correct an erroneous translation to a semantically correct one, with comparable effort of correction. Lommel et al. (2014b) describes a real example of this effect, where the Spanish word "es" could be annotated alternatively as a mistranslation, omission or an agreement error, depending on whether the correct version is interpreted to be "tu eres", "usted es" or "eres".

Using the strict or flexible methods of reference comparison sidesteps the problem of language variability, as it constrains the annotator to a single interpretation. But it also entails the problem of reference sparsity, as there's no guarantee that the reference is or even resembles the valid translation that the hypothesis can be corrected to with the least effort. As with automated metrics, this reference sparsity can be alleviated by using a targeted reference, which are nowadays often available in the form of post-edited MT.

EA with targeted references can be seen as a two-step process, where the post-editor first implicitly marks the errors in the hypothesis by performing corrections, and then the error annotator categorizes the errors. One problem with this approach is that the intention of the post-editor normally is not to mark errors but to produce translations efficiently. This means that the edits that the post-editor makes are probably biased towards the ergonomically and cognitively light errors, and in many cases the post-editor will reject the hypothesis entirely and translate from scratch, which invalidates the error analysis entirely.

Annotation consistency then seems to be the main problem of EA, and it is something that has be to taken into account when comparing different systems with EA. It is always possible that two equal MT systems receive very different error counts from different annotations sessions, if the annotators have different interpretations of error types and annotation practices. Trade-offs have to be made between consistency, granularity of the taxonomy and the correctness of the analysis. Consistency can be achieved by constraining the error annotation and using a simple taxonomy, but this leads to error annotations that are often incorrect or not informative enough.

3.6.3 Automatic error analysis

Manual error annotation is expensive and inconsistent, as was discussed above. It would also obviously be desirable if error annotation could be performed automatically, as that would allow MT developers to run error analysis on demand for every iteration of their systems. However, automating error analysis is even a harder task than automating score-based quality evaluation, as it requires implementing mechanisms for identifying a sufficiently large range of error types. It is also difficult to validate that the automatic error analysis actually produces results comparable to manual error analysis, partly due to the low consistency of human annotations.

There is a natural connection between automatic intrinsic metrics and automatic error analysis methods: most automatic metrics include components (ngrams in BLEU, edits of different types in TER etc.), which each contribute to the metric score according to some formula. Often these components are justified by claiming that they are related to certain aspects of translations, and that a weak component score indicates an error in the aspect related to that component. As was mentioned, in BLEU lower ngrams (or in some variants lemma ngrams) are hypothesized to correspond with adequacy, and the higher ngrams with fluency. In edit distance metrics, insertion and deletion

can be interpreted as indicating missing or extra words, and block movement as reordering errors.

Automatic metrics were first utilized for error analysis in Popovic et al. (2006), in which a combination of WER and a variant of PER (modified to behave more like precision/recall) was used to identify inflection and missing word errors. This method was expanded in Popović and Ney (2011) to include reordering errors, extra words and incorrect lexical choice, and it is used in the open source tool Hjerson.

The basic idea is to specify conditions for errors based on properties of the WER and PER scores of the hypothesis. For instance, the condition for identifying a reordering error is that the erroneous word is marked as a WER error but not as a PER error, i.e. it occurs in both the hypothesis and the reference but not in the same location. For inflectional errors, lemma comparison is used. Zeman et al. (2011) tested a similar method (available in the form the Addicter tool) using word alignments between the hypothesis and the reference translations rather than WER/PER.

The results of the method in Popović and Ney (2011) were compared to human error annotations produced using either strict or flexible reference comparison, and the correlation was found to be sufficient for both the strict and flexible annotations (although noticeably worse for the flexible annotations). In Zeman et al. (2011), a corpus of translations manually annotated with errors (with source and a semantically interpreted reference as sources of information) and reference translations was used for validating the method. The precision and recall of the errors that the automated method identified was calculated against the manually annotated errors, but they were disappointingly low (the best F-score was 25.80).

Several error-annotated corpora of MT were consolidated and homogenized in Fishel et al. (2012) to serve as a resource for testing automated error annotation methods. The corpus is partly annotated freely and partly flexibly, and tests using Hjerson and Addicter on the different parts of the corpus again indicate that automated error analysis methods correlate with human annotations much better if the human annotations have been constrained to the reference translation in some way. This is of course to be expected, since the methods of automated error analysis used are functionally similar to the strict method of error annotation, i.e. they do not allow valid alternative constructions or synonyms.

As automatic error analysis is tied to strict reference comparison, it is probably consistent, but will severely overstate the amount of true errors. It suffers from reference sparsity just as much as automatic metrics, and prob-

ably more because of its greater granularity. Problems caused by reference sparsity in error analysis have recently been studied in the related field of grammatical error correction (GEC), where the effect of reference sparsity (or low coverage bias, as it is known in GEC) was found to be profound (Choshen and Abend, 2018).

Using multiple references or especially post-edited targeted references alleviates this problem considerably (Fishel et al., 2012), but multiple references are rarely available and the use of targeted references requires a semiautomatic process. Despite these concerns, automatic error analysis is a valuable part of an automatic evaluation toolkit, since other automatic methods have comparable weaknesses of different nature.

4 Generating large sets of semantically highly similar translations for MT evaluation

As was discussed in the previous chapter, reference sparsity is one of the main reasons behind the unreliability of automated MTE methods. Some methods used to alleviate reference sparsity, including stemming, lemmatization, string similarity, synonym sets and phrasal paraphrase tables generated from corpora, were also mentioned. Two of these methods, synonym sets and phrasal paraphrase tables, belong to the wider field of paraphrase generation, which is used in many areas of NLP (see Madnani and Dorr (2010) for a survey).

The EvalGen system developed in this thesis is also based on paraphrasing, and in particular sentential paraphrase generation (SPG), where the intention is to create paraphrases of whole sentences. Furthermore, the system is based on a hand-crafted grammar, which generates the sentential paraphrases from a semantic representation derived from the sentence to be paraphrased.

The crucial restriction that makes EvalGen implementable is that the sentences to be paraphrased are manually selected, edited to reduce certain ambiguities, and then analyzed manually into semantic structures. This sidesteps the most error-prone and complex part of paraphrase generation, namely the automatic parsing and disambiguation of meaning. The sentences generated from these semantic structures will then constitute the set of reference translations. While this simplification will mean that the reference translations form an unnatural fragment of NL, this appears to be the only way to accurately cover a large enough portion of the space of possible translations for a source sentence.

The concepts of precision and recall are again relevant here. Precision and recall of an SPG system can be defined as follows:

$$precision = \frac{|\text{correct paraphrase candidates produced}|}{|\text{paraphrase candidates produced}|} \tag{6}$$

$$recall = \frac{|\text{correct paraphrase candidates produced}|}{|\text{correct paraphrases}|} \tag{7}$$

For certain types of MTE evaluation using paraphrases, such as error analysis, both precision and recall must be relatively high. At their current state, corpus-based paraphrasing methods are not reliable enough. Metzler et al. (2011) compares three different corpus-based methods and reports a maximum precision of just 0.25 for sentential paraphrases created by verb phrase substitution (only a small subset of possible sentential paraphrases). Pavlick et al. (2015) compares different domain-specific paraphrasing methods, and no single model can simultaneously achieve both precision and recall of over 0.5, even though the recall is defined over a very limited set of possible paraphrases.

4.1 Related work

4.1.1 Sentential paraphrasing for MTE

The use of sentential paraphrases as automatic MTE metric references was first explored in Kauchak and Barzilay (2006), where reference translations were paraphrased in a way that maximized their word overlap with the MT hypotheses. The paraphrasing method used was synonym substitution with contextual restrictions, which means that the paraphrases always retained the grammatical structure of the original reference translations.

Owczarzak et al. (2006) generated a larger set of paraphrased reference sentences by utilizing phrase pairs extracted from a bilingual corpus. The phrase pairs were divided into sets based on the source phrase, and paraphrases were generated by substituting reference sentence phrases with phrase alternatives from the phrase pair sets.

Both of the above methods resulted in better correlation with manual MTE results, even though they only entailed fairly superficial paraphrasing.

Barancikova and Rosa (2015) describes a more sophisticated paraphrasing method, where paraphrases are generated by lemma substitution and con-

strained reordering using a rule-based NLP system. Both lemma substitutions and reordering were found to improve correlation with manual evaluation scores, when paraphrases generated using these methods were used with BLEU and METEOR.

The most linguistically intricate method of generating sentential paraphrases for MTE is found in Fomicheva et al. (2015). The article defines a typology of structural changes typically involved in translation, which is then used as a basis for hand-crafted transformation rules. These rules transform the changed structure in a reference translation into a more literal (but still admissible) translation of the equivalent source language structure. The motivation for the method is that MT will often replicate the structure of the source sentence more precisely than human translators, so more literal versions of the reference translations generally resemble the MT hypotheses more.

While all of the methods referred to in this section improve the accuracy of reference-based MTE metrics, their paraphrasing capabilities are severely limited and cover only a small fraction of acceptable variation in translations. The next section describes prior work in which this problem is alleviated by utilizing manually generated paraphrases.

4.1.2 Manually created paraphrase sets for MTE

Among existing literature, the methods that resemble EvalGen the most are Dreyer and Marcu (2012) and Bojar et al. (2013). In these prior systems, bilinguals or native speakers were asked to come up with target language paraphrases for parts of source sentences (or their human translations), and the set of reference translations was made up of all the possible combinations of the paraphrases that covered the whole source sentence. A separate semantic or pragmatic representation is not used, but semantic and pragmatic analysis is implicitly performed by the annotators as they come up with different ways of expressing similar meanings. This method will be called compositional manual SPG from here on.

In Dreyer and Marcu (2012), 102 Chinese and Arabic source sentences from the 2010 Open MT NIST progress set were used, and English paraphrases (called meaning equivalents in the article) for parts of the source sentences were generated by three different protocols:

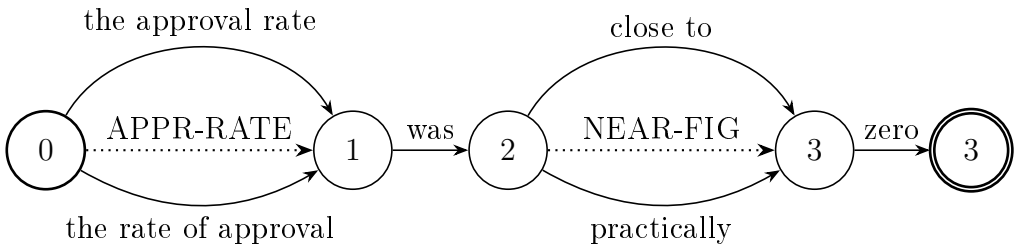
1. Foreign language natives created the meaning equivalents based on the source sentences.

2. English natives created the meaning equivalents on the basis of a best translation identified in the NIST evaluation.

3. Same as 2, but the annotators also had access to three other human translations.

The meaning equivalents were stored as recursive transition networks (RTNs, called "cards" in the article) with no morphological or syntactic context, so there was no mechanism for ensuring that combining them in different ways would create grammatical sentences. However, due to the morphological simplicity of English, this seems not to have been a major problem: when a selection of generated sentences (the ones closest to translations to be evaluated) was presented to three English speakers, 90 percent of the sentences were judged grammatical by at least one of them. Despite the use of a RTN formalism, the structure of the networks encoding the meaning equivalents is not recursive, i.e. no card may have itself as its descendant. The networks therefore have equivalent finite-state automatons.

This graph demonstrates the structure of the networks:



The parallel edges represent different cards, i.e sub-RTNs that can be used to replace labeled sub-RTN edges in the larger RTN (glossed here as "APPR-RATE was NEAR-FIG zero", with capitalized words representing the sub-RTN labels). This method is not easily extensible to morphologically complex languages. Sub-RTN boundaries do not necessarily follow linguistic phrase or construction boundaries. If a constituent within a sub-RTN governs a constituent in the parent or a sibling RTN, and the meaning equivalents for the sub-RTN have different government properties, the correct government relation cannot be enforced in this setup. For instance, if the above example was to be translated directly into Finnish, the meaning equivalents of the sub-RTN NEAR-FIG would require the translation of the word "zero" to be either in the partitive or the nominative case depending on the selected meaning equivalent, but there's no way to enforce that restriction with this method. The Finnish translation of this network would therefore generate one grammatically incorrect sentence alternative for each correct one.

To evaluate translations, an edit distance metric based on FST composition was used to calculate the edit distance from the translation to the nearest reference translation. This method will be covered at length in chapter 6.

Bojar et al. (2013) tested a similar method with English to Czech translation. As Czech is morphologically much more complex than English, the meaning equivalents were stored in a unification-based format (with constituents called "bubbles"), which placed morphological constraints on the combination of the parts. This meant that the annotators had to record morphological properties of the meaning equivalents that they added, such as gender and case. Prolog programming language was used to perform the annotation, and the authors report that the annotators had no trouble with this seemingly complex task, despite not having a programming background. Meaning equivalents were created for 50 sentences from the WMT11 test set. Evaluation was performed using standard BLEU.

Both of the prior articles report some success with the generated test sets. The HyTER system is shown to be capable of distinguishing HT from MT much more accurately than BLEU, TERp or Meteor, and it also appears to be better at ranking MT systems in the same order as human annotators. In general, its characteristics were closer to human evaluation than the conventional automated metrics. In Bojar et al. (2013) the test set was evaluated by seeing if the ranking of MT systems it provides correlates better with human rankings than the rankings made with different-sized test sets containing 1,2 or 3 reference translations. The result was that just by using five generated references for each of the 50 sentences in the annotated test set, the BLEU score correlated approximately as well (Pearson correlation 0.69) with human rankings as when the whole single-reference WMT11 test set (3003 sentences) was used. The correlation increases when more generated references are used, reaching a correlation of 0.78 at 50.000 generated references.

Despite these positive results, both of these efforts apparently only managed to cover a fairly small portion of the possible translations for the selected source sentences. Dreyer and Marcu (2012) reports that average normalized edit distances between independently produced human reference translations and the closest generated reference were relatively high (34 percent for Chinese, 19 percent for Arabic). While Bojar et al. (2013) did not perform similar tests, the BLEU correlation curve flattens and stays below 0.8 with more than 5.000 generated references, which may indicate low recall.

The low recall is not a serious problem if the purpose is simply to use the generated references in conventional automated metrics. However, if the sys-

tem is to be used for error analysis, low recall may lead to false negatives and make the error counts too noisy to provide real insight into the characteristics of the MT system.

4.2 Semantic aspects of compositional manual SPG

As mentioned, semantics is included in the HyTER and Bojar et al. (2013) systems only implicitly through the judgements of meaning equivalence that the annotators make as they generate paraphrases for parts of the sentence. EvalGen is different from these systems in that the semantics are more explicitly defined.

Before moving onto a description of EvalGen, it is worth trying to make the semantical component of paraphrasing by human translators more explicit. First it should be noted that the process which leads to the proliferation of the amount of acceptable translations for any source sentence must be compositional. This process can be described in simplified, abstract form as follows (this description is not meant to be a theory of translation or meaning, it is simply meant to illustrate the process, whatever its real nature may be):

The semantic and pragmatic meaning or function (called simply "meaning" from here on) of the source sentence is parsed by the translator, and then the translator comes up with a construction that can express the same (or at least a similar) meaning in the target language. There are usually several target language constructions available that can perform the desired role (probably with slightly different connotations). If the source sentence meaning is compositional, i.e. it can be said to have constituents that can be varied with predictable changes in the meaning, the chosen target language construction will probably also have the same compositional structure (let's call the compositional structure of the meaning "semantic tree" and the compositional structure of the target language construction "target language tree"). The translator assigns the constituents in the target language tree values that have the same (or similar) meaning as the corresponding constituents in the semantic tree. These constituents and their subconstituents may also have multiple acceptable target language equivalents. Each constituent alternative will increase the amount of acceptable sentence translations by the amount of possible target language trees that they can be part of.

For example, consider an extremely simple source sentence: "the ball is red". Without any context, this sentence would most likely be parsed as a simple predicate clause, which semantically ascribes a color to a physical object. Its meaning could be represented in function-argument form as HAS-

COLOR(RED,BALL). In this example, the function HASCOLOR has two arguments, color and physical object. For sake of clarity, let's assume we are translating into English, where this function would correspond to e.g. the following constructions:

1. The OBJECT is COLOR
2. The color of the OBJECT is COLOR
3. The OBJECT is COLOR in color
4. The OBJECT's color is COLOR
5. The OBJECT has a COLOR color

So even this extremely simple meaning can be expressed with five different constructions. This set of constructions is shared by many meanings that ascribe a simple property, such as size or shape, to a physical object. The constituents (COLOR and OBJECT) of these constructions in this example, RED and BALL, are very simple and have only one obvious alternative each in English, "red" and "ball". If we add internal structure to one of the arguments, the number possible paraphrases grows quickly. For instance, if we replace "ball" with "ball on the table", we have at least the following options for the OBJECT argument values (to save space, they are written in regular expression format, where the '|' characters inside parentheses indicate possible variation):

1. ball (on|on top of) the table
2. ball (located|sitting|placed) (on|on top of) the table
3. ball (that's|that is|which is) (on|on top of) the table
4. ball (that's|that is|which is) (located|sitting|placed) (on|on top of) the table

You can distinguish three sources of variability here: "on" and "on top of", verbs expressing location, and a NP/PP or NP/S phrase structure. This sort of variability quickly leads to a combinatorial explosion of paraphrases, even with relatively simple and non-ambiguous sentences. It is clear that building a relatively complete and correct set of constituent paraphrases will require

specifying complex lexical, syntactic, morphological and semantic rules, especially if dealing with morphologically complex languages. Dreyer and Marcu (2012) and Bojar et al. (2013) attempt to capture this multiplicity of paraphrases in a simple and fairly cost-effective way. However, their methods rely on input by annotators who cannot be relied to have specialist skills in linguistic analysis, which are needed to construct such rules.

4.3 Evaluation using grammar engineering

The task of building rulesets that automatically generate correct natural language sentences has traditionally been a part of the field of grammar engineering. Using grammar engineering for evaluation is not a novel idea, it has for instance been used to evaluate the correctness of linguistic theories (Bender, 2007). In that particular use case, a linguistic hypothesis is encoded in a grammar, which is then used to parse a test suite ideally containing "hand-constructed negative and positive data and naturally occurring corpus data". The parsing performance of the grammar then supports or disproves the correctness of the linguistic hypothesis.

This kind of linguistic hypothesis evaluation relies on the method of fragments, originally introduced by the logician Richard Montague. Hendriks and Partee (1997) describes this method as follows:

> What is meant is simply writing a complete syntax and semantics for a specifiable subset ("fragment") of a language, rather than, say, writing rules for the syntax and semantics of relative clauses or some other construction of interest while making implicit assumptions about the grammar of the rest of the language.

The EvalGen approach (as well as HyTER and Bojar et al. (2013)) can be seen as an adaptation of this method: it isolates a semantic, cross-linguistic fragment of natural language and then tries to cover all the constructions related to that fragment as completely as possible. Instead of grammars encoding linguistic hypotheses, EvalGen uses grammars that are not motivated by linguistic theories but rather by how easily and concisely they can encode the set of possible correct sentences. Of course, the direction of evaluation is also reversed: EvalGen evaluates the data (i.e. MT output) based on how closely it conforms to the grammar, instead of evaluating the grammar on how well it predicts the data (although the correctness of the EvalGen grammars is also confirmed by checking how well they predict human translations, which will be covered in chapter 7).

Within the field of MTE, a precursor for EvalGen is the MLA-MT task (Castellanos et al., 1994). In this task, source sentences and their translations are generated with grammars and MT is evaluated based on how many test sentences it translates correctly. The task was only used with a very limited domain to evaluate MT systems trained on a corpus generated by the same grammars. There was no paraphrasing functionality, as the MT systems had no chance of learning different paraphrases from the corpus, so the task was not applicable for general MTE. Rather, it was meant for comparing different training methodologies in a restricted context, which is also a potential use case for EvalGen.

