Master Thesis Universiteit Leiden

“I know the man you saw yesterday not”

Improving Statistical Machine Translation on Negation from Dutch-to-English

name:	S. M. Edelenbos	university:	Leiden University
student number:	s0932353	faculty:	Humanities
date:	31-5-2016	department:	Linguistics
e-mail:	s.m.edelenbos@umail.leidenuniv.nl	supervisor:	dr. C. L. J. M. Cremers
		second reader:	dr. M. Elenbaas

Table of Contents

Chapter 1: Introduction 2
1.1 Translation Applications 2
1.2 The Structure and Scope of this Paper 4
Chapter 2: Statistical Machine Translation 6
2.1 Introduction to Statistical Machine Translation 6
2.2 How Statistical Machine Translators work 6
2.3 Training in Statistical Machine Translation 10
2.4 An Analysis of Statistical Machine Translation Applications 18
Chapter 3: Linguistic Theories on Negation 26
3.1 English and Dutch sentence structures 26
3.2 Negation and Scope 26
3.3 Negation and Polarity Items 29
3.4 Negation and WH-Movement 31
3.5 Semantic Features of Negation 37
3.6 Merging Semantics with Syntax 41
3.7 Combining Theories on Negation 48
Chapter 4: Critical Analysis and Comparison 51
4.1 Problems with Machine Translation 51
4.2 Improvements for Statistical Machine Translators 52
4.3 Conclusion 55
References 57

Chapter 1: Introduction

1.1 Translation Applications

Today, more and more people are making use of online applications that are being designed in order to make day-to-day life easier. Amongst others, companies such as Google Incorporated,[1] Microsoft Corporation and Apple Incorporated either develop or are involved in the production of such online applications, often referred to as 'apps'. These applications often use various technologies or approaches on the vast amount of data made available through the Internet in order to deliver the comforts they offer their users. Amongst these apps are translations applications; in case of the before-mentioned companies, examples of such translation apps are Google Translate™ , Microsoft Translator™ and iTranslate™ respectively.[2] Online translation applications are extensively used on various platforms, with Google Translate™ alone serving over 200 million people a day in 2013.[3] This is not without good reason – these translation applications are highly effective, being able to give translations, as well as alternative suggestions within fractions of seconds for over 30 different languages. However useful these applications might be to the enormous amount of users they serve each day, they occasionally do fail to produce satisfying correct sentences.

The examples of translation applications presented above are not radically new technological advances in information technology from the last decade. Rather, the usability and accessibility of such software (as well as the data they require in order to operate) have become radically easier and cheaper to produce to such a large user audience. The immense popularity of smart phones and the greater accessibility and range of the Internet being mainly responsible for these developments. The general programming approach to this electronic method of translating has been around since the 1950s, which has gained interest again in the

---

[1] Note that whilst working on this thesis, the company Google Incorporated has split in order to form its own mother company called 'Alphabet Inc.'. This mother company not only houses Google Inc., but rather more Information Technology-related companies which may or may not have been part of Google Inc. originally. As relatively little is known about how the original Google Incorporated has been split across Alphabet Incorporate, I will simply assume that its application and translation departments are still housed under the Google-branch of the corporation. See: D'Onfro, J. (2015, October 2) Google is now Alphabet. *Business Insider UK*. Retrieved from http://uk.businessinsider.com/google-officially-becomes-alphabet-today-2015-10?r=US&IR=T

[2] As found on Google Play, https://play.google.com/store/apps/details?hl=en&id=com.google.android.apps.translate, Microsoft.com, http://www.microsoft.com/en-us/translator/apps.aspx, and on Apple iTunes, https://itunes.apple.com/us/app/itranslate-free-translator/id288113403?mt=8 on 02-02-2016.

[3] Cnet, Google Translate now serves 200 million people a day, May 18 2013. By Stephen Shanklank http://www.cnet.com/news/google-translate-now-serves-200-million-people-daily/

1980s and 1990s with the large increase of use in computers.[4] This particular method of translating through the use of machines such as computers is called 'Statistical Machine Translation'. As the name of this methodology suggests, translation is performed through the use of statistics. This seems as a relatively simple use of the computing in order to facilitate the translation of words as well as sentences; it is, however, not perfect.

Take the example below in (1), where a Dutch sentence is translated into English using two of the machine translators mentioned above:[5]

(1) (i) Ik ken de man die jij gisteren zag niet.
I do not know the man you saw yesterday.
a. I know the man you saw yesterday. (Google Translate™)[6]
b. I know the guy you know yesterday saw not. (Microsoft Translator™)

As both (1a) and (1b) show, both Statistical Machine Translators fail to produce the right translation of the sentence in (i). Out of the two translations, (1b) seems to have had the most difficulty in translating (i). However, just as in (1a), there seems to be a particular difficulty in properly translating the end of sentence (i). To give