4.4 Two-way paraphrasing

The main benefit of using grammar engineering to build the test set is that it is conceptually easy to extend the paraphrasing into a multilingual process. Paraphrasing is handled as a process of constructing semantic trees and then compositionally generating surface trees from the semantic trees. This means that as long as the semantic trees are language-independent, generating similar paraphrases for another language is simply a question of defining a new set of mapping rules from the semantic trees to surface trees. The Finnish constructions corresponding to the elements of the example in the previous section would be the following:

HASCOLOR

1. OBJECT on COLOR
2. OBJECT on (COLOR väriltään|väriltään COLOR)
3. OBJECT-GEN väri on COLOR
4. OBJECT on COLOR-GEN värinen

OBJECT

1. pallo (pöydällä|pöydän päällä)
2. (pöydällä|pöydän päällä) oleva pallo
3. (pöydälle|pöydän päälle) (laitettu|asetettu) pallo
4. pallo, joka on (pöydällä|pöydän päällä)

With these rulesets, sets of sentences with a highly similar meaning can be created for both languages. If any of the sentences from the sentence of one language is translated, it can then be evaluated by comparing it to any of the sentences in the other language's sentence set.

As was mentioned, generating these parallel paraphrases requires that the semantic representation is language-independent, which is a very limiting requirement. In fact, it is almost certain that no such representation is possible for any reasonably large fragment of language. The feasibility of the method therefore depends on whether it is possible to isolate fragments of natural language that can be reduced to a language-independent representation and that are also general enough to yield useful information when used for MTE.

From here on, these language-independent semantic representations will be called semantic templates.

4.5 Selection of semantic templates

In HyTER and Bojar et al. (2013), the sentences for which the paraphrases are generated are selected from testsets used in translation evaluation campaigns. This means that they have at their disposal both machine translations from a variety of systems, and the manual evaluation scores for those machine translations. It is therefore easy to test the performance of the evaluation method by comparing its results against the manual judgments. The drawback of using this selection method, which is essentially random, is that the selected sentence might not be suitable for paraphrasing due to ambiguity, complexity, or incoherence.

The EvalGen method is much more sensitive to what might be called the paraphrasability of sentential meanings, i.e. whether the set of possible paraphrases for the sentence is both syntactically and lexically versatile and semantically uniform. There are several reasons for this, but the main reason is practical: the semantic templates and their bilingual surface representations have to be created by hand and the task is work-intensive, so the efforts should be concentrated on those meanings which can be easily converted into semantic templates and are also particularly interesting from the point of view of MTE. This kind of selection of test cases (which admittedly has the look of cherry picking) raises the issue of whether the semantic templates are representative of translation in general. The issue will be discussed in 4.5.4 after the selection principles have been laid out.

4.5.1 Ambiguity and sense fixing

One factor that complicates paraphrase generation is ambiguity. With normal monolingual paraphrasing the effect of ambiguity is probably modest and controllable, but with two-way paraphrasing it becomes much more problematic. Take for instance a fragment from the WMT17 testset, "the two still have major differences". The phrase "have major differences" is ambiguous in this fragment: it can mean 1. "have different characteristics" or 2. "disagree on some issues".

If a semantic template to be used for bilingual sentence generation is created on the basis of this sentence and rules generating this source sentence are included in the source ruleset, the task of generating a target ruleset that covers a sufficiently large part of acceptable translations becomes much more difficult. When the MT system translates the ambiguous sentence, both the translation expressing the sense "have different characteristics" and "disagree on some issues" should be accepted as correct, since there is no disambiguating context available to the MT system. Ensuring the acceptance of both senses requires adding considerably more rules, as both senses can be expressed with multiple constructions, which are listed in Table 1.

As the table demonstrates, some of the constructions are shared between the two senses and the ambiguity could be implemented with simple lexical rules. However, many of the constructions are allowed with just one of the senses, so accounting for ambiguity does increase the complexity of creating the rule set significantly, as different syntactic constructions need to be added.

One way of dealing with ambiguity is to ensure that there is always enough context within the sentence to resolve it. For instance, the sample sentence above "the two still have major differences" was originally embedded in a larger sentence: "Still, despite the public display of reconciliation, the two still have major differences." The phrase "public display of reconciliation" strongly indicates that the sense "disagree on some issues" is correct, although the other reading might still be possible if "reconciliation" and "the two" are interpreted as related to e.g. accounting. The sense can be fixed even more strongly by substituting a more precise expression for the underspecified phrase "the two". This sentence originates from a news article about the reconciliation between Turkey and Russia, so the context can be extended by replacing "the two" with for example "Russia and Turkey" or "the two countries". With this extra piece of information, the sense "disagree on some issues" becomes the only feasible sense.

This kind of sense-fixing is essential in creating semantic templates with a

Table 1: Possible Finnish constructions equivalent to "have differences".

Construction	X and Y have different characteristics	X and Y disagree on some issues
NP-NOM V NP-ELA-REFL	X ja Y eroavat toisistaan X AND Y DIFFER FROM EACH OTHER	
NP-NOM AP-PTV	X ja Y ovat erilaisia X AND Y ARE DIFFERENT	X ja Y ovat erimielisiä X AND Y ARE DIFFERENT-MINDED
NP-NOM NP-PTV		X ja Y ovat eri mieltä X AND Y ARE OF DIFFERENT MINDS
NP-ADE NP-PTV	X:llä ja Y:llä on eroavaisuuksia X AND Y HAVE DIFFERENCES	X:llä ja Y:llä on erimielisyyksiä X AND Y HAVE DIFFERENCES

wide variety of expressions in both languages and high coverage on the target language side. The optimal semantic template has a wide variety of surface expressions on the source side to allow the evaluation of multiple construction types with the same target set, while simultaneously having a target set that covers as much of the acceptable translations as possible.

4.5.2 Semantic template complexity and MT difficulty

Realistically, the sentences used as a basis of the semantic templates also have to be relatively simple and short. Even if it is possible to ensure that a long sentence has an unambiguous meaning, the different possibilities of expressing that meaning will quickly multiply with sentence complexity and length. The grammar designer will easily be overwhelmed by the multiplicity of different expressions, which will lead to errors and gaps in coverage.

Another consideration when selecting sentences as basis of the semantic templates is the estimated MT difficulty related to the semantic template. As the grammar design for the semantic templates is very work-intensive, it would be best to concentrate on those semantic templates that are associated with known MT problems. Before the introduction of NMT, almost any English-Finnish semantic template would be almost guaranteed to be problematic in some way, but with NMT the translations may in many cases be flawless. This was discovered during exploratory testing of the method, when the semantic template corresponding to "rebuilding real trust will be difficult, perhaps impossible" was tested. Practically all NMT systems correctly translated a dozen or so significantly different surface representations of this semantic template, even though it required enforcing morphological congruence in several places and significant word order changes.

The exploratory testing of the method also indicated that NMT is still affected by some traditional MT problems. The most obvious problems are to do with translating constructions that have no clear analogue in the target language (known as lacunary units in translation studies), and cases where disambiguating the meaning of some part of the source sentence requires the semantic parsing of other parts in the sentence.

Meteorological sentences are a prototypical example of a part of language where there's a lot of cross-linguistic variation between the constructions used to express similar meanings (Eriksen et al.). This kind of variation is quite significant between English and Finnish, so meteorological sentences are therefore a good source for semantic templates. This can be illustrated by the constructions related to raining in English and Finnish (constructions

types from Eriksen et al.):

Construction	English	Finnish
atransitive		sataa
expletive	it rains it is raining	
intransitive predicate		taivaalta tulee vettä/sadetta taivaalta sataa vettä vettä sataa
intransitive argument	rain is falling rain falls	 sadetta tulee
existential	there is rain	on sadetta

Of course, some of these forms of expression are only used is specific contexts, and they all have different connotations relating to the properties (such as duration and concurrency) of the raining event. For instance, the existential construction type is generally used with an adverbial of time or place in both Finnish and English (as in "there was rain during the night").

What's noteworthy is that there's not a single construction in the table that has a straightforward match in the other language's column. The intransitive argument and existential constructions are present in both languages, but their constituents are very different. In the intransitive argument construction, the Finnish uses the verb "to come" and English the verb "fall", and the Finnish "rain" argument is an adverbial while the English is a subject. The English existential construction uses the dummy subject "there", while the Finnish one has a predicate-subject structure. This means that there's no straight-forward mapping from the English words and word forms into the corresponding Finnish words and word forms. Simple experiments with online MT engines confirm that NMT does in fact struggle when translating these sorts of sentences.

Cross-linguistic incompatibility of meteorological sentences is ubiquitous between English and Finnish. Many meteorological events can be encoded as verbs in one language, but not in the other. For example, Finnish has a verb ("tuulla") with the meaning "wind blows", while English has a verb for snowing, which Finnish somewhat surprisingly lacks (the construction "to rain snow" is used instead).

There are many other linguistic functions which are encoded in different ways in English and Finnish and which NMT seems incapable of addressing,

particularly cases where English auxiliaries correspond to Finnish verb forms. For instance, the English causative construction with a "have" auxiliary often translates into Finnish as a curative construction, where the causation is encoded as a (semi-productive) verb affix. "I had my car repaired" is most directly translated as "korjautin autoni", but this also seems to be a very difficult sentence for MT to translate correctly due to the obvious structural mismatch.

As mentioned, the other traditional MT problem that still affects English to Finnish MT is disambiguating lexical meaning, even if the context within the sentence makes clear what the correct lexical sense is. Pronoun disambiguation is the aspect of this problem that has received most attention during the recent years. In the case of English to Finnish MT, the English pronouns that cause most problems are the ones that do not specify information that is obligatory in the Finnish equivalent pronouns: "you", which does not specify number, and "they", which does not specify animacy. One potential use case for EvalGen is as a test suite for detecting problems in the translation of these kinds of constructions, along the lines of Guillou and Hardmeier (2016).

Another disambiguation problem occurs if the senses associated with an English word cannot be mapped one-to-one to the senses of a single Finnish word. One such English word is "tile". This is almost exclusively translated into Finnish as "laatta", except in the case of a roof tile, which is translated as "tiili" (cognate of "tile", but means "brick" in Finnish). This distinction seems to be lost on NMT, even if the context makes disambiguation simple (an example from the Internet: "a tile had blown off my roof"). This is an extremely common problem with rare senses of common words, and also affects such words as "car" (sometimes means "train car") and "device" (sometimes means "method"). These are all examples of the classic MT problem first demonstrated by Yehoshua Bar-Hillel with his famous example, "the box is in the pen" (y. Bar-Hillel, 1965). NMT still seems to fail this test even with cases with plentiful sentential context (such as "I threw the ball to the child in the pen").

4.5.3 Extending the sentence sets by using interchangable constituents

The most novel aspect of EvalGen is that it can produce a large number of significantly different but semantically similar source translations for each semantic template. This means that each semantic template provides a large number of data points for MT evaluation. There is a simple to way to make

the amount of possible source variants for each semantic template even larger.

Sentences usually have constituents which can be exchanged with other constituents of the same class without introducing semantic changes other than those directly related to the constituent. In other words, the constituents are independent and not a tightly integrated part of a larger construction within the sentence. Prototypical examples of these kinds of constituents are proper names: they generally have a fairly straight-forward function of referring to things, places, people etc., except for certain idiomatic uses, such as "Judas". At the other end of the scale are constituents that are a completely fixed part of a larger construction, such as the pleonastic "it" in "it rains", which cannot be substituted by any other pronoun.

Generally, constituents headed by nouns are more likely to be interchangable than other constituents. Substituting constituents headed by verbs is especially difficult, since they generally govern other constituents in the sentence, and the verb can only be exchanged with another verb that has a frame structure consistent with the other constituents in the sentence. For instance, "sent" in "I sent him a note by mail" can only be syntactically exchanged with other ditransitive verbs, and semantically the prepositional phrase "by mail" restricts the set of valid substitute verbs even further. Even if the verb substitution were syntactically and semantically valid in one language, the set of paraphrase constructions might be different: "I sent him a note by mail" can be paraphrased as "I mailed him a note", but that paraphrase construction would not be available with other verbs.

By allowing limited semantic variation of independent constituents, much larger set of sentence variations can be produced with a single semantic template. For instance, if the semantic template is modeled on the sentence "it rained yesterday", the constituent "yesterday" can be treated as a semantic variable, which may be substituted by expressions from the class of expressions denoting time, such as "last week" or "on 21st of June".

When these semantic variables are used, semantic templates generate multiple sets of source and target sentences, and the translation of source sentence should only be compared against the target sentence set generated with the same semantic variable values as the source sentence.

The constituents that are suitable for use as semantic variables are those that are structurally the simplest, so implementing more variants for them is comparatively easy. Semantic variables can be used to inexpensively expand the number of data points available for MT evaluation. Therefore, another desirable property of a semantic template is that it has constituents which can be adopted as semantic variables.

4.5.4 Applicability of hand-crafted cases for evaluation

One of the proposed uses of EvalGen is intrinsic quality evaluation of MT output. Hand-picking semantic templates mainly from simple cases known to be problematic for MT would appear to make any intrinsic evaluation unreliable. The intrinsic score assigned to MT system would seem to depend more on how they address the limited amount of problem cases encoded in the semantic templates rather than how well they translate in general.

Ultimately the suitability of EvalGen as a standalone general quality metric can only be verified by how well its' score for MT systems correlates with the manual evaluation results of the same MT systems on a conventional testset. Regardless of whether there is a strong correlation, EvalGen scores can be used to supplement other intrinsic metrics, as the different approach that it uses might give useful into e.g. how two systems with comparable BLEU scores but different manual evaluation scores differ.

The primary use case for EvalGen is an error analysis tool. In this use case the focus is more on understanding the workings of MT systems and discovering flaws in them, so the possible biases in the semantic templates are not as problematic. In this respect EvalGen resembles Isabelle et al. (2017), where MT systems are evaluated against a hand-crafted challenge set consisting of sentences exhibiting cross-linguistically interesting phenomena.

5 Implementation of semantic templates

To recap, semantic templates are sets of language-independent, compositional semantic trees. Constituents of the trees are semantic constructions which may contain subconstituents. Some constituents are semantic variables, which may have several different values. A constituent can only be a semantic variable if its value can be varied without any effect on the semantics of the rest of tree. Each semantic tree constituent is mapped to sets of paraphrases in the source and target languages. The paraphrases have the same compositional structure as the corresponding semantic tree constituent.

The natural way to implement a semantic template is as a grammar which generates all the paraphrases: the set of paraphrases is the language that the grammar describes. The concept of encoding paraphrase sets as grammars first appears in Dras (1997), and synchronous paraphrase grammars learned from corpora have been popular in statistical SPG (Weese et al., 2014).

In EvalGen, paraphrase generation is done for two languages, so a pair of

grammars is needed. Also, the set of paraphrases varies depending on what permutation of possible semantic variable values is used, so some mechanism is needed to connect the pair of grammars so that a translation of a source paraphrase is only compared to target paraphrases generated with the same permutation of semantic variables.

It should be noted that the language (i.e. set of paraphrases) that the grammar has to describe is very simple, with no recursion and a limited amount of structures. Therefore the expressive power of the grammar formalism is not particularly relevant, as almost any formalism is powerful enough to encode the paraphrases (and they are in fact converted into finite-state machines later). For EvalGen, the most important aspects of a grammar formalism are how easily the paraphrase sets can be encoded with the grammar formalism and how suitable the grammar formalism is for cross-linguistic comparison.

A minimal grammar pair could be defined as context-free grammars (CFG) in the following manner (using the "the ball is red" semantic template as an example, with "red" as semantic variable with an alternative value of "green"):

English

⟨*S*⟩ → The OBJECT is COLOR
| The color of the OBJECT is COLOR
| The OBJECT is COLOR in color
| The OBJECT's color is COLOR
| The OBJECT has a COLOR color

⟨*OBJECT*⟩ → ball

⟨*COLOR*⟩ → red
| green

Finnish

⟨*S*⟩ → OBJECT-NOM on COLOR-NOM
| OBJECT-NOM on COLOR-NOM väriltään
| OBJECT-NOM on väriltään COLOR-NOM
| OBJECT-GEN väri on COLOR-NOM
| OBJECT-NOM on COLOR-GEN värinen

⟨*OBJECT-NOM*⟩ → pallo

⟨*OBJECT-GEN*⟩ → pallon

⟨*COLOR-NOM*⟩ → punainen
| vihreä

⟨*COLOR-GEN*⟩ → punaisen
| vihreän

The point of this minimal grammar is simply to demonstrate that a more refined grammar formalism is required. If a CFG would be used, the Finnish

grammar would require a much larger set of productions, since sub-constituents occur in many more forms in Finnish than in English. A production would be required for each surface form, which would require a lot of redundant work that can be done more efficiently with other formalisms. It would also be difficult to map the much smaller English grammar to the Finnish one, which would make it difficult to compare the structure of an English sentence to the structure of the Finnish sentence.

Hand-crafted grammars in linguistics and computational linguistics are usually unification-based grammars (Shieber, 1986) which utilize complex feature structures in rules instead of the simple strings used by CFG rules. Linguistic phenomena are implemented as a process of unifying the feature structures of the relevant words, e.g. the subject and the verb in the case of subject-verb agreement. There are several unification-based frameworks available, out of which LinGo Grammar Matrix (Bender et al., 2002) and ParGram (Butt et al., 2004) are especially relevant to EvalGen, since they are designed with cross-linguistic grammars in mind. However, the aim of most unification-based formalisms is to provide a complete description of natural languages based on specific linguistic theories, which makes them a bad fit for projects motivated mainly by practical concerns, such as EvalGen. That is why a more flexible formalism, Grammatical Framework, is used instead.

5.1 Grammatical Framework

Grammatical Framework (Ranta, 2011) (GF) is a grammar framework, which is extremely flexible and not committed to any particular linguistic theory. In fact, the core of GF is simply a functional programming language designed for implementing multilingual grammars. GF also includes a resource grammar library, which contains definitions of syntax, morphology and lexicon for over twenty languages. These factors make GF the most suitable off-the-shelf grammar framework for an application such as EvalGen.

5.1.1 Abstract and concrete grammars

The main difference between GF and other grammar formalisms used in natural language processing is the division of grammars into abstract and concrete components, which is more commonly used in programming language compilers. The abstract syntax defines how the grammar can form syntax trees: it defines the categories of the grammar and the functions which construct the possible values of those categories. Concrete grammars map the syntax

trees constructed with the abstract grammar into string representations (a process which is called linearization in GF). Each abstract grammar can have an unlimited amount of concrete grammars, which provides a natural way of implementing multilingual grammars.

The most intuitive interpretation for abstract and concrete grammars is to see the abstract grammar as defining the semantics of the languages defined by the concrete grammars, while the concrete grammars define the syntax, morphology and lexicon of the individual languages. This kind of interlingual interpretation is central to many applications of GF, but the formalism itself does not impose any such interpretation: the grammars just define a mapping from a syntax tree to strings, and any interpretations of their meaning are external to the formalism.

The minimal CFGs in the previous section can be adapted as a GF grammar with a semantically motivated abstract grammar in the following way:

Abstract grammar

```
abstract BallSimple = {
    flags startcat = SemanticTemplate;
    cat
        SemanticTemplate;
        Object;
        Color;

    fun
        HasColor : Object -> Color -> SemanticTemplate;
        Ball : Object;
        Red, Green : Color;
}
```

The `cat` section of the abstract grammar defines the three categories used in the grammar: semantic templates, object and colors. `fun` section defines the `HasColor` function used to build semantic templates out of `Object` and `Color` values, one atomic value for the `Object` category (`Ball`), and two atomic values for the `Color` category (`Green` and `Red`). The `startcat = SemanticFrame` flag indicates that the grammar will attempt to parse and generate semantic frames by default.

English concrete grammar

```
concrete BallSimpleEng of BallSimple = open Prelude in {
    lincat
        SemanticTemplate = Str;
        Object = Str;
```

```
        Color = Str;

    lin
        HasColor object color =
            variants
            {
                "The" ++ object ++ "is" ++ color;
                "The color of the" ++ object ++ "is" ++ color;
                "The" ++ object ++ "is" ++ color ++ "in color";
                "The" ++ object ++ "'s color is" ++ color;
                "The" ++ object ++ "has a" ++ color ++ "color"
            };
        Ball = "ball";
        Red = "red";
        Green = "green";
}
```

The English concrete grammar is a simple mapping from the abstract grammar into strings. For each definition in abstract grammar's `cat` section, there is a corresponding definition in the concrete grammar's `lincat` section (short for "linearization category"). Since the concrete grammar maps the syntax trees to strings, it may seem redundant to have type definitions for the linearizations. However, generally only the top category (defined with the `startcat` flag in the abstract grammar) needs to be convertible into a string. The constituent categories may have more complex linearization types.

The `lin` section contains the actual definitions of the strings that different abstract grammar values are converted into. For the English grammar, simple string concatenation is sufficient. Note that the linearizations of the `HasColor` function are inside a `variants` block. The `variants` keyword can be used to define multiple different linearizations for a single category. In this grammar, it is used to define the different constructions which can convey the meaning of the semantic frame in English. This feature of GF would appear to be ideal for paraphrasing, but there are some performance issues related to it and it is problematic from the point of view of translation analysis. Because of these issues, EvalGen primarily uses an alternative paraphrasing method (introduced section 5.2).

Finnish concrete grammar

```
concrete BallSimpleFin of BallSimple = open Prelude in {
    lincat
        SemanticTemplate = Str;
        Object = {nom : Str;gen : Str};
```

```
        Color = {nom : Str;gen : Str};

    lin
        HasColor object color =
            variants
            {
                object.nom ++ "on" ++ color.nom;
                object.nom ++ "on" ++ color.nom ++ "väriltään";
                object.nom ++ "on väriltään" ++ color.nom;
                object.gen ++ "väri on" ++ color.nom;
                object.nom ++ "on" ++ color.gen ++ "värinen";
            };
        Ball = {nom = "pallo";gen = "pallon"};
        Red = {nom = "punainen";gen = "punaisen"};
        Green = {nom = "vihreä";gen = "vihreän"};
}
```

The Finnish concrete grammar is largely similar to the English one, but there are some changes due to Finnish morphology: the linearization type of the `Object` and `Color` categories is no longer a string, but instead a complex type with two labeled string fields (`nom` for nominative and `gen` for genitive), and the `HasColor` variants now specify which string field of `Object` and `Color` should be used in linearization. The complex linearization types are called record types in GF, and in addition to standard types they may also contain finite function types (called tables), which allow the implementation of more complex linguistic features. The record fields can be accessed in linearization definitions with the dot notation (e.g. `color.gen` above). Unification-like operations can be easily implemented using record types.

The simple GF grammar described above demonstrates how easy it is to create grammars for narrow domains with GF. Furthermore, GF grammars enforce a strict separation between the language-independent and language-specific parts of the grammar, which is ideal from the point of view of EvalGen.

5.1.2 GF Resource Grammar Library

GF includes a component called Resource Grammar Library (RGL) (Ranta, 2009), which considerably simplifies the creation of multilingual grammars. The RGL is in itself a GF grammar, with an abstract grammar and concrete grammars for over twenty languages. The abstract grammar of the RGL defines cross-linguistic syntactic trees instead of semantically motivated trees.

The RGL abstract grammar contains categories for linguistic universals such as nouns, verbs, adjectives, conjunctions, clauses, tenses and sentences, and functions for forming values of these universal types in various ways. For instance, the function `PredVP` (standard NP/VP predication) can be used to form a value of category `Cl` (clause) when provided values of categories `VP` and `NP` as arguments.

As an example of the structure of the RGL, its `lincat` definitions for the abstract grammar categories `CN` (common noun) and `NP` (noun phrase) are given below for English and Finnish:

English

```
NP = {s : NPCase => Str ; a : Agr}
CN = {s : Number => Case => Str ; g : Gender}
```

Finnish

```
NP = {s : NPForm => Str ; a : Agr ; isPron : Bool ; isNeg : Bool}
CN = {s : NForm => Str ; h : Harmony}
```

The types with the "=>" operator are the previously mentioned finite functions (tables). They are here used to create a noun paradigm, which can be accessed for unification purposes when using the values of these types in linearization. Both of the languages have a finite function called `s` fulfilling the role of the noun paradigm (note that the CN noun paradigm includes number but the NP does not, as number is fixed in NPs) and an `Agr` (agreement) field for subject-verb agreement, but they diverge in other respects.

The English CN type includes gender, which is used in reflexive expressions ("himself", "herself", "itself" etc.), while the Finnish CN type includes harmony, which is used to enforce vowel harmony. The Finnish NP type also has two boolean values, which indicate whether the NP is a pronoun or a negative NP. These values are used to select correct constructions when the NP is embedded in some other type. Note that the most important difference between English and Finnish, the number of cases (2 for English, 11 for Finnish), is not directly visible here, since they are encoded into the finite function argument types NForm, NPForm, Case and NPCase, which are defined elsewhere.

Values of the type CN can be used to construct values of type NP with many functions. One of the simplest of these functions is MassNP (mass noun phrase), which creates a determinerless noun phrase used in sentences such as "bread is good". RGL linearizes mass noun phrases in the following ways in English and Finnish:

English

```
MassNP cn = {
      s = \\c => cn.s ! Sg ! npcase2case c ;
      a = agrP3 Sg
      } ;
```

In English, making a mass noun phrase out of a common noun is simply a question of restricting the noun paradigm to singular forms only (mass noun phrases can only be singular), and fixing the agreement to third person singular. The `\\c => ...` expression is lambda-like notation for defining finite functions.

Finnish

```
MassNP cn =
      let
        n : Number = Sg ;
        ncase : Case -> NForm = \c -> NCase n c ;
      in {
        s = \\c => let k = npform2case n c in
                   cn.s ! ncase k ;
        a = agrP3 Sg ;
        isPron = False ; isNeg = False
      } ;
```

The Finnish MassNP linearization is essentially the same as for English, i.e. it restricts the noun paradigm to singular forms and sets agreement to third person singular, but it also defines the `isPron` and `isNeg` values as false, as they can never be true for mass noun phrases.

What is interesting about this example is that demonstrates how the RGL can make wide cross-linguistic generalizations while still accounting for the very different ways in which different languages realize those generalizations. Here the common parts are the categories CN and NP, the fixing of NP number and agreement when converting a CN into an NP, and the definition of a noun paradigm as a finite function from a number and case (or a complex type containing them, such as NForm here) to strings. However, the processes which eventually convert those parts into surface realizations differ considerably.

This example also raises a common issue with the RGL, which is to do with the separation of syntax and semantics in a cross-lingual representation. The RGL abstract grammar is syntactically motivated, and mass noun phrases have a clear-cut syntactic role in English as determinerless noun phrases.

However, in Finnish mass noun phrases are not marked in any explicit way, since Finnish does not have obligatory determiners. For Finnish, mass noun phrases are more of a semantic concept, and identifying a noun phrase as a mass noun phrase often requires contextual information, as it cannot be done strictly on the basis of syntactic clues.

In addition to the primary abstract grammar (core grammar), the RGL also contains additional abstract grammars (Extra grammars), which cover syntactic structures that are not universal enough to qualify for inclusion into the core grammar. These may be quite central structures for many languages, such as the genitive form of a noun phrase, which is excluded from the core grammar, as it is missing from e.g. French. The selection of which syntactic structures are included in the core grammar is by necessity somewhat arbitrary (especially as the set of universal features changes as languages are incrementally added to the RGL), so many common structures have to be accessed through the Extra grammars.

5.1.3 Using the RGL in application grammars

The RGL is interesting in its own right as a fairly complete and wide-ranging catalog of cross-linguistic similarity and divergence, but its primary purpose is to facilitate the creation of multilingual grammars for other purposes. The grammars built on top of the resource grammar for a particular application are called application grammars in GF terminology. The RGL is used in application grammars via a feature called grammar composition.

With grammar composition, an abstract category of grammar A can be assigned as a linearization category of an abstract category in grammar B. Similarly, functions in the abstract grammar A can be assigned as linearization functions for abstract functions of grammar B. When grammar composition is used, the abstract category or function in grammar B assumes the same linearization category or function that has been defined for the assigned category or function in the concrete grammar of grammar A.

Grammar composition means that semantic structures in an application grammar can be mapped into the categories of the RGL in such a way that they can be used to form valid RGL trees that can be linearized into grammatical strings. So the surface representations of an application grammar can be defined simply by mapping its categories to the categories of the RGL, which requires very little linguistic expertise besides basic knowledge of grammatical categories.

The concrete grammars of the example grammar from the previous section

can be implemented with the RGL and grammar composition in the following way:

English concrete grammar

```
concrete BallSimpleRGLEng of BallSimpleRGL =
open Prelude,LangEng,ParadigmsEng,ExtraEng in {
  lincat
    SemanticTemplate = S;
    Object = NP;
    Color = AP;
  lin
    HasColor object color =
      let
        cl : Cl = variants
        {
          PredVP object (UseComp (CompAP color));
          PredVP
            DetCN
            (
              (DetQuant (GenNP object) NumSg)
              color_kind
            )
            (UseComp (CompAP color))
        };
      in
        UttS (UseCl (TTAnt TPres ASimul) PPos cl);

    Ball = DetCN
      (DetQuant DefArt NumSg)
      (UseN (mkN "ball"));
    Red = PositA (mkA "red");
    Green = PositA (mkA "green");
  oper
    color_kind : CN = UseN (mkN "color");
}
```

The pragmatically motivated `lincat` definitions of the previous English concrete grammar have been replaced by linguistic categories, which are fully defined in the RGL concrete grammars (`S`, `NP`, `AP`). The linearization function of `HasColor` has also been implemented through grammar composition, and the linearization is created by building RGL abstact grammar trees instead of concatenating strings (all the capitalized functions used in the linearization definitions are RGL abstract grammar functions).

This grammar also demonstrates two central features in GF, `oper` (operation) definitions and RGL API operations (functions of the form `mk[RGL category]`, such as `mkN` and `mkA`).

Operations are the fourth type of function available in GF, the other three are the previously covered abstract syntax functions (`fun` definitions), linearization functions (`lin` definitions) and finite functions (tables). Operations are the most versatile of these function types, and they are mainly used as auxiliary functions in linearization function definitions for the purposes of code reuse: if the same functionality is duplicated in multiple linearization functions, the functionality can be separated into an operation, which can be used in any linearization function.

In this simple grammar the only defined operation is `color_kind`, which is a constant function returning the common noun "color", which is used in a variant of the `HasColor` linearization. The operation is only used in one linearization, so it could have been used inline, but there are other benefits to separating language-specific parts of linearizations into operations: with the `color_kind` operation, the `HasColor` definition is entirely language-independent, and it can be used as it is in the concrete grammars of other languages. In fact, the English grammar can be converted into a Finnish grammar just by substituting a few strings with their Finnish equivalents:

Finnish concrete grammar

```
    Ball = DetCN
      (DetQuant DefArt NumSg)
      (UseN (mkN "pallo"));
    Red = PositA (mkA "punainen");
    Green = PositA (mkA "vihreä");
  oper
    color_kind : CN = UseN (mkN "väri");
```

(This kind of reuse can be partly automated by using functors, but that is out of the scope of this thesis.)

5.1.4 RGL API

Using the RGL directly in the manner shown in the previous section is fairly complicated, as it requires searching for correct functions and constructing trees with multiple levels of linguistic description. As was mentioned, the RGL includes an API, which makes it simpler to use RGL in application grammars. The RGL API functions are concrete grammar operations, some

of which are constant and some of which have arguments. The constant functions generally have a descriptive name, such as `and_Conj` for the conjunction "and". The functions that take arguments usually have the form `mk[RGL category]`, and they are used to construct values of the specified category from the values provided as arguments.

Another distinction in the API is between cross-lingual syntactic and structural functions, and language-specific rules, which are used for non-universal structures and for generating values of lexical categories, such as `N`, `A` and `V`. The cross-lingual API functions are mostly wrappers for the RGL abstract syntax trees (grammar composition is used), while the language-specific functions construct values of record types defined in the concrete grammars.

For instance, the cross-lingual API function `mkS` with the type signature `Cl -> S` (constructs a value of category `S`, i.e. sentence, from a value of category `Cl`, i.e. clause) is defined like this:

```
mkS : Cl -> S = UseCl (TTAnt TPres ASimul) PPos
```

`UseCl` is an RGL function that takes three arguments: temporal and aspectual features (`Temp`), polarity (`Pol`) and a clause (`Cl`). `mkS` is defined as partial application of UseCl, with `Temp` and `Pol` fixed to their most probable values (present simple).

Multiple operations with the same name but different type signature can be defined in GF by using the `overload` keyword. This feature is extensively used in the RGL API, with different overloads wrapping different RGL syntax trees or operations. For instance, the `mkS` function has overloads for constructing a sentence with arguments for tense and aspect, and an overload for constructing a conjoined sentence from a conjunction and two sentences:

```
mkS : Temp -> Pol -> Cl -> S = UseCl
mkS : Conj -> S -> S -> S = \c,x,y -> ConjS c (BaseS x y)
```

The benefit of using overloaded operations instead of straight RGL abstract functions names is obvious: finding the correct function is much easier, and shortcuts can be made to commonly used RGL syntax trees by using partial application. The linearization of the HasColor function from the example grammar is much more concise and easier to read when implemented with the RGL API:

```
HasColor object color =
  let
    cl : Cl = variants
    {
```

```
      mkCl object color;
      mkCl (mkNP (GenNP object) color_kind) color;
    };
in
  mkS cl;
```

(Note that `GenNP` function cannot be replaced with an API function, since genitive formation was not part of the RGL core grammar.)

5.2 Paraphrasing in GF

As mentioned, the `variants` feature of the concrete grammars would seem like a natural way of implementing paraphrasing in GF. This was the method initially chosen for EvalGen, but it was abandoned for several reasons. The most important was the practical reason that implementing widescale `variants` paraphrasing on many levels of the syntax trees deteriorated the compilation speed of the grammars significantly.

With the `variants` method, all the variants would also be parsed as the same semantically-motivated syntax tree, which would mean that cross-lingual comparisons of constructions would be much harder. For instance, it would be useful to automatically detect whether the translation retains the type of construction used in the source language. Related constructions are not always syntactically identical cross-lingually, so the similarity relation has to be explicitly defined, which is something that's not possible with the `variants` method.

An example of a construction requiring explicit cross-lingual similarity definition is the existential encoding for meteorological phenomena, e.g. "there is rain" / "on sadetta". There are no simple morphological or syntactic similarities here that would detect the similarity between the pair, but semantically it is obvious.

In order to detect this kind of semantic constructional similarity, the constructions need to be defined in the abstract syntax, so that the similarity can be checked by parsing sentences into syntax trees. This means that the abstract grammar can no longer be naively semantics-based, like the example grammar in the previous section. The semantic template becomes a set of cross-lingual constructions representing different information structures that can be used to convey the meaning. The concrete syntax will then simply convert the cross-lingual constructions into strings with very little paraphrasing (an exception can be made for near-synonyms, spelling variants etc.).

This kind of abstract grammar is half-way between purely semantically motivated application grammars and the syntactically motivated resource grammars.

5.2.1 Defining construction equivalence

If the paraphrasing is moved from the concrete syntax into the abstract syntax, the relation between constructions with similar meanings has to be made explicit in some way.

For practical reasons, the construction equivalence is defined in an adhoc fashion. Each semantic template has its own grammar, and all functions returning a value of a particular category are considered to be paraphrases of each other. In the example grammar, this would entail splitting the `HasColor` function into several different functions with the same type signature (`Object -> Color -> Sentence`), one for each construction type found in English and/or Finnish:

Function	English	Finnish
`XIsY`	the ball is red	pallo on punainen
`AdvPossXIsY`	the color of the ball is red	
`XGenColorIsY`	the ball's color is red	pallon väri on punainen
`XIsYAdvColor`	the ball is red in color	pallo on punainen väriltään
`XIsAdvColorY`		pallo on väriltään punainen
`XIsYGenColor`		pallo on punaisen värinen
`XHasYColor`	the car has a red color	

The selection and granularity of the constructions are pragmatically determined. The purpose of defining the constructions is to allow the analysis of how MT systems transfer different constructions from one language to another. This is why it is important to make distinctions that reveal structural differences between languages, such as the distinction between AdvPossXIsY and XGenColorIsY. Semantically these are almost identical: both express a possession-like relationship, but the adverbial form is only available in English. Similarly, XIsYAdvColor and XIsAdvColorY are near-identical in meaning, but they are separated into two functions because Finnish allows the movement of the adverb.

5.2.2 Implementing semantic variables in GF

Semantic variables have been omitted from the description above for purposes of clarity. They are implemented through a naming convention: semantic variables are indicated by prefixing the abstract grammar function name with a *SemVar* code, which consists of the string "SemVar", a digit identifier for distinguishing different semantic variables in the grammar, and a letter identifier for distinguishing different paraphrase sets of the same semantic variable. In the example grammar, the `Object` and `Color` are good candidates for semantic variables:

```
SemVar1aBall : Object;
SemVar2aRed : Color;
SemVar2bGreen : Color;
```

Semantic variable `SemVar1` has one semantic value (`SemVar1aBall`), while `SemVar2` has two (`SemVar2aRed` and `SemVar2bGreen`). Each value of the semantic variable consists of just one equivalent function, so the same SemVar code is not duplicated. As an example of a SemVar value with multiple functions, let's add the functions corresponding to the phrases "the ball on the table" and "the ball which is on the table":

```
SemVar1bBallOnTable : Object;
SemVar1bBallRelativeOnTable : Object;
```

The two functions share the SemVarCode, and therefore they are treated as paraphrases in the EvalGen processing pipeline.

Note that this is a simplified example grammar, in the actual EvalGen grammar the type signatures are more complex to allow for more variation. For instance, the actual EvalGen functions corresponding to the example have the following type signature:

```
BaseObject -> Location -> Relation -> Object;
```

The arguments in this type signature allow for the variation of the object ("ball", "cube"), the location on which it is placed ("table", "desk", "chair" etc.) , and the relative location of the object to the location ("on", "under" etc.).

5.3 Designing the semantic templates

Because the semantic templates are designed pragmatically based on the cross-lingual properties of SL and TL, the design should ideally be performed by persons fluent in both languages, as monolingual designers would not be

able to identify cross-lingual features. The actual process of designing the grammars is by necessity based on introspection, but the results of introspection should be verified by corpus searches, unless the construction is obviously valid.

5.3.1 Paraphrase set coverage

The design of the surface representations requires extensive knowledge of the languages, as the usefulness of EvalGen depends largely on the coverage and correctness of the paraphrase sets. Designing the target language surface representations requires native-level expertise of the target language, and also extensive knowledge about the conventions of the standard written form of the target language.

For the source language surface representations, native-level expertise is not required, since complete coverage is not necessary: more variation is always better, as it provides more data points for the evaluation, but the correctness of the results is not affected by low coverage, unlike with the target language. In the case of English, it might be argued that a non-native English speaker with a good command of written English can perform the task just as well as a native English-speaker: a large portion of English text is produced by such non-native speakers, so their introspective judgements may be more representative of the texts that would generally be translated from English into other languages.

5.3.2 Design workflow

The natural first step of the semantic template design is to come up with a single sentence or a small sentence group that appears to have some interesting cross-lingual properties, such as the meteorological sentence "it is raining" discussed earlier. This is followed by an introspection phase where the meaning of the selected sentence is examined for possible ambiguities, and the designer explores the paraphrasability of the sentence by coming up with different ways of expressing the same meaning.

If the sentence is found ambiguous, the designer checks whether the sense can be fixed to resolve the ambiguity. For instance, the Finnish translation for the bare meteorological sentence "it is raining" ("sataa" in Finnish) is arguably ambiguous between habitual and progressive senses ("it is raining now" and "it rains sometimes"), but the ambiguity can be resolved by adding an adverb expressing time, as in "it is raining now".

Another step is trying out different semantic variables with the semantic template, to see how easily it can be expanded to other contexts. In mete-

orological sentences the most natural semantic variables are duration, time, and place. Other possible sources of variation are intensity ("heavy rain") and different tenses. Note that the present tense is actually quite inconvenient, since it is incompatible with many expressions of time and duration ("last year"/"three weeks"), and it is ambiguous between present and future. That is why it makes more sense to use past tenses in the semantic template, as they can be varied quite freely ("it had rained"/"it rained"/"it has rained").

The result of this introspection phase should be lists of sample paraphrases for both languages, which can then be used to define the abstract grammar based on the cross-lingual similarities between the constructions used in the two languages. After that the concrete grammars are built using the lists of sample paraphrases as a guide. The grammar can be validated by testing whether it parses the list of sample paraphrases.

5.4 Generating the paraphrases

All the paraphrase sentence sets can be generated by using standard GF functionalities. The GF grammars are first compiled into GF's Portable Grammar Format (PGF) files, which can then be accessed by the GF runtime, which has bindings for many programming languages. EvalGen is primarily developed in Python, so the Python bindings for the GF C runtime are used. Generating all the sentences defined by the grammar takes some time, but not excessively, and in any case the generation step has to be done only once.

When generating the paraphrases, the different semantic templates and all the possible permutations of semantic variables need to be kept track of. The following steps are performed for each semantic template (each with its own grammar):

1. All possible abstract syntax trees are generated.

2. Generated syntax trees are divided into groups based on what set of semantic variable prefixes they contain.

3. Source and target paraphrase sentences are generated for each group of semantic variable prefix sets and stored separately.

The generation results in separately stored paraphrases for both source and target language for each semantic template. The paraphrase sentences of each semantic template are further subdivided into groups based on semantic variable sets. The example grammar contains two semantic variables

(SemVar1 for Object, SemVar2 for Color) each with two values, so the paraphrase generation produces four semantic variable sets. The English surface representations of those sets are partially listed in the table below:

	1a (ball)	1b (ball on the table, ball which is on the table)
2a (red)	the ball is red	the ball on the table is red
	the color of the ball is red	the color of the ball on the table is red
	...	...
2b (green)	...	...
	the ball's color is green	the ball which is on the table is green
	the ball is green in color	the ball which is on the table is green in color
	the ball has a green color	the ball on the table has a green color

Complete sets of paraphrase sentences are generated for both languages, but the extent of the sets and the storage methods are different for the two languages. The source language paraphrase sentences are generally fewer in number, as they do not need to cover all possible paraphrases, but there's still generally a very large number of paraphrases. It is redundant to translate and evaluate a very large number of slightly different translations, which is why only a subset of the source sentences are utilized in the evaluation (although using the whole set might be useful for detecting whether small changes in the source text trigger large changes in the translation etc.).

5.4.1 Selecting source sentences

The subset of source sentences is selected on the basis of the dissimilarity of the sentences. The set of all sentences is first shuffled, and then the first sentence is added to the set. For each subsequent sentence, edit distances between the sentence and all the sentences already selected for the subset are calculated. If the sentence has a length-normalized edit distance below a threshold with any sentence in the subset, the sentence will not be added to the subset. The edit distance checks are computationally somewhat expensive (the edit distance function from the NLTK toolkit is used), but since they are performed only once during preprocessing, optimizing them is not

a priority. This naive method very likely will not find the most dissimilar subset possible, but it will still guarantee a sufficiently dissimilar subset of source sentences.

5.4.2 Storing target sentences

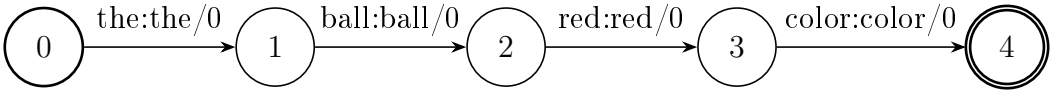
For the target sentences, all the sentences need to be recorded, as the translations to be evaluated will be compared against all of them. As the coverage of the target set is more complete, the number of sentences may be very high. Because of this, the sentences will added one by one to a trie as they are generated. The trie is stored in the OpenFST text format, which can be directly used in the evaluation phase of EvalGen.

6 Evaluation and error analysis methods

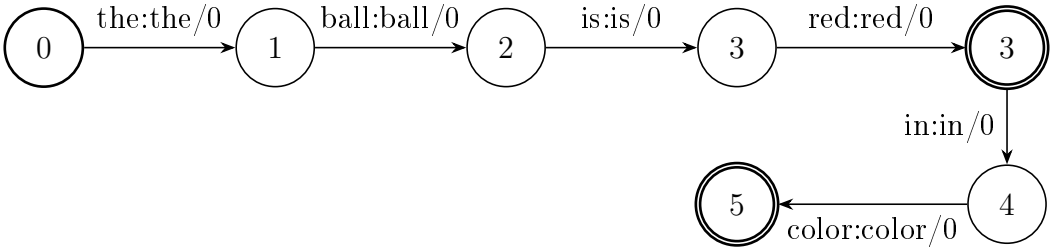
The semantic templates are used to generate a large amount of reference translations for each source sentence. Still, a large of portion (usually most) of machine translations will not match any reference exactly, usually because of an error in the translation. A method to partially match translations to reference translations is therefore required. The simplest method would be to use any existing automatic multireference metric, as was done in Bojar et al. (2013), where BLEU was used. As mentioned, HyTER encodes the references as non-recursive RTNs for which equivalent finite-state machines (FSM) exist. Converting these RTNs into equivalent FSMs makes it possible to efficiently calculate edit distances from the translation hypothesis to the closest reference translation by using weighted finite-state transducer (FST) composition (Mohri, 2003). This method has also been recently used with automatically generated paraphrase networks (Apidianaki et al., 2018).

The edit distance calculation using FST composition is performed as follows:

First, the translation hypothesis is converted into a finite-state acceptor, i.e. an FSM that accepts the translation hypothesis. For purposes of weighted FST composition, the acceptor is represented as an FST with identical input and output symbols, with each transition having a weight of 0. As an example, let's assume the source sentence is the meaning equivalent of "the ball is red" in some language, and we are translating into English. Let's also assume an incorrect translation hypothesis "the ball red color" ,which has the following FST representation:



The set of reference translations also needs to be in the form of an acceptor. In EvalGen, the reference translations are natively stored as FSMs, so no conversion is required. Let's assume we have the reference translations "the ball is red" and "the ball is red in color", which are represented by the following FST (note the two final states):



The actual edit distance calculation is achieved through the use of an edit transducer, which contains transitions for transducing each symbol in the translation hypothesis alphabet into any symbol in the reference translation alphabet. Both alphabets include the epsilon symbol, which consumes no input or produces no output and can be used to implement insertion and deletion edits. The basic edit transducer has only one state in which all the transitions start and end (although it can be expanded with edits that require the use of multiple states). Figure 2 shows a partial diagram of an edit transducer.

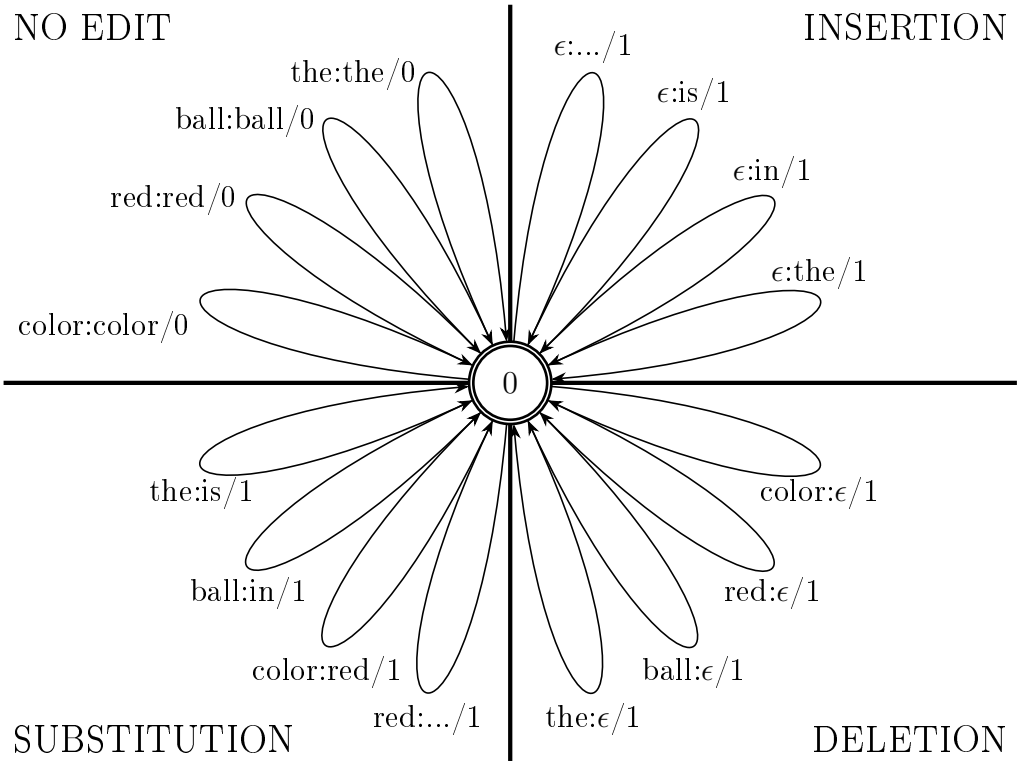
The diagram is partial, as the full diagram would contain insertion transitions for each symbol in the reference alphabet and substitution transitions for each pair of hypothesis and reference alphabets' non-epsilon symbols.

The three transducers (H for hypothesis, E for edit and R for reference) need to be composed in order to calculate the edit distance. For edit distance calculation, the transducers are defined over the tropical semiring, which means that composition for weighted transducers $T1$ and $T2$ is defined as follows:

$$[T_1 \circ T_2](x, y) = \min_z(T_1(x, z) + T_2(z, y)) \tag{8}$$

This definition states that for any output string z in $T1$ which is accepted by $T2$, the weight of the output string y for the input string x in the composed transducer is the sum of the weights of output string z for input string x in $T1$ and of output string y for input string z in $T2$. There can be multiple

Figure 2: Edit transducer



paths of different weights for transducing x to z in $T1$ or for transducing z to y in $T2$, so the paths with the lowest weights are used for calculating the composed weight.

Composition is associative, so H, E and R can be composed in two ways: $[[H \circ E] \circ R]$ or $[H \circ [E \circ R]$. Composition of H and E (HE) is a transducer that accepts the strings in H and outputs any string in the alphabet of R, where the weight assigned to output string is the edit distance from the input string to the output string. Composition of E and R (ER) is a transducer that accepts any input string in the alphabet of H and outputs any string in R, with the weight of the string being the edit distance from the input string to the output string. Composing HE with R constrains the output of the composition to strings in R, and composing H with ER constrains the input of the composition to strings in H. While both composition orders are possible, in practise composing HE makes more sense, since R stays the same across hypotheses while H and E both vary.

After the FSTs have been composed, the edit distance from the hypothesis to the closest reference can be calculated by searching the shortest path in the composed FST.

The FSM composition method has many attractive features: it is compact, efficient and simple. What makes it especially suitable for EvalGen is that it is easy to add new edit types and implement granular error analysis. This can be achieved by adding new paths consisting of one or more transitions to the edit transducer. EvalGen adds the following new edit types:

1. Lemma-preserving transitions: These transitions are created for hypothesis tokens if the reference network contains tokens which have the same lemma but different surface form. This achieves the same effect as lemmatization or lemma matches in other metrics.

2. Morphology-preserving transitions: Morphological transitions are created for each pair of hypothesis and reference tokens that share morphological features. The categories that are taken into account are nominal case and number, and verb tense, person and mode. These transitions make it possible to reward the MT for grammatical correctness.

3. Compound transitions: These transitions are created for hypothesis tokens that match components of compound words in the reference network. Compound word errors are common when translating from English to Finnish, as compounds are always concatenated in Finnish

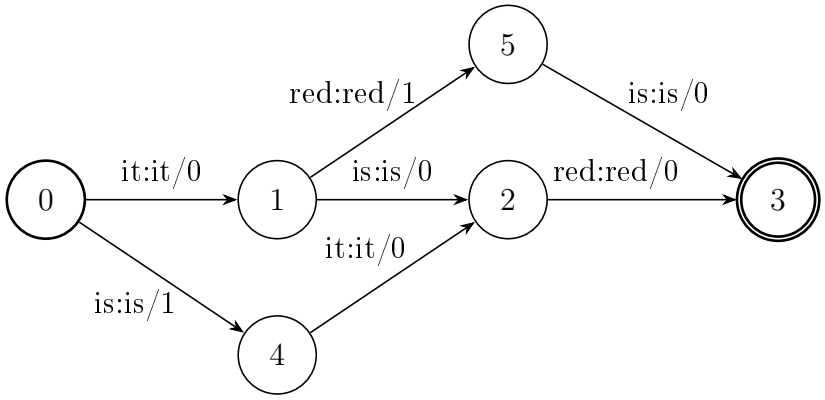
typography whilst in English they may be concatenated ("wallpaper"), hyphenated ("rent-seeking") or left as separate tokens ("car park").

6.1 Movement edits

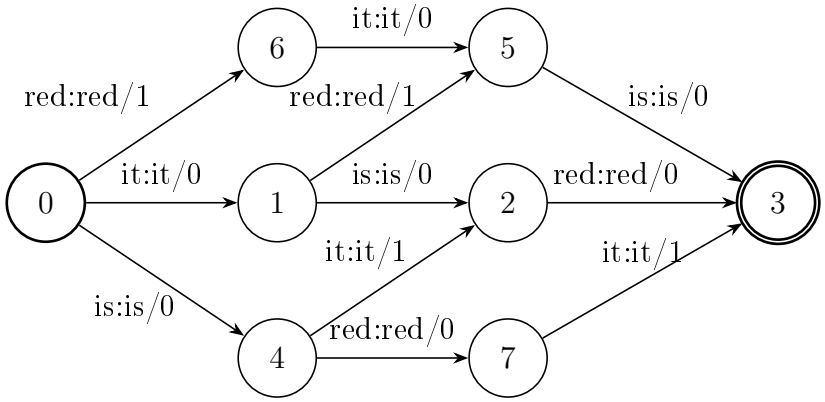
The edit distance method above is very much like WER, and it also resembles WER in not having an edit type for token movement. As was mentioned, this is problematic: errors in token order are very common in MT, and implementing them as a combination of deletion and insertion edits overstates their severity. As EvalGen is intended for error analysis, token movement is perhaps a larger problem, as the lack of a token movement edit type would mean misdiagnosing an entire error category.

In HyTER, this problem was addressed by allowing local reordering of tokens. The reordering was implemented by extending the hypothesis transducer with permutations of the original hypothesis. The permutations were restricted by allowing movement only within a fixed window, so that each token may be at most n places from its original position in the sentence (3 was discovered to be the most practical reordering limit). For each permutation, the transitions differing from the original order of the hypothesis were weighted with a token movement edit cost. With an extensive set of references, this approach probably solves most of the reordering issues, since the reference set will already contain many alternative word orders in the target language. EvalGen uses a similar approach but with limited long-distance movement.

In EvalGen, the movement restriction is not based only on a window around the token, but also on the total cost of the permutations of the hypothesis. So instead of allowing all permutations where tokens move at most x places from their original positions, the cost of all moves in the permutation must also be less than y. This means that it is possible to use much longer movement windows, as long as the overall amount of moves stays within the limit. To demonstrate this method, consider the three word hypothesis "it is red". With a local reordering window of 1 (and movement cost of 1), the hypothesis transducer would be as follows:



With a local reordering window of 2 and with total moved tokens within a permutation limited to 1, the transducer is slightly different:



Each path above has a cost of 1, except for the nonpermuted path with a cost of 0. Note that using a local reordering window of 2 without the total cost restriction would allow one path of cost 2, i.e. "red is it".

The amount of permutations allowed with long-distance reordering and a total cost restriction n grows much faster with sentence length than the amount of permutations allowed with local reordering window of the same n. Due to the short sentences used in EvalGen and the availability of different translations with different word orders, limited reordering is usually sufficient. A movement window of 6 with the restriction that the total amount of moves can be at most 3 seems to be a good compromise between coverage and scoring speed.

The justification for restricting the amount of total moves is that there's a limit to how permuted a translation can be in order for it to be considered incorrectly ordered rather than just generally incorrect. MTE is always ultimately based on human judgement, and humans are not very good at analyzing sentences as permutations of other sentences. For instance, Kopo-

nen (2012) indicates that edits requiring movement of tokens are cognitively more demanding than other edit types. Even with only local reordering, maximally permuted sentences appear nonsensical rather than correctable. For instance, consider the following permutations of "the quick brown fox jumps over the lazy dog":

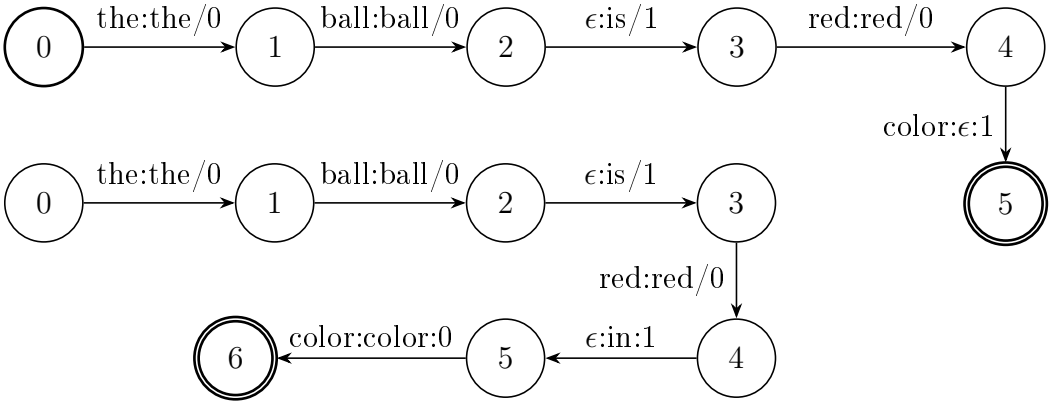
1. "quick fox the jumps brown over dog lazy the".
2. "brown the the over dog jumps fox lazy quick".

For purposes of MTE, it is not useful to be able to reorder these sentences into reference translations, as reordering would not be a optimal strategy for a human evaluator, who would probably find it most efficient to use substitution, deletion and insertion in a linear order starting from the left. Even local reordering produces many permutations (like the first sentence above) that would not be cognitively available to humans, which may make some edits appear considerably easier in terms of edit distance than they actually are for a human editor. Therefore it makes sense to restrict the amount of changes in permutations to an amount that is feasible for humans to process.

6.2 Selecting the edit weights

The standard WER weights are uniform, which is not ideal for MTE. This is apparent from the fact that optimal weights for tunable edit distance metrics, such as TERp, are far from uniform. For EvalGen there is no manual evaluation data available to tune the weights, so they have to be justified by other means. If EvalGen was used strictly for intrinsic evaluation, it might be best to directly adapt the TERp weights. However, as EvalGen is also intended for error analysis, the effects of edit weights on the plausibility of the error analysis should be considered.

Let's consider the translation hypothesis above, "the ball red color". Composing the hypothesis transductor with the edit and the reference transducers produces a transducer with two shortest paths:



These edits can be interpreted as members of error categories (Vilar et al., 2006) in a straight-forward manner:

Edited	Edits	Error category
Ball is red	insert "is", delete "color"	missing filler word, extra content word
Ball is red in color	insert "is", insert "in"	missing filler word, missing filler word

Missing filler words are generally considered to be lesser errors than extra words, as the filler words are not essential for communicating the meaning of the sentence. This demonstrates that using uniform weights can lead to incorrect error analyses. In this case it might lead to the developer to misdiagnose the MT system as producing spurious words, when a more valid diagnosis would be undergeneration of function words. In general, it would appear that preserving hypothesis tokens whenever possible would lead to paths that provide intuitively correct error analyses. Therefore it is justified to make deletion and substitution more expensive than token insertion and movement.

A further refinement for English error analysis could be to add less expensive insertion edits for function words. In the case of Finnish, most of the tasks of English function words are performed via inflection, so lemma-preserving transitions already serve a similar function.

Substitution is the most opaque edit type in terms of error analysis, so its cost should be increased to favour other, more informative types of edits.

6.3 Implementation of the evaluation functionality

The evaluation functionality is implemented as a set of Python scripts that utilize OpenFST (Allauzen et al., 2007) for FST processing and Omorfi (Pirinen, 2015) for morphological analysis.

6.3.1 Reference transducer

As mentioned, the target strings outputted by the GF grammars are directly stored as tries in OpenFST text format. The reference transducers can then be simply created by compiling the text FST with the OpenFST's *fstcompile* shell command. The compiled transducers are minimized with the *fstminimize* shell command, as they have a lot of redundancy in them (minimization reduces the size of the FST from megabytes to kilobytes).

6.3.2 Hypothesis transducer

Hypothesis transducers are generated from MT output, which is first tokenized and lower-cased. If no token movement is allowed, the transducer simply contains subsequent transitions for each token, otherwise the permutations required for token movement are generated with Python. In both cases, the FST is stored in the OpenFST text format and compiled with *fstcompile* and minimized with *fstminimize*.

6.3.3 Edit transducer

Constructing the edit transducers is the most complex part of the evaluation process. Initially, transitions are created for each pair of hypothesis and reference alphabet symbols, and the cost is assigned according to the nature of the transition. The costs below are chosen so that insertion is preferred to deletion and substitution is discouraged (it is only slightly less than the combined cost of the equivalent deletion and insertion pair):

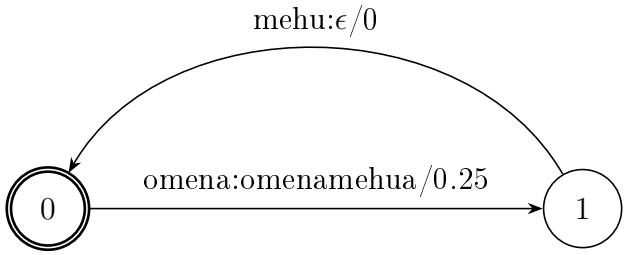
Symbols	Edit type	Cost
input = output	no edit	0
input $\neq$ output	substitution	1.5
input = ϵ	insertion	0.8
output = ϵ	deletion	0.9

Next step is creating morphological edits. All hypothesis and reference symbols are analyzed with Omorfi. The following edits are then created based on the properties of the analyses:

Analysis condition	Edit type	Cost
input lemma = output lemma	keep lemma	0.25
input (case + num) = output (case + num)	keep case + num	0.5
input case = output case	keep case	0.75

All of the morphological edits are of the form *a:b*, i.e. they are substitution edits with lower weights than normal substitution. Morphological edits supersede the identical normal substitutions which were made in the first phase. Identical substitutions in the edit transducer are later pruned, and only the edit with lowest cost will be included in the compiled transducer.

Last phase is the creation of compound edits. Compound words in the reference alphabet are identified with Omorfi, and a compound edit is created for each sequence of tokens in the hypothesis that has the same lemmas as the compound word in the reference alphabet. For instance, let's assume the reference alphabet contains the compound "omenamehua" (constituent lemmas "omena" and "mehu") and the hypothesis has the token sequence "omena" and "mehu". To allow editing of the hypothesis sequence to the reference compound, the following path is added to the edit transducer:



This demonstrates how edit transducers can be used to implement more complex edits than simple single symbol operations. Since the movement edits are implemented in the hypothesis transducer, it is also possible to reorder the tokens before converting them to compounds.

6.3.4 Composition and finding the shortest path

OpenFST is used to compose the transducers. Due to the complexity of the transducers, completely composing the hypothesis, edit and reference trans-

ducers requires considerable amounts of time and disk space. Since the composition has to be done each time a hypothesis is evaluated, lazy composition is used to reduce the time and space requirements. With lazy composition, the composed transducer is expanded only when relevant paths are actually used. Since the composed transducer is mainly used for finding the shortest path, most of the time the majority of its paths are not expanded.[4]

OpenFST's standard shortest path algorithm is used to find the shortest path through the composed FST. The shortest path is saved as an FST in the OpenFST text format. The transitions in the shortest path FST can be used to extract the edits that were required to convert the hypothesis into the closest reference translation. The edit types can be identified based on the weights of the transitions, as each edit type has a different weight. n shortest paths can be retrieved efficiently with a single search, which makes it possible to compare all close reference translations and check whether a different edit profile with a similar cost is available.

7 Experiments and evaluation of the method

7.1 Experiment setup

7.1.1 Test sets

For testing EvalGen as a method, a set of 7 semantic templates was prepared. The templates were designed in two phases. The first set of four templates was designed by searching the WMT17 test set for promising, simple sentence candidates to serve as the basis for the prototypical meaning of the templates. These sentences were then further simplified, and different ways of expressing the meanings of the constructions used in the prototypical meaning were introspectively explored for both English and Finnish. The original and variant expressions were then encoded as GF grammars. In cases where the prototypical meaning was discovered to be unsuitable for use as a semantic template, minor changes were made to the meaning (see 4.5.1 for one example of such a case).

The prototypical meanings of the phase 1 semantic templates are listed in figure 3.

The semantic templates created with this method turned out to be subopti-

[4]The code for composition and other custom OpenFST functionalities of EvalGen are based on Benoit Favre's openfst-utils tools (https://github.com/benob/openfst-utils)

Figure 3: Prototypical meanings of phase 1 semantic templates. Semantic variables are written in regex format. The first two figures in parentheses are the number of English and Finnish sentences generated, and the third figure is the amount of semantic variable sets for the template.

1. This was a massive and, at the same time, very precise operation. (2964/32808/3)
2. Rebuilding real trust will be (easy|hard|hard, perhaps impossible). (216/13500/3)
3. Despite the public display of reconciliation, the (two countries|Russia and Turkey|Russia and China|Russia, Turkey, and China...) still have major differences. (255150/8265600/15)
4. Those who exercise regularly reduce their chances of getting (cancer|cancer and depression|...). (22032/74504430/6)

mal in multiple ways. The sentences 1 and 2 above were at the same time too simple and cross-linguistically unstable to be good starting points. For instance, the word "operation" in sentence 1 has a wider sense in English than in Finnish, where it is mostly used with military and medical operations. Both sentences were also rather easy for MT systems, and they are preserved in the set to provide an easy challenge for MT.

Sentences 3 and 4 are more complex, and perhaps too complex, since accounting for all the valid variants proved difficult (especially problematic was the multitude of expressions possible for the chance of getting a medical condition, since it can be expressed in terms of chance, possibility, danger, probability etc.). These semantic templates are however still very good at eliciting MT errors, and they have natural slots for semantic variables to provide variety.

The semantic templates of phase 2 were created on the basis of the experience gained from phase 1. Instead of using in-the-wild sentences as prototypes, the semantic templates were built around simple semantic propositions, which had some interesting cross-linguistic properties and potential semantic variables as constituents. The prototypical meanings of the templates are listed in figure 4.

Each of the phase 2 semantic templates represents a different approach to

Figure 4: Prototypical meanings of phase 2 semantic templates.

5. The (bottle|cup|mug|plate) (on the dinner table|on the silver tray)? is (red|yellow|...). (7872/113760/72)
6. It (has|had) snowed (heavily)? in (the Alps|Lapland|...) for (several|two|...) (hours|days|weeks). (11178/10812960/108)
7. We wanted to buy (a can of soda|cassette player|...) from the store, but they had none for sale. (2520/14878080/9)

semantic template creation. The first is a semantic proposition describing a simple physical state of affairs, with three semantic variables. The simplicity of the proposition makes paraphrase creation much easier, as the effects of ambiguity are much clearer. Note the sense fixing in alternatives "dinner table" and "silver tray", which eliminates the ambiguities in "table" and "tray".

The second template represents the approach where the semantic template is selected because its surface expressions are cross-linguistically very different. The third template is a fairly complex proposition, but one in which ambiguity is reduced through the use of plentiful context. The pronoun reference in the subclause is linguistically interesting, as are different expressions for not having something for sale, since they involve negation and pronouns.

The first phase templates use GF's `variants` functionality for paraphrasing, while the second phase templates use semantically equivalent syntactic paraphrases in the abstract grammar (some `variants` paraphrases are still used for lexical synonyms).

Two test sets are generated from the semantic templates, one with 100 sentences (Testset100) and the other with 500 sentences (Testset500). The shorter, more heterogenuous Testset100 is used as source text for producing human translations for manual evaluation, while Testset500 is used for tests requiring more data points.

7.1.2 Manual evaluation participants

EvalGen is manually evaluated by two evaluators (evaluators 1 and 2), both of who are experienced translation professionals, with English and Finnish

Identifier	Description
SMT-1	Moses-based SMT system
SMT-2	Online SMT system
NMT-1	Marian-based NMT system
NMT-2	Online NMT system
MT-1	Online system with unknown implementation
MT-2	Online system with unknown implementation
RBMT-1	RBMT system
RBMT-2	RBMT system

Table 2: MT systems evaluated

as their working languages. Evaluator 1 is the designer of EvalGen and the author of this thesis, which means that the evaluator is familiar with the type of output that the system expects. Evaluator 1's results are therefore not reliable indicators of the real performance of the system. Evaluator 2's results are only included as a point of comparison for the results of Evaluator 2, who has no prior knowledge of EvalGen.

7.1.3 Evaluated MT systems

Because the intent of EvalGen is to provide more granular insight into the workings of MT systems, a mix of systems using different MT paradigms is evaluated. In addition to easily available SMT and NMT systems, the author is privileged to have access to two mature English to Finnish RBMT systems[5]. (It should be noted that the main designer of one of these systems rejects the use of "rule-based" as a descriptor of their MT system, as the rules are incidental to the design objective of the system, which is to model the linguistic competence of a human translator. They prefer to characterize their system as "Machine Translation Powered by Artificial Intelligence".)

The design details of most of the tested systems are unknown, and they are in any case not directly relevant to this thesis, so the MT systems are anonymized and only basic information is provided about them. The systems included in the evaluation are listed in Table 2.

[5]I would like thank the designers of both these systems for allowing their use in evaluation.

7.2 Error analysis

EvalGen can be used for error analysis by extracting the edits in the shortest edit path from the hypothesis to a reference, and mapping those edits to error types. The counts of different errors are then used to diagnose problems in the MT system.

7.2.1 Nondeterminism of corrections with high edit counts

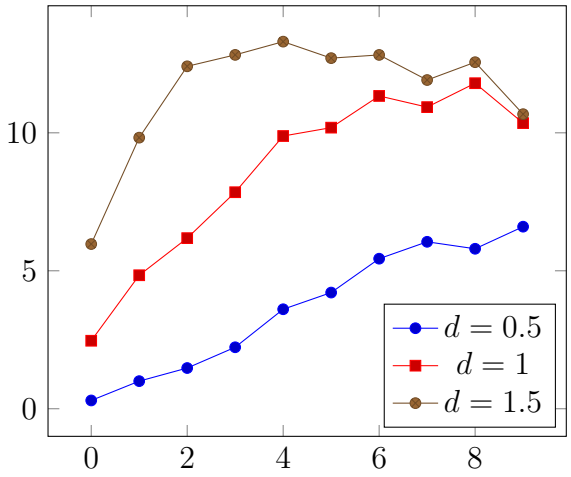
As mentioned in chapter 3, automatic error analysis is usually validated by checking the correlation or F-score between the automatic error annotation and a separate error annotation made by a human annotator. It was also mentioned that the correlations or F-scores attained from these validations are generally low, which seems to be caused by an issue related to reference sparsity: just like there are multiple valid translations for a source sentence, there are also multiple valid corrections for an incorrect translation.

One way to improve the correlation is to constrain the human error annotations to the same reference translation that is used with the automatic methods, but this will lead to misdiagnosing of errors in many cases, where the translation is much more easily corrected to another valid translation other than the reference translation. In the case of EvalGen, the automatic method has access to a much more complete reference set, so this problem does not affect it to the same extent. However, there's another factor which affects the correlation between the automatic and human error annotations: when the number of errors in a translation rises, the number of potential corrections of comparable cost will also rise.

An example of an incorrect translation with two valid corrections was given in the previous chapter: "the ball red colour" can be corrected with two edits, either to "the ball is red" or "the ball is red in colour". Neither correction is obviously better, so the automatic and human annotations may differ even if both are correct and comparable in effort. To see how the number of valid corrections increases as the number of errors rises, consider a worst-case translation for the same sentence, "colour the": it can be corrected to practically all of the valid variants with a comparable cost:

1. the ball is red (one deletion, two insertions, one move)
2. the colour of the ball is red (five insertions)
3. the ball is red in colour (one move, four insertions)

Figure 5: Amount of corrections in the nbest list close to the best correction (distance less than 0.5, 1.0 or 1.5)



4. the ball has a red colour (one move, four insertions)

5. the ball's colour is red (one move, three insertions)

A human annotator would be justified in choosing any of these corrections, so the likelihood of the annotation matching the automatic error annotation output is much lower than with cases where there are fewer errors. In other words, the human method of selecting the correction is nondeterministic, while the error annotation algorithm is deterministic.

To test how prevalent this proliferation of comparable-cost corrections with overall error count is, EvalGen was used to generate nbest lists of corrections for a set of MT hypotheses. For each MT hypothesis, the amount of corrections with a cost close (less than 0.5, 1.0 or 1.5) to the cost of the cheapest correction was counted (see Figure 5). As many corrections are identical with the exception of synonym choice, all identical costs were counted only once (different edit types have different costs, so a different cost implies a genuinely different correction strategy, although it should be noted that many structurally different corrections will also have identical costs and are therefore excluded from these counts).

The nbest list counts confirm that the amount of comparable-cost edits does rise considerably with edit count. As this affects the reliability of error annotation, it might be best if the automatic error annotation was based on weighted sums of the edits for all the corrections in the nbest list within a certain threshold instead of one arbitrarily chosen correction. However,

implementing this was not possible because of time constraints.

7.2.2 Manual evaluation of automatic annotation accuracy

Because of the nondeterministic nature of corrections with high edit counts, and also because correcting sentences with more than a few errors is cognitively hard for annotators, manual evaluation is performed only for sentences for which EvalGen identifies a maximum of three edits (as edit costs differ, the total costs of corrections may be very different, from just over 1 to potentially over 4). A simple annotation program was written for the purpose. The program displays the source sentence, the MT hypothesis, and the indices of the tokens in the MT hypothesis:

```
source: the bottle on the table is red
hypothesis: pullo pöydällä on punainen
              0       1    2      3
```

The annotator inputs the indices for the tokens which need to be edited in order to correct the sentence. As only existing tokens can be marked, indices cannot be used to indicate insertions. Instead, insertions are indicated by specifying the amount of tokens that should be inserted by entering "i" for each token to be inserted.

This annotation method does not specify the kind of edit that is to be performed nor the token that is inserted or substituted, so it is only indicative of the quality of the automatic error analysis. However, this type of evaluation is easy to grasp and quick to perform for the annotators, and it will at least give a negative result in cases where the edits are clearly incorrect.

The focus on the positions of the deletions and substitutions and the amount of insertions also has the advantage of allowing multiple valid corrections. For instance, the MT hypothesis "hopeapohjan levy on vihreä" for the source sentence "the plate on the silver tray is green" has at least three valid, structurally different corrections (and many more with synonym substitution) where the first two tokens are deleted or substituted and one token is inserted:

1. [hopeapohjan/hopeatarjottimella] [ϵ/oleva] [levy/lautanen] on vihreä
2. [hopeapohjan/ϵ] [levy/lautanen] [ϵ/hopeatarjottimella] on vihreä
3. [hopeapohjan/lautanen] [levy/ϵ] [ϵ/hopeatarjottimella] on vihreä

Figure 6: Results of the manual annotation

Evaluator	Precision	Recall	F-score
1	0.90	0.86	0.88
2	0.71	0.75	0.73

With the chosen annotation method, the annotator can quickly enter an annotation covering all of these possibilities. If edits were annotated with more detail, the annotator would have to consider each possibility and choose the most likeliest, which would increase the risk of mismatch.

The results of the annotation test for a set of 50 MT hypotheses are listed in table 6. Despite the predicted low amount of errors in sentences and the simplified annotation method, both evaluators found the task difficult. Especially difficult were sentences requiring insertions and deletions, as they often involved several valid corrections, none of which was obviously best (which confirms that the problem of nondeterminismic corrections seems to also affect corrections with low edit counts). Still, precision and recall of the annotations was relatively high, which indicates that the automatic error identification is functional.

7.2.3 Observations on edit accuracy

The output of EvalGen was inspected many times during the development, which provided insight into the factors affecting the accuracy of the output. The most accurate edits (i.e. feasible from the human point of view) seem to be the ones where a token is edited to another form of the same lemma, or the inflectional properties of the token are retained while the lemma changes. Moves and compound word joins also seem highly accurate. The reason for the apparent accuracy of these edits is probably that the edits retain information in some linguistic dimension, i.e. lemma, inflection or both, which psychologically justifies the edit.

On the other hand, the traditional Levenshtein edits, i.e. insertion, deletion and substitution, are often inscrutable. While deletion and insertion do correspond to recognized error categories of missing and extra words, they are often caused by other issues. Substitution is fundamentally a back-off category: it is used whenever a better-motivated category is not available. Generally the use of substitutions seems to indicate that the edit selection of the method is missing more specific edit types.

Bulk of insertions, deletions and substitutions are caused when the transla-

tion contains an expression which is semantically related to a correct translation or is a feasible mistranslation for it (e.g. a translation of a homonym of the source word). This would be usually characterized as a mistranslation error, since it is clear that the word is intended as translation for a word that should be translated, but the translation has failed in lexical and inflectional aspects. These edits could be more accurately classified by using SMT translation tables (for feasible mistranslations) or a lexical resource, such as WordNet (for semantically similar expressions).

Substitution may also indicate that the MT hypothesis contains an extra word in a position which is missing some other unrelated words. In that case it would be more helpful to consider the edit a combination of an insertion and a deletion. This applies only to cases where the token that is removed and the one that is added are not similar in any aspect.

One strategy for eventually dealing with the opaqueness of insertions, deletions and substitutions is to progressively add more specific error types that cover some subportion of these problematic edits. In EvalGen, inflectional and lexical edits have already covered the largest portion of what would have been substitution edits. By adding these more specific edits, the amount of unreliable edits can be eventually reduced, and insertions, deletions and substitutions can be seen more reliably as indicating missing and extra words.

7.2.4 Mapping edits to errors

EvalGen's primary output is a list, which contains an entry for each MT hypothesis in the evaluated MT set. An example of the list entries can be seen in Figure 7. These individual entries are mostly used to debug EvalGen and to inspect interesting edits.

EvalGen also summarizes the data from entries into an edit list, which is used for general analysis. The edit counts for MT systems evaluated are listed in Table 8.

In order to perform an informative error analysis, these edits will have to be mapped to a taxonomy of translation errors. Due to the limitations of the automatic method, the simplified taxonomy (loosely based on Vilar et al. (2006)) in Table 3 is used. The error categories are listed in order of reliability, with the three last categories being especially unreliable, as was discussed in Section 7.2.3.

Figure 7: EvalGen sample output

```
segment_id: bingtrans_workdir/6.28.Eng.strings.0
source: the Andes have had heavy snowfall for weeks
hyp: andien on ollut raskas lumi sade viikkoja
edits: [CASESHIFT-GEN-ADE: andien/Andeilla (0.34968)] on ollut
[CASESHIFT-NOM-PAR: raskas/raskaita (0.34875)]
[comp_lemma_join: lumi/lumisateita (0.352)]
[comp_lemma_join: sade/<eps> (0.352)] viikkoja
ref: Andeilla on ollut raskaita lumisateita viikkoja
score: 0.23373833333333335 (1.40243/6)
sentence errors: 4.best edit path cost: 1.402430
close to best: [1, 5, 18]
error list
CASESHIFT-GEN-ADE: 1
CASESHIFT-NOM-PAR: 1
comp_lemma_join: 2
```

Figure 8: EvalGen edits by system with Testset500, with local reordering limit of 3 and max 3 reordering moves.

0	MT-1	SMT-2	NMT-2	RBMT-2	SMT-1	RBMT-1	NMT-1	MT-2
insert	130	123	100	173	169	66	81	106
sub	67	108	126	150	205	58	173	135
tensenum_sub	22	22	41	32	22	66	34	21
move	49	100	37	61	125	157	44	47
comp_lemma_join	294	22	4	0	0	0	0	98
unkdel	7	0	0	1	26	0	14	3
animacy_sub	1	31	21	0	0	9	1	2
delete	178	192	71	250	377	273	122	298
casenum_sub	365	376	424	314	432	366	383	315
unksub	10	19	19	1	40	5	15	2
lemma_sub	157	127	110	124	261	106	153	164
caseshift	348	562	291	394	682	329	220	573

Table 3: Simple taxonomy

Error category	**EvalGen edit types**
Word order	Move
Incorrect noun case	Case shifts
Other incorrect form	Other inflectional edits
Unknown words	UNK deletion and substitution
Wrong lexical choice	Lexical edits
Missing words	Insertions
Extra words	Deletions
Other mistranslation	Substitution

7.2.5 Analyzing the error data

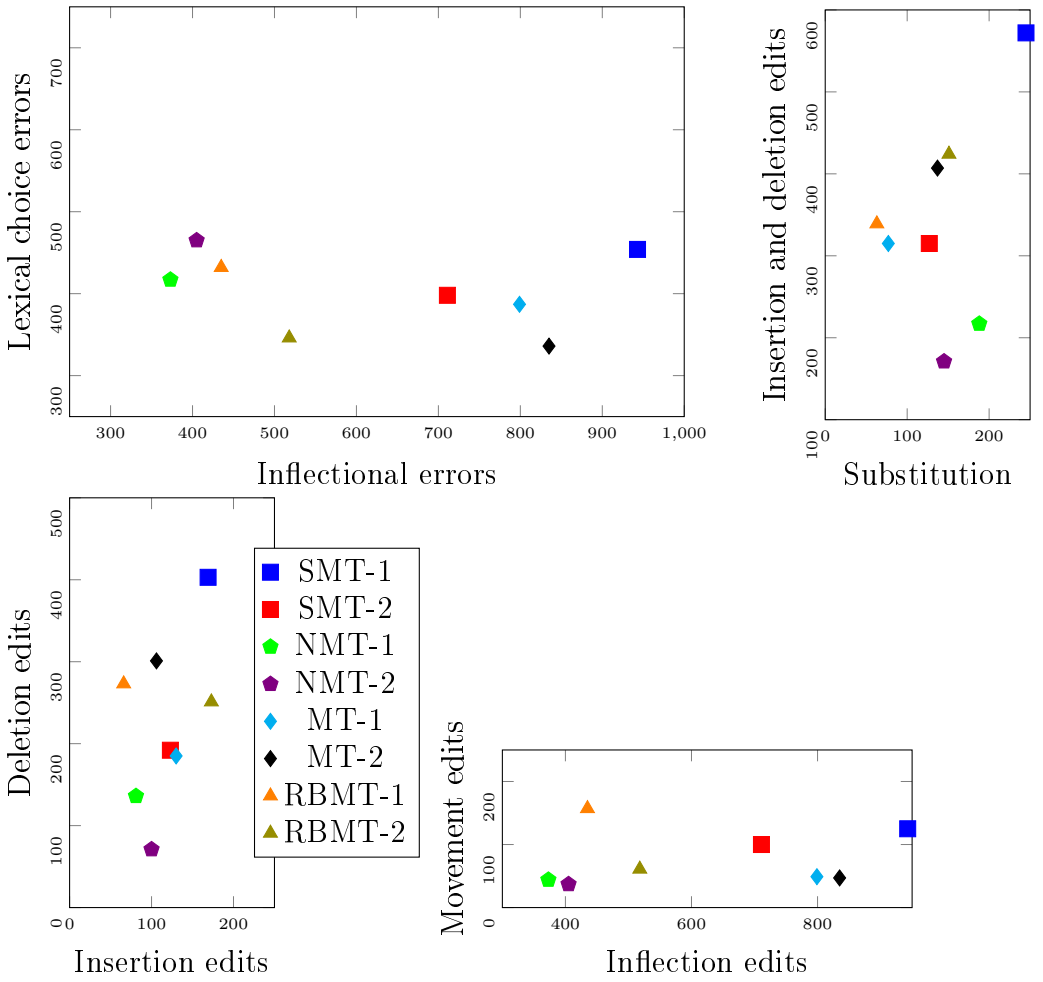
The results clearly indicate large differences in the types and amounts of errors (called error profile from here on) that different MT systems make. The NMT systems are clearly superior in most respects, the only exceptions are the wrong lexical choice errors, substitution edits, and unknown words. Both RBMT and NMT systems require fewer inflection edits than the other systems. But while the two NMT systems have similar levels of edits in almost all categories, the RBMT systems differ greatly from each other and the NMT systems. SMT-1 is an outlier, it is weaker than any other system in almost all aspects. The three other systems occupy a middle ground, and they have similar error profiles.

comp_lemma_join is an interesting edit type, as it occurs only with the online systems (NMT-2, SMT-2, MT-1 and MT-2. It is particularly prevalent with MT-2 (98 occurrences) and especially with MT-1 (294 occurrences). With MT-1, the edit count clearly signals a fault in the system. Inspection of the edits indicates that compound post-processing may be nonfunctional in the system.

Figure 9 visualizes the error profiles of the systems in multiple dimensions. Plotting inflectional errors (incorrect noun case or other incorrect form) against lexical choice errors indicates that there's a slight inverse relationship between the two error types (if the outlier is disregarded). For insertion and deletion edits (primarily indicative of missing and extra words), there's no inverse relationship: the NMT systems are clearly better than the rest.

As mentioned, it is possible that a substitution edit indicates an error that is a combination of an extra word error and a missing word error, and that

Figure 9: Top left: Inflectional vs. lexical choice errors. Top right: Insertions and deletions vs. substitutions. Bottom left: Insertions vs. deletions. Bottom right: Movement vs. inflectional errors



a pair of insertion and deletion edits indicates a mistranslation error (other deviations from the primary error mapping of these edits are also possible). To check if this effect is large, substitutions were also plotted against the sum of insertions and deletions, in order to see whether a system exhibiting relatively many substitutions will also exhibit relatively few insertions and deletions, and vice versa. There may be a minor effect, as the NMT systems with the fewest insertions and deletions also have relatively many substitution edits (especially as a proportion of total edit count).

There are other explanations for the larger amount of substitutions for the NMT systems, though. The primary cause of substitution edits is that the hypothesis token does not share any specified lexical or inflectional properties with the token in the same position in the optimal reference translation. The greater occurrence of lexical errors with NMT means that inflectional errors have a higher chance of co-occurring with lexical errors, therefore resulting in a substitution.

However, there's no straight-forward connection between lexical errors and substitutions, as the RBMT-1 system makes approximately as many lexical errors as the NMT systems, but has a much lower substitution rate than any other system. The explanation for this could be that some feature of RBMT-1 makes it likely to get either the inflection or the lemma correct, but there is no easy way to confirm that. This demonstrates that the substitution edits are very difficult to interpret, and more specific edit types should be introduced to replace substitution.

7.2.6 Effect of recurring edits on error analysis

Since the EvalGen test set is generated from a limited amount of semantic templates, some expressions tend to recur in source sentences. As the source grammar should ideally contain a variety of expressions for each part of the template, recurrence should be relatively limited. However, omissions during test set design or simply the lack of alternatives for the expressions mean that some expressions are repeated almost in all variants of a given semantic template.

Within the test set used in this experiment, this recurrence is most noticeable in the expression "differences" in semantic template 3. There are no alternatives for this expression, so it recurs in all of the source variants of the semantic template. As it happens, "differences" is also a difficult expression to translate, since it requires semantic disambiguation. In fact, almost all of the MT systems consistently fail in this disambiguation task, using the most

frequent sense (having different properties) instead of the sense entailed by the context (disagreeing on things).

The problem is that due to the recurrence of the expression and the consistent errors made by the systems, a large proportion of the edit count of the system will consist of these recurring edits. It would then seem that these errors are weighted too heavily and that they distort the true edit frequencies of the systems. This is a real risk, but there are also mitigating factors. First of all, since these recurring expressions occur embedded in sentences with different structures and elements, they constitute independent translation problems. Total failure in disambiguating this expression in all contexts indicates that the system simply has very limited disambiguation capabilities, and probably will fail with other similarly ambiguous expressions.

Recurring edits undoubtedly have the effect of amplifying the deficiencies of the systems, which is both a negative and a positive. They may make the error profiles less reliable, but on the other hand they emphasize the problems of the system and make them easier to identify and diagnose. Still, it is important to limit the amount of recurring expressions by providing a sufficient amount of alternatives for each source expression (or at least those predicted to be problematic to translate). Another solution that was explored was reducing the weight of the edits with each recurrence, but this was abandoned as it would have penalized inconsistent errors and rewarded consistent errors.

It is important to remember the effect of the recurring edits when interpreting the results of the error analysis. This is especially true with lexical errors, where there is a significant difference between the results of the two NMT systems and RBMT-1, despite their total lexical edit counts being nearly identical. RBMT-1 was the only system to correctly disambiguate the expression "differences", which accounts for almost one fourth of the lexical edits for both of the NMT systems.

RBMT-1 seems to perform bolder disambiguation than the others. However, the analysis is only fair if the disambiguation method that RBMT-1 uses is generally reliable, i.e. if the good performance with "differences" is not just a coincidence. Alternatively, the complete analysis may be fair even if the disambiguation method of RBMT-1 is unreliable, in case there are enough of other disambiguation problems in the test set to make this unreliability affect the lexical edit counts. There are indeed some edits that indicate that RBMT-1's disambiguation is unreliable ("trust" translated in the sense of a "foundation" instead of "confidence"), but the effect of "differences" is probably still large enough to make the total lexical edit count an inaccurate

representation of the capabilities of RBMT-1.

7.2.7 Case shifts

The most accurate edit type in EvalGen is probably the case shift, which occurs when the case of a noun lemma is changed. The accuracy is due to the fact that the lemma information can be used to apply the edit unambiguously, while in other edit types several different edit combinations are possible (see the example in section 7.2.1, where the same sentence can be edited in multiple ways). Consequently, case shifts allow for more careful analysis than other edit types.

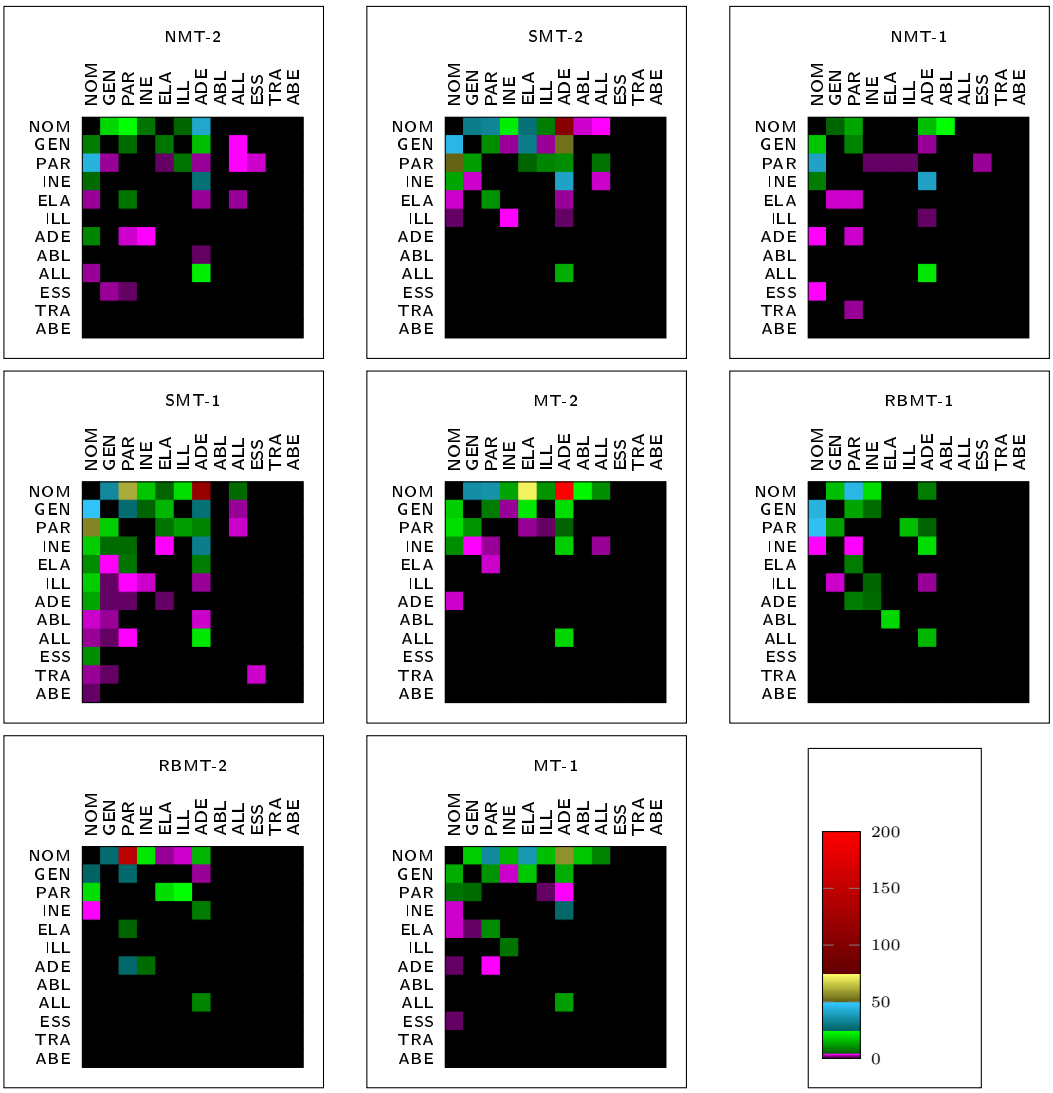
It is apparent from the error counts that outputs of different types of systems tend to require different types of case shifts. The most visible example of this are case shifts from nominative to other cases for SMT systems. To understand the distribution of case shifts for each system, the case shift edits are visualized as heatmaps, where each cell represents a shift from the case specified on the left of the heatmap (source case) to the case specified above it (target case).

The heatmaps are organized so that the grammatical cases (nominative, genitive, and partitive) are first on both axes, followed by the semantic cases. Of the semantic cases, inside locative cases are first, then outside locative cases, and finally the other semantic cases. Comitative and instructive cases are not included, as there were no occurrences of them as source or target cases.

It should be noted that interpreting the heatmaps is not straightforward. As in automatic error analysis in general, the edits may be caused by different types of underlying errors. The ideal case is a sentence with just a single edit, in which case the relation between the edit and the error is clear: for instance, if the word in the subject position uses a case that cannot mark a subject, the edit represents a clear inflection error. If the sentence contains many errors, there may be multiple possible corrections for it, perhaps involving different case forms, and the types of the edits in the chosen correction become somewhat arbitrary. Despite this, clear patterns emerge from case shift analysis, and it can be used to diagnose problems with systems.

Unsurprisingly, most of the case shifts for all systems are from the three grammatical cases into other grammatical or semantic cases. The grammatical cases are the most commonly used ones, so it is to be expected that MT systems overgenerate them (and the edits involving them are likely to be more common just on count of them being more frequent in language).

Figure 10: Case shift heatmaps



Out of the semantic cases, the most common source cases are inessive and elative (corresponding to English "in" and "from"), which are often used in a grammatical role (as are "in" and "from"). Rest of the semantic cases occur as source cases fairly rarely, except for the outlier SMT-1, which contains case shifts from all cases in the heatmap.

There are several areas of interest in the heatmaps. The upper left 3x3 part of the heatmap, which represents shifts from one grammatical case to another, contains a large amount of shifts for most systems. These shifts can be interpreted as problems in modeling the expression syntactic roles, as the cases are used to mark subjects and objects. Within this area, shifts with nominative as source or target case are particularly common. This is somewhat unexpected, since these shifts in most cases signify a shift from subject to object case or vice versa.

It would have been more intuitive for the majority of grammatical case shifts to have been between the two primary object cases, nominative and partitive, as the selection of the correct case requires semantic disambiguation. The unexpected result is partly due to analysis errors (the morphological analyzer interpreted one common genitive form as a nominative), but the main cause seems to be that the test set contains many constructions in which the subject should be in partitive form and the object in nominative form, and the MT systems use the more prototypical subject and object cases instead. In one particular case the grammatical structure of the MT is correct, but it omits a word. In the corrected sentence, the omitted word is placed as the head word of the word that occupies the object position of the original MT. This changes the syntactic properties of the construction, and motivates the case shift edit.

Another interesting area of the heatmap is the adessive column, which contains a large amount of edits for every system. Inspection of edits with adessive as target case suggests several reasons for this. The first is that the English test sentences contain several expressions which are naturally translated as adessives in Finnish ("in the Alps/Andes", "on the dinner table"), and these are for some reason especially difficult for MT to translate. However, the main reason seems to be that MT systems transfer certain English structures into Finnish incorrectly, and converting parts of these translations into adessive expressions is the easiest way to correct them.

For instance, consider the sentence "Russia and China still have major differences". Finnish has no verb with the semantic and grammatical range of "have", the closest Finnish equivalent ("omistaa") strictly denotes owning something. The grammatical functions of "have" (such as denoting some

relation between things, as in the example sentence), are expressed by other means in Finnish. However, many MT systems will try to translate the "have" sentence using the same subject-verb-object structure as in English, with "have" translated as the Finnish copula (possibly due to the fact that "have" and the Finnish copula are both markers of the perfect tense). This produces an incorrect translation, which can however be corrected easily by converting the expression in the subject position to adessive case, as adessive performs the same function of denoting some relation between things that "have" performs in English:

- Source: Russia and China still have major differences.
- MT hypothesis: Venäjä [NOM] ja Kiina [NOM] ovat edelleen suuria erimielisyyksiä. * (Russia and China still are major differences)
- Corrected: Venäjällä [ADE] ja Kiinalla [ADE] on edelleen suuria erimielisyyksiä.

One hypothesis for this apparent difficulty of utilizing adessive expressions in translations is that the combination of semantic and grammatical roles that adessive possesses has no counterpart in English. In addition to denoting relations in the same way as "have", adessive performs similar roles to the English prepositions "on" and "at" in their prototypical locative senses. It can also be used as an instrumental case, in the same way as the English preposition "by". This wide range of uses is probably what makes it difficult for MT systems to utilize adessive correctly.

There are significant differences in adessive usage among the different MT systems: SMT systems perform poorly with them, NMT systems are much better, and RBMT systems are clearly the best. This indicates that the RBMT systems have been provided with rules that can identify English expressions translatable as adessives.

The general layouts of the heatmaps are also interesting, as there seem to be similarities between the patterns of different systems. Three subtypes of case shift patterns can be distinguished:

1. Scattered (NMT systems): The patterns are characterized by the overall low amount of edits, which are scattered over the different parts of the heatmaps. The implication is that the NMT systems mostly select correct cases, and that they utilize a wider range of cases than other systems.

2. Wedge (RBMT systems): The RBMT systems have higher edit counts than the NMT systems, but the edits are more concentrated, leading to a more concise pattern. The systems appear to be fairly accurate with semantic cases, and most of the shifts are between grammatical cases. The wedge-like shape is due to there being no or few edits from the rarer semantic cases to grammatical cases. An explanation for this could be that rare semantic case use is probably triggered by prepositions, which are unlikely to be translated with grammatical cases.

3. Top-heavy (SMT-2 and MT systems): In these systems most of the edits occupy the first two or three rows of the heatmaps, i.e. most edits have a grammatical case as the source case. The systems overuse the most common cases, which indicates that they lack the ability to use context to select correct cases. This is the expected behavior for SMT systems that rely only on translation and language models.

4. Waterfall (SMT-1): This pattern is only exhibited by the outlier system SMT-1. The system output requires a very large amount of edits, and unlike other systems, there are edits from most semantic cases to grammatical cases (including the only abessive shift across all systems). Like the NMT systems, SMT-1 utilizes a wide range of cases, but mostly incorrectly.

An explanation for the aberrant pattern with SMT-1 may be that it uses token segmentation with the intention of improving morphological accuracy. In reality, though, the morphological accuracy deteriorates. The point of segmentation in SMT is to make it possible to translate English function words as segmented Finnish morphemes. The heatmaps indicate that there's a significant chance that the translation from function word to morpheme fails. This is understandable, since each English function word would be aligned to a very large amount of possible morphemes in Finnish, due to mismatches in the semantic ranges of e.g. prepositions and semantic cases (as described for adessive above).

While the case shift analysis is unreliable in some aspects due to nondeterminism of many edits, it can be used to pinpoint problematic areas for different types of MT systems and MT in general. It gives support to the hypothesis that MT systems struggle most with translation tasks where semantic parsing is required (shifts between grammatical cases are very common) or where the source and target language constructions with similar functions are structurally very different (e.g. "have" and adessive case).

Case shift analysis can also be used to identify what MT paradigm a system utilizes. From the heatmaps it would seem very probable that the systems MT-1 and MT-2 are SMT systems, since their heatmap pattern is very similar to SMT-2, and differs from the NMT and RBMT patterns.

7.2.8 Comparison with published manual error analysis results

Bentivogli et al. (2016a) provided the first detailed error analysis of NMT systems. The study utilized post-edited references and automatic error analysis, and it found that NMT systems made significantly less morphological, lexical (including missing and extra words), and reordering errors than SMT systems. These results are consistent with EvalGen results, although EvalGen separates the missing and extra words from incorrect lexical choice errors (which was the weakest category for NMT).

Klubicka et al. (2018) contains the results of a manual error analysis for SMT and NMT systems. The most common type of error for NMT was mistranslation, although even in that category NMT was superior to the two SMT systems in the study. Mistranslation errors included both lexical choice errors and some grammatical errors, so the result is compatible with EvalGen results (where lexical choice errors are separate). The only error type in which NMT was weaker than SMT was omissions. This corresponds to EvalGen's missing words error type, in which NMT systems are only slightly superior to SMT systems (and some missing words may be included in EvalGen's substitution edits, which was a weak point of NMT). As for inflectional errors, the NMT system in the study made only about a third of the inflectional errors that the better SMT system made, which is again consistent with EvalGen results.

Castilho and Gaspari (2017) contains another manual error analysis of SMT and NMT systems. Again, NMT was found to improve fluency and reduce word order errors, while mistranslation and omission errors remained at similar levels. One deviation from the EvalGen results was that NMT was not found to reduce addition errors significantly. s Overall then, EvalGen results seem consistent with published manual error analysis results.

7.3 Quality metric

Another use case for EvalGen is as an automatic metric that provides a single quality score for an MT system. To implement this, the counts of different edits made by EvalGen need to be converted into a single figure of merit

Table 4: System scores sorted by EvalGen scores (best first)

System	EvalGen	BLEU-1	BLEU-10	BLEU-100	BLEU-1660
NMT-2	0.196	3.31	16.79	33.99	41.13
NMT-1	0.201	3.98	17.69	34.50	41.80
RBMT-1	0.212	3.47	15.04	29.86	34.43
MT-1	0.219	3.25	14.68	30.88	37.66
RBMT-2	0.237	3.31	15.48	30.80	36.69
SMT-2	0.242	2.65	11.87	24.47	30.72
MT-2	0.256	2.82	11.86	24.66	30.50
SMT-1	0.372	2.12	8.55	16.07	19.69

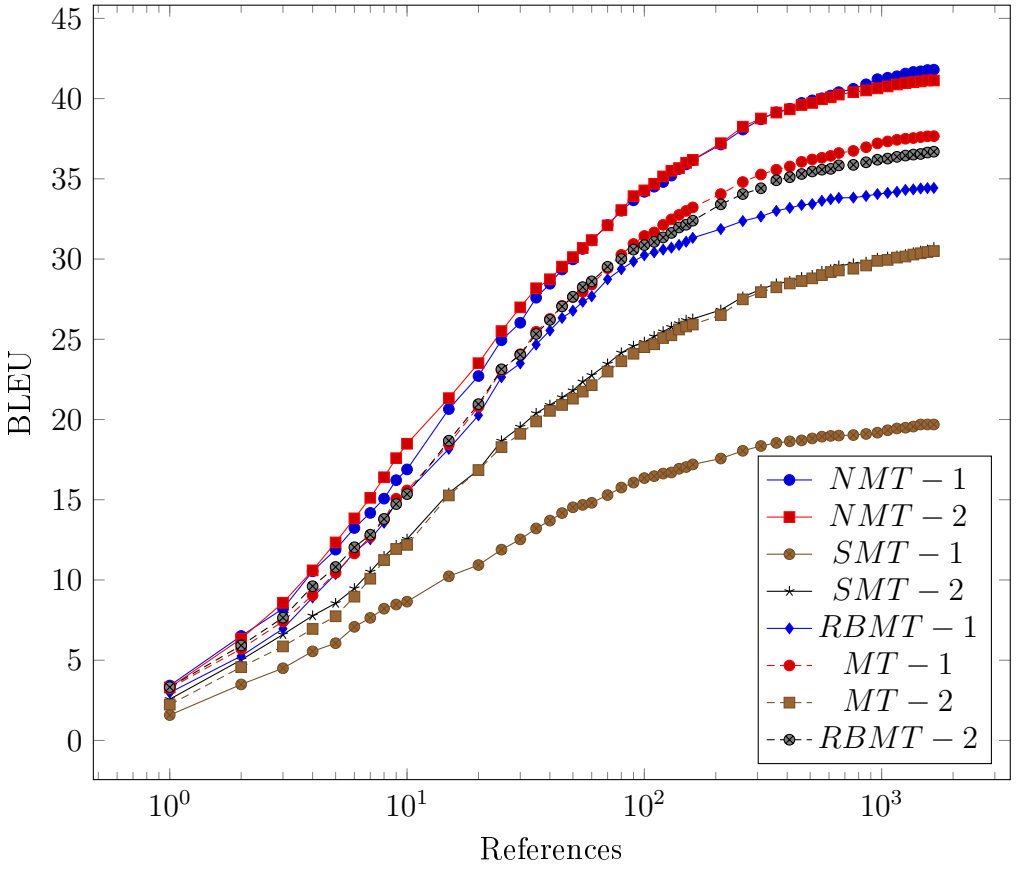
by some formula. Intuitively, the formula should give more weight to edits that are cognitively harder for humans to process, in order to match human perception of MT quality. Alternatively, the weights should be optimized to maximize correlation with manual evaluation scores. However, in absence of any clear data on the cognitive weights of the edits or a set of manual annotations, the score is calculated simply by weighing the errors with the same weights as were used for the edit distance calculation. The weighted error counts are then summed and normalized by corpus length.

The scores for the tested MT systems on Testset500 are listed in table 4. The simple metric produces a ranking where the NMT systems are expectedly on the top, although the RBMT-1 system is close. The good performance of the RBMT systems is most likely due to the relatively simple structure of the sentences in the test set. The outlier system SMT-1 gets a clearly weaker score than the others.

BLEU scores were calculated against different numbers of references using the Moses (Koehn et al., 2007) script multi-bleu-detok.perl. The BLEU scores are listed next to the EvalGen score. The score with just one reference is very low for all systems, which is due to random selection of the reference. In a normal MTE scenario, there tend to be structural similarities between the source and reference, which makes it more likely that the MT produces a translation resembling the reference. With a randomly selected EvalGen reference, the structures may be very different even though the meaning is equivalent. This random selection of references can be seen as a worst-case scenario for BLEU.

The BLEU scores increase quickly with additional references, with typical English to Finnish scores achieved somewhere between 10 and 100 references.

Figure 11: System BLEUs as number of references grows



To see how the increase in references affects the score, the BLEU scores were calculated for each system for different reference amounts ranging from 1 to 1660. The evolution of the scores is displayed in Figure 11.

What's notable is that the fairly large distinctions between systems in EvalGen scoring are not reflected in the BLEU ranking. BLEU seems to group the systems into four distinct groups, which are not present in EvalGen scores to the same degree.

7.3.1 Distinguishing HT from MT

As demonstrated in Lommel (2016), the most common MT metric BLEU fails to reliably distinguish HT from MT. While this does not diminish the usefulness of BLEU (it is always been known to have its limitations), it does indicate a clear lack of resolution at the higher end of the quality scale. To

test whether EvalGen is superior in this respect, Testset100 was translated by both evaluators.

As mentioned, evaluator 1 is the designer of EvalGen and familiar with the expected input, so their translation is only used for comparison. The following instructions were provided to evaluator 2 (rules listed in order of priority):

1. Translate the source sentences into formal standard Finnish of acceptable quality.
2. Try to retain the overall structure of the source sentence (but avoid unnatural expressions, even if they are borderline grammatical).
3. Do not infer meaning from any other knowledge than what is present in the sentence. The sentence should provide all necessary context for word disambiguation.
4. If some lexical ambiguity is not resolvable from the context provided in the sentence, use the most common meaning.
5. Meanings of the sentences are simple and straight-forward, and there are no tricks, so do not spend too much time exploring all feasible translation possibilities, but choose the one that seems the most likely.
6. Punctuation is not taken into account when evaluating the translation, so commas may be omitted.

These instructions are meant to rein in the creativity of the translator and to prevent the addition and omission of information. Instruction 2 is important, as it both prevents unforeseen translations and ensures that the translator varies the translations for the variants of the semantic templates.

The test translation was performed with a computer-assisted translation (CAT) tool, as the segments contain a lot or repetitive elements, and the use of a translation memory considerably speeds up the work. A CAT tool is also the normal working environment for a professional translator.

The results of the test translations were inconclusive: while both human translators achieved a better score than any of the MT systems, the difference between the best NMT system and evaluator 2 was not decisive. The results are listed in table 5. As MT systems are rapidly improving, it is probable that some system will soon be capable of a result comparable to that of evaluator 2.

Table 5: System scores for Testset100, best first. (Testset100 scores were not available for SMT-2 and RBMT-2, so they are not included.)

System	**EvalGen**
HT-1	0.01
HT-2	0.143
NMT-2	0.172
NMT-1	0.176
RBMT-1	0.195
MT-1	0.216
MT-2	0.253
SMT-1	0.362

To understand better why the score difference between the evaluator 2's translation (HT-2) and the best MT (NMT-2) was not decisive, the two outputs were examined in detail. The most obvious difference between the two is that much greater proportion of the edits in HT-2 are recurring. For NMT-2, 129 out of 216 edits are unique (60 percent), while the same ratio for HT-2 is 72/163 (45 percent). A closer look at the edits reveals that HT-2 edits are more systematic, with many edit groups with the same underlying cause: for instance, there are four different edits associated with translating "chance" as "todennäköisyys" (probability).

Over a third of HT-2 edits can be traced to just three constructions:

1. In template 3, evaluator 2 prefers a valid structure which is not included in EvalGen alternatives and which requires different case forms than other alternatives. (27 edits)

2. In template 1, evaluator 2 interprets "precise" in a different way than intended (sense fixing is not implemented properly), and translates it with a two-word phrase. (22 edits)

3. Also in template 3, evaluator 2 uses a valid translation for "differences" ("näkemyserot"), which is not included in EvalGen. (15 edits)

Many of the other edits are valid lexical alternatives not included in EvalGen, such as the verb "pyryttää" for the expression "to snow heavily" in template 6. Another cause of edits was that evaluator 2 used adverb placement to emphasize meanings in a way that was not anticipated. About ten edits were

caused by acceptable repetition of information (as in "a cassette player for cassettes" instead of "a player for cassettes", and also using the translation for the adverb "regularly" with the Finnish verb "harrastaa" which already implies regularity). There was also a small number of actual errors identified in HT-2 output.

To check whether MT systems also used the expressions that caused the unnecessary edits for HT-2, the lexical items related to these edits were searched for in the outputs of all systems for both test sets (600 sentences per system). Though some occurrences were found, they seem very rare. The clearest EvalGen omission, i.e. rejecting the word "todennäköisyys" as translation for "chance", occurred six times in SMT-1 output and nowhere else (this is probably due to the fact that the direct English equivalent "probability" was not used in the source sentences, which is another EvalGen omission). Another unnecessary edit that occurs in MT is the word "laitettu" (translation for "placed" in template 5), which occurs 7 times in RBMT-1 output and 3 times in MT-2 output. The other unnecessary edits from HT-2 do not seem to occur in MT outputs.

The unnecessary edits in HT-2 also raise question whether there are similar systematic unnecessary edits (due to EvalGen omissions) in the MT output. This is unlikely, since the MT outputs were closely inspected during EvalGen development, and EvalGen omissions were corrected as they were discovered. The MT outputs seem fairly strictly constrained to the structures of the source sentences, so they rarely contain the sort of imaginative translations that cause the problems in HT-2.

Output of the evaluator 1 (HT-1) received a nearly perfect score, which was to be expected, since evaluator 1 was the designer of EvalGen. While the result is not reliable as an indicator of the general performance of EvalGen, it does indicate that EvalGen sufficiently encodes the preferences of evaluator 1. This confirms that it is realistic to reach a high coverage of alternatives with the GF-based methods used in EvalGen.

Omissions in EvalGen were clearly the reason for the worse than expected score for HT-2. Three things can be learned from this experiment:

1. Intuitions of a single designer are not sufficient to provide a near-complete coverage of possible alternatives. The design process should be iterative and involve multiple different people as designers and evaluators.

2. Sense fixing is extremely important, as ambiguity leads to unanticipated output.

3. Template complexity should probably be limited, at least initially. The simple template 5, which was included in the test set mainly for comparison, clearly has the best coverage and seems to be sufficiently hard for the MT systems to yield useful information. By contrast, the complex templates suffer from lower coverage and false negatives.

Because the unnecessary edits in HT-2 output were due to only a handful of omissions, most of them lexical, it is simple to extend EvalGen to cover them. Despite the unnecessary edits, the experiment indicates that the coverage of EvalGen is high, and it should be possible to make it nearly complete by implementing the omissions discovered, and then subjecting EvalGen to more iterations of human testing (by different translators) and correction.

7.3.2 Usability of EvalGen as a metric

EvalGen system scores could potentially correlate better with human quality scores: even though the test set is fairly uniform and small, the availability of references should compensate for that. Assuming that MT systems possess a generic translation quality and that manual evaluation scores reflect it, EvalGen should be able to correlate better with manual evaluation scores than single-reference metrics.

While there were no resources available to manually evaluate all the system outputs, several of the test systems also took part in the WMT18 manual evaluation (NMT-1, NMT-2, MT-1, MT-2 and one of the RBMT systems). If EvalGen reliably measures the general quality of MT systems, its scores should correlate with the results of the WMT18 evaluation. However, this does not seem to be the case: while the two NMT systems have comparable EvalGen and WMT manual scores, MT-1 and MT-2 have very different EvalGen scores but almost identical manual evaluation scores. EvalGen also scores the RBMT system high compared to its manual scores.

It should be noted, though, that the systems that EvalGen appears to misjudge have quite low manual evaluation scores, which are likely to be less consistent than higher manual scores. This means it is theoretically possible that EvalGen is capable of ranking systems better at the high end of the quality scale. However, this lack of correlation with manual evaluation suggests that as a metric, EvalGen is best used as a supplement to traditional metrics using larger test sets.

8 Conclusions and future work

The main objective of this thesis was to provide a proof of concept for a new approach to machine translation evaluation, based on manually created multilingual paraphrase grammars. Sections 4, 5, and 6 described the theoretical basis and implementation of this approach.

The results of the evaluation in Section 7 confirm that the approach (EvalGen) is a viable method of MTE. The manual annotation experiment indicates that EvalGen error annotations are reasonably accurate, and the human translation comparison shows that EvalGen can distinguish HT from MT, indicating high coverage of the reference set. EvalGen automatic error analyses are consistent with the manual and semiautomatic analyses published, and they clearly manage to differentiate MT systems belonging to different MT paradigms. Section 7 also introduced case shift analyses, which demonstrate that EvalGen can be used to perform novel types of MT analysis.

It was also shown that EvalGen error analyses can be used to diagnose MT system problems, as they highlighted serious problems with compound concatenation in MT-1 and probable word segmentation problems in SMT-1. In NMT system development, EvalGen could be used to automatically track NMT systems' lexical disambiguation capability, which currently is the weakest point of NMT.

Overall, the objective of producing a working proof of concept has been achieved. However, the results of the experiment also pointed out the following shortcomings in EvalGen:

1. Some edit types are too generic (especially substitution) to yield usable information about the errors that cause them.

2. Recurring elements in source sentences bias the results of systems by amplifying individual errors.

3. High-coverage paraphrase grammars cannot be created by introspection alone. Input from several designers and an iterative design workflow seem necessary.

4. Implementing complex and in any way ambiguous semantic templates is extremely difficult, and will almost certainly lead to gaps in coverage.

5. For most translations with a significant amount of errors, there are multiple acceptable corrections potentially containing different types of

edits. Therefore the edits that form the chosen correction are somewhat arbitrary, and are not as reliable indicators of a MT system's properties and faults as edits from simpler corrections.

6. When Evalgen is used as a metric producing a single figure of merit, the results with the fairly simple test set do not seem to reflect general MT quality. The metric fails to rank systems in the same order as manual evaluation on a larger test set.

Items 1-4 are fixable, and item 6 was somewhat expected. Item 5 however, the nondeterminism of corrections with high edit counts (or occasionally even with small edit counts, as Lommel et al. (2014b) demonstrates), seems like a fundamental problem in error analysis. Similar problems affect the related field of grammatical error correction, where they have been more closely studied (Choshen and Abend, 2018). Ultimately, though, it seems that this problem of correction nondeterminism mainly adds noise to the error profiles of the systems with low quality outputs. Due to the relatively simple semantic templates in EvalGen, the outputs are generally of a fairly high quality, which minimizes the problem.

8.1 Future work

There are many possible extensions and use cases for EvalGen which were not explored due to the limited scope of the thesis. The most interesting use case is as a framework for test suites for evaluating MT systems on some narrow linguistic phenomenon, such as pronoun reference or the effect of punctuation (such as the use of serial comma) on MT output. This could be achieved by creating new semantic templates involving these phenomena or by modifying existing templates.

Another interesting use case would be as a challenge resembling CAPTCHA (von Ahn et al., 2003) or Winograd schemas, intended to differentiate reliably between HT and MT. Currently there are no challenges of this kind for MT (although see Davis (2016) for discussion of using Winograd schemas as MT challenges). However, there have recently been multiple claims of MT quality approaching or even matching HT quality (human parity), so a reliable test for distinguishing MT from HT would clearly be useful for validating these claims.

As for extensions to EvalGen, the most important ones would be to increase variability in the source sentences to avoid recurring edits, and to add an

separate edit type for mistranslation, based on SMT translation tables or WordNet relations. These two improvements would enhance the error analysis significantly by reducing bias and allowing more precise application of edits.

References

Yasuhiro Akiba, Marcello Federico, Noriko K, Hiromi Nakaiwa, and Michael Paul. Overview of the iwslt04 evaluation campaign. In *in Proc. of the International Workshop on Spoken Language Translation*, pages 1–12, 2004.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. Openfst: A general and efficient weighted finite-state transducer library. In Jan Holub and Jan Žďárek, editors, *Implementation and Application of Automata*, pages 11–23, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-76336-9.

Hiyan Alshawi, Shona Douglas, and Srinivas Bangalore. Learning dependency translation models as collections of finite-state head transducers. *Comput. Linguist.*, 26(1):45–60, March 2000. ISSN 0891-2017. doi: 10.1162/089120100561629. URL `http://dx.doi.org/10.1162/089120100561629`.

Vamshi Ambati and Alon Lavie. Improving syntax-driven translation models by re-structuring divergent and nonisomorphic parse tree structures. pages 235–244, 01 2008.

Marianna Apidianaki, Guillaume Wisniewski, Anne Cocos, and Chris Callison-Burch. Automated paraphrase lattice creation for hyter machine translation evaluation. In *NAACL-HLT*, 2018.

Bogdan Babych and Anthony Hartley. Comparative evaluation of automatic named entity recognition from machine translation output. 2004.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Petra Barancikova and Rudolf Rosa. Targeted paraphrasing on deep syntactic layer for mt evaluation. In *DepLing*, 2015.

Emily M. Bender. Grammar engineering for linguistic hypothesis testing. 2007.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *COLING-02: Grammar Engineering and Evaluation*, 2002. URL http://www.aclweb.org/anthology/W02-1502.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. Neural versus phrase-based machine translation quality: a case study. In *EMNLP*, 2016a.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. Neural versus phrase-based machine translation quality: a case study. *CoRR*, abs/1608.04631, 2016b. URL http://arxiv.org/abs/1608.04631.

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, John R. Gillett, John D. Lafferty, Robert L. Mercer, Harry Printz, and Luboš Ureš. The candide system for machine translation. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 157–162, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. ISBN 1-55860-357-3. doi: 10.3115/1075812.1075844. URL http://dx.doi.org/10.3115/1075812.1075844.

A.L. Berger, P.F. Brown, S.A.D. Pietra, V.J.D. Pietra, A.S. Kehler, and R.L. Mercer. Language translation apparatus and method using context-based translation models, April 23 1996. URL http://www.google.com/patents/US5510981. US Patent 5,510,981.

Ondrej Bojar, Matous Machácek, Ales Tamchyna, and Daniel Zeman. Scratching the surface of possible translations. In *TSD*, 2013.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-3001.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn,

Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-4717.

P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. A statistical approach to language translation. In *Proceedings of the 12th Conference on Computational Linguistics - Volume 1*, COLING '88, pages 71–76, Stroudsburg, PA, USA, 1988. Association for Computational Linguistics. ISBN 963 8431 56 3. doi: 10.3115/991635.991651. URL http://dx.doi.org/10.3115/991635.991651.

Franck Burlot and François Yvon. Evaluating the morphological competence of machine translation systems. In *Proceedings of the Second Conference on Machine Translation, Volume 1: Research Papers*, pages 43–55, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4705. URL http://aclweb.org/anthology/W17-4705.

Miriam Butt, Helge Dyvik, Tracy King, Hiroshi Masuichi, and Christian Rohrer. The parallel grammar project. 06 2004.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 136–158, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1626355.1626373.

Carme Carme Armentano-Oller, Antonio M. Corbí-Bellot, Mikel L. Forcada, Mireia Ginestí-Rosell, Boyan Bonev, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Gema Ramírez-Sánchez, and Felipe Sánchez-Martínez. An open-source shallow-transfer machine translation toolbox: consequences of its release and availability. In *OSMaTran: Open-Source Machine Translation, A workshop at Machine Translation Summit X*, pages 23–30, September 2005.

John B Carroll. An experiment in evaluating the quality of translations. *Mech. Translat. & Comp. Linguistics*, 9(3-4):55–66, 1966.

M. Asunción Castaño, Francisco Casacuberta, and Enrique Vidal. Machine translation using neural networks and finite-state models *. 1997.

Antonio Castellanos, Isabel Galiano, and Enrique Vidal. Application of ostia to machine translation tasks. In Rafael C. Carrasco and Jose Oncina, editors, *Grammatical Inference and Applications*, pages 93–105, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. ISBN 978-3-540-48985-6.

Sheila Castilho and Federico Gaspari. A comparative quality evaluation of pbsmt and nmt using professional translators. 2017.

Stanley F. Chen and Joshua Goodman. An emprirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, August 1998. URL `http://acl.ldc.upenn.edu/P/P96/P96-1041.pdf`.

Andrew Chesterman. Catford revisited. In D. Santos, K. Lindén, and W. Ng'ang'a, editors, *Shall We Play the Festschrift Game?: Essays on the Occasion of Lauri Carlson's 60th Birthday*, SpringerLink : Bücher. Springer Berlin Heidelberg, 2012. ISBN 9783642307737. URL `https://books.google.fi/books?id=Wxz015kTE3oC`.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219873. URL `http://dx.doi.org/10.3115/1219840.1219873`.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST@EMNLP*, 2014.

Leshem Choshen and Omri Abend. Inherent biases in reference-based evaluation for grammatical error correction and text simplification. *CoRR*, abs/1804.11254, 2018. URL http://arxiv.org/abs/1804.11254.

Kenneth W. Church and Eduard H. Hovy. Good applications for crummy machine translation. machine translation, 1993.

Ernest Davis. Winograd schemas and machine translation. *CoRR*, abs/1608.01884, 2016.

Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

Mark Dras. Representing paraphrases using synchronous tags. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98, pages 516–518, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. doi: 10.3115/976909.979687. URL https://doi.org/10.3115/976909.979687.

Markus Dreyer and Daniel Marcu. Hyter: Meaning-equivalent semantics for translation evaluation. In *HLT-NAACL*, 2012.

EPSO. Faqs: Why is my score so low for my preliminary test?, 2018. URL https://epso.europa.eu/help/faq/2348_en.

Pål Kristian Eriksen, Seppo Kittilä, and Leena Kolehmainen. Weather and language. *Language and Linguistics Compass*, 6(6):383–402. doi: 10.1002/lnc3.341. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/lnc3.341.

Mark Fishel, Ondrej Bojar, and Maja Popovic. Terra: a collection of translation error-annotated corpora. In *LREC*, 2012.

Kevin Flanagan. Subsegment recall in translation memory — perceptions, expectations and reality. *The Journal of Specialised Translation*, (23), 2015.

Mary A. Flanagan. Error classification for mt evaluation. In *In Proceedings of 1st Conference of the Association for Machine Translation in the Americas (AMTA*, pages 65–72, 1994.

Marina Fomicheva, Núria Bel, and Iria da Cunha. Neutralizing the effect of translation shifts on automatic machine translation evaluation. In *CICLing*, 2015.

Jesus Gimenez, Lluís Màrquez, Elisabet Comelles, Irene Castellón, and Victoria Arranz. Document-level automatic mt evaluation based on discourse representations. pages 333–338, 07 2010.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. Continuous measurement scales in human evaluation of machine translation. pages 33–41, 08 2013.

Liane Guillou and Christian Hardmeier. Protest: A test suite for evaluating pronouns in machine translation. In *LREC*, 2016.

Francisco Guzmán, Ahmed Abdelali, Irina Temnikova, Hassan Sajjad, and Stephan Vogel. How do humans evaluate machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 457–466, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

H. L. W. Hendriks and B. Partee. Montague grammar. In Benthem and Meulen, editors, *Journal of Philosophy*, pages 278–312. MIT Press, 1997.

Hieu Hoang and Philipp Koehn. Improved translation with source syntax labels. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 409–417, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 978-1-932432-71-8. URL `http://dl.acm.org/citation.cfm?id=1868850.1868911`.

J.S. Holmes. *The Name and Nature of Translation Studies.* APPTS, Amsterdam publications and prepublications in translation series. Translation Studies Section, Department of General Literary Studies, University of Amsterdam, 1975. URL `https://books.google.fi/books?id=ad8pHQAACAAJ`.

John Hutchins. Example-based machine translation: a review and commentary. *Machine Translation*, 19(3):197–211, 2005. ISSN 1573-0573. doi: 10.1007/s10590-006-9003-9. URL `http://dx.doi.org/10.1007/s10590-006-9003-9`.

W. John Hutchins. *Machine Translation: From Real Users to Research: 6th Conference of the Association for Machine Translation in the Americas,*

AMTA 2004, Washington, DC, USA, September 28 - October 2, 2004. Proceedings, chapter The Georgetown-IBM Experiment Demonstrated in January 1954, pages 102–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-30194-3. doi: 10.1007/978-3-540-30194-3_12. URL http://dx.doi.org/10.1007/978-3-540-30194-3_12.

Pierre Isabelle, Colin Cherry, and George F. Foster. A challenge set approach to evaluating machine translation. *CoRR*, abs/1704.07431, 2017. URL http://arxiv.org/abs/1704.07431.

Frederick Jelinek. The dawn of statistical asr and mt. *Comput. Linguist.*, 35(4):483–494, December 2009. ISSN 0891-2017. doi: 10.1162/coli.2009.35.4.35401. URL http://dx.doi.org/10.1162/coli.2009.35.4.35401.

Dave Lewis John Moran, Christian Saam. Towards desktop-based cat tool instrumentation. In *Third workshop on post-editing technology and practice (WPTP-3)*, pages 99–112, 2014.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, 2013.

David Kauchak and Regina Barzilay. Paraphrasing for automatic evaluation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 455–462, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220835.1220893. URL https://doi.org/10.3115/1220835.1220893.

Margaret King, Andrei Popescu-belis, and Eduard Hovy. Femti: creating and using a framework for mt evaluation. In *Proc. MT Summit IX*, pages 23–27, 2003.

Filip Klubicka, Antonio Toral, and Víctor M. Sánchez-Cartagena. Quantitative fine-grained human evaluation of machine translation systems: a case study on english to croatian. *Machine Translation*, pages 1–21, 2018.

Kevin Knight. Decoding complexity in word-replacement translation models. *Comput. Linguist.*, 25(4):607–615, December 1999. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=973226.973232.

Philipp Koehn and Hieu Hoang. Factored translation models. pages 868–876, 01 2007.

Philipp Koehn and Christof Monz. Manual and automatic evaluation of machine translation between european languages. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 102–121, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1654650.1654666.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase based translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2003. URL http://acl.ldc.upenn.edu/N/N03/N03-1017.pdf.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL*, 2007.

Maarit Koponen. Comparing human perceptions of post-editing effort with post-editing operations. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 181–190, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2393015.2393041.

Susanne Lauscher. Translation quality assessment. *The Translator*, 6(2):149–168, 2000. doi: 10.1080/13556509.2000.10799063. URL https://doi.org/10.1080/13556509.2000.10799063.

Alon Lavie, Kenji Sagae, and Shyamsundar Jayaraman. The significance of recall in automatic metrics for mt evaluation. In *In Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA-2004*, 2004.

Arle Lommel. Blues for bleu : Reconsidering the validity of reference-based mt evaluation. 2016.

Arle Lommel, Aljoscha Burchardt, Maja Popovic, Kim Harris, Eleftherios Avramidis, and Hans Uszkoreit. Using a new analytic measure for the annotation and analysis of mt errors on real data. pages 165–172, 01 2014a.

Arle Lommel, Maja Popovic, and Aljoscha Burchardt. Assessing inter-annotator agreement for translation error annotation. 2014b.

Nitin Madnani and Bonnie J. Dorr. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Comput. Linguist.*, 36(3):341–387, September 2010. ISSN 0891-2017. doi: 10.1162/coli_a_00002. URL http://dx.doi.org/10.1162/coli_a_00002.

Madras, Madras (India : Presidency). Board of Revenue, and R.A. Dalyell. *The Standing Orders of the Board of Revenue from 1820 to 1865*. J. Higginbotham, 1866. URL https://books.google.fi/books?id=G1oSAAAAYAAJ.

B. Maegaard. Perspectives in artificial intelligence vol. 2: Machine translation, nlp, databases and computer-aided instruction. chapter EUROTRA: The Machine Translation Project of the European Communities, pages 39–47. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. ISBN 0-470-21435-X. URL http://dl.acm.org/citation.cfm?id=65686.65688.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. Precision and recall of machine translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*, NAACL-Short '03, pages 61–63, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073483.1073504. URL https://doi.org/10.3115/1073483.1073504.

Donald Metzler, Eduard Hovy, and Chunliang Zhang. An empirical evaluation of data-driven paraphrase generation techniques. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 546–551, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL http://dl.acm.org/citation.cfm?id=2002736.2002844.

G.A. Miller and J.G. Beebe-Center. Some psychological methods for evaluating the quality of translations. *Mechanical Translation*, 3:73–80, 1958.

Teruko Mitamura. Controlled language for multilingual machine translation. In *In Proceedings of Machine Translation Summit VII*, 1999.

Mehryar Mohri. Edit-distance of weighted automata. In Jean-Marc Champarnaud and Denis Maurel, editors, *Implementation and Application of Automata*, pages 1–23, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-44977-5.

Makoto Nagao. A framework of a mechanical translation between japanese and english by analogy principle. In *Proc. Of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, New York, NY, USA, 1984. Elsevier North-Holland, Inc. ISBN 0-444-86545-4. URL http://dl.acm.org/citation.cfm?id=2927.2938.

NIST. The nist 2002 machine translation evaluation plan (mt-02), 2002. URL https://catalog.ldc.upenn.edu/docs/LDC2010T10/DARPATIDESMT02EvalPlan_-v1-3.pdf.

F. J. Och. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. PhD thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, October 2002.

Franz Josef Och and Hans Weber. Improving statistical natural language translation with categories and rules. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 985–989, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics. doi: 10.3115/980691.980731. URL http://dx.doi.org/10.3115/980691.980731.

Karolina Owczarzak, Declan Groves, Josef van Genabith, and Andy Way. Contextual bitext-derived paraphrases in automatic mt evaluation. In *WMT@HLT-NAACL*, 2006.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-jing Zhu. Bleu: a method for automatic evaluation of machine translation. 10 2002.

Ellie Pavlick, Juri Ganitkevitch, Tsz Ping Chan, Xuchen Yao, Benjamin Van Durme, and Chris Callison-Burch. Domain-specific paraphrase extraction. In *ACL*, 2015.

Sheila M. Pfafflin. Evaluation of machine translations by reading comprehension tests and subjective judgments. *Mechanical Translation*, 8:2–8, 1965.

John R. Pierce and John B. Carroll. *Language and Machines: Computers in Translation and Linguistics*. National Academy of Sciences/National Research Council, Washington, DC, USA, 1966.

Tommi A Pirinen. Development and use of computational morphology of finnish in the open source and open science era: Notes on experiences with omorfi development. *SKY Journal of Linguistics*, 28:381–393, 2015.

Andrei Popescu-Belis. The place of automatic evaluation metrics in external quality models for machine translation, 2007.

Maja Popovic. chrf: character n-gram f-score for automatic mt evaluation. In *WMT@EMNLP*, 2015.

Maja Popovic. *Error Classification and Analysis for Machine Translation Quality Assessment*. 09 2018.

Maja Popović and Hermann Ney. Towards automatic error analysis of machine translation output. *Comput. Linguist.*, 37(4):657–688, December 2011. ISSN 0891-2017. doi: 10.1162/COLI_a_00072. URL `http://dx.doi.org/10.1162/COLI_a_00072`.

Maja Popovic, Adrià de Gispert, Deepa Gupta, Patrik Lambert, Hermann Ney, José B. Mariño, Marcello Federico, and Rafael E. Banchs. Morphosyntactic information for automatic error analysis of statistical machine translation output. In *WMT@HLT-NAACL*, 2006.

Mark A. Przybocki, Kay Peterson, and Sebastien Bronsart. Translation adequacy and preference evaluation tool (tap-et). In *LREC*, 2008.

Aarne Ranta. The gf resource grammar library. 2009.

Aarne Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. Center for the Study of Language and Information/SRI, 2011. ISBN 1575866269, 9781575866260.

Rudolf Rosa, David Marecek, and Ondrej Dusek. Depfix: A system for automatic correction of czech mt outputs. In *WMT@NAACL-HLT*, 2012.

Henry S. Thompson. Automatic evaluation of translation quality: Outline of methodology and report on pilot experiment. 04 1991.

Holger Schwenk. Continuous space translation models for phrase-based statistical machine translation. In *COLING*, 2012.

Alina Secară. Translation evaluation- a state of the art survey.

Rico Sennrich. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. *CoRR*, abs/1612.04629, 2016. URL `http://arxiv.org/abs/1612.04629`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015a. URL http://arxiv.org/abs/1508.07909.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709, 2015b. URL http://arxiv.org/abs/1511.06709.

Dana Shapira and James A. Storer. Edit distance with move operations. In *Proceedings of the 13th Annual Symposium on Combinatorial Pattern Matching*, CPM '02, pages 85–98, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43862-9. URL http://dl.acm.org/citation.cfm?id=647821.736368.

Stuart M. Shieber. An introduction to unification-based approaches to grammar. In *CSLI Lecture Notes*, 1986.

Dimitar Shterionov, Riccardo Superbo, Pat Nagle, Laura Casanellas, Tony O'Dowd, and Andy Way. Human versus automatic quality evaluation of nmt and pbsmt. *Machine Translation*, May 2018. ISSN 1573-0573. doi: 10.1007/s10590-018-9220-z. URL https://doi.org/10.1007/s10590-018-9220-z.

H. W. Sinaiko and R. W. Brislin. Evaluating language translations: Experiments on three assessment methods. 1973.

Georges Van Slype. Critical study of methods for evaluating the quality of machine translation. final report, 1979. URL http://aei.pitt.edu/39751/. Sections are bookmarked.

Matthew Snover, Bonnie Dorr, Richard Schwartz, John Makhoul, Linnea Micciulla, and Ralph M. Weischedel. A study of translation error rate with targeted human annotation. 2005.

Matthew G. Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2/3):117–127, 2009a.

Matthew G. Snover, Nitin Madnani, Bonnie J. Dorr, and Richard M. Schwartz. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23:117–127, 2009b.

Harold Somers. Review article: Example-based machine translation. *Machine Translation*, 14(2):113–157, 1999.

ISSN 1573-0573. doi: 10.1023/A:1008109312730. URL http://dx.doi.org/10.1023/A:1008109312730.

Milos Stanojevic and Khalil Sima'an. Beer: Better evaluation as ranking. In *WMT@ACL*, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

Midori Tatsumi. Correlation between automatic evaluation metric scores, post-editing speed, and some other factors. 01 2009.

Henry Thompson and Chris Brew. Automatic evaluation of computer generated text: Final report on the texteval project. 08 1996.

C. Tillmann, S. Vogel, H. Ney, and A. Zubiaga. A dp based search using monotone alignments in statistical translation. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, EACL '97, pages 289–296, Stroudsburg, PA, USA, 1997a. Association for Computational Linguistics. doi: 10.3115/979617.979654. URL https://doi.org/10.3115/979617.979654.

Christoph Tillmann, Stephan Vogel, Hermann Ney, A Zubiaga, and Hassan Sawaf. Accelerated dp based search for statistical translation. 01 1997b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

David Vilar, Jia Xu, Luis D'Haro, D Haro, and Hermann Ney. Error analysis of statistical machine translation output. 05 2006.

Sami Virpioja and Stig-Arne Grönroos. Lebleu: N-gram-based translation evaluation score for morphologically complex languages. In *WMT@EMNLP*, 2015.

Sami Virpioja, Jaakko J. Väyrynen, Mathias Creutz, and Markus Sadeniemi. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. 2007.

Luis von Ahn, Manuel Blum, Nicholas Hopper, and John Langford. Captcha: Using hard ai problems for security. In *EUROCRYPT*, 2003.

Warren Weaver. Translation. 1949. reprinted in Machine translation of languages / fourteen essays. Ed. by W.N.Locke and A.D.Booth.

Bonnie Webber, Marine Carpuat, Andrei Popescu-Belis, and Christian Hardmeier. *Proceedings of the Second Workshop on Discourse in Machine Translation (DiscoMT 2015)*. Association for Computational Linguistics, September 2015. ISBN 978-1-941643-32-7. URL http://aclweb.org/anthology/W15-25.

Jonathan Weese, Juri Ganitkevitch, and Chris Callison-Burch. Paradigm: Paraphrase diagnostics through grammar matching. pages 192–201, 01 2014.

John White. The rather-larger-than-expected utility of black-box mt evaluation. Contribution to panel, 1994a.

John White and Kathryn B. Taylor. A task-oriented evaluation metric for machine translation. 01 1998.

John S. White. The arpa mt evaluation methodologies: Evolution, lessons, and further approaches. In *Proceedings of the 1994 Conference of the Association for Machine Translation in the Americas*, pages 193–205, 1994b.

John S. White, Theresa A. O'Connell, and Francis O'Mara. The arpa mt evaluation methodologies: Evolution, lessons, and future approaches. 1994.

Tony Whitecomb. Statistical methods gaining ground. 05 1992.

Yorick Wilks. *It Works but How Far Can It Go: Evaluating the SYSTRAN MT System*, pages 65–86. Springer US, Boston, MA, 2009. ISBN 978-0-387-72774-5. doi: 10.1007/978-0-387-72774-5_4. URL https://doi.org/10.1007/978-0-387-72774-5_4.

Dekai Wu. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 152–158, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. doi: 10.3115/981863.981884. URL http://dx.doi.org/10.3115/981863.981884.

y. Bar-Hillel. Language and information–selected essays on their theory and application. *British Journal for the Philosophy of Science*, 16(63):253–255, 1965.

Jitka Zehnalová. Tradition and trends in translation quality assessment. OMLS, Vol 2:41–56, 01 2013.

Daniel Zeman, Mark Fishel, Jan Berka, and Ondrej Bojar. Addicter: What is wrong with my translations? *Prague Bull. Math. Linguistics*, 96:79–88, 2011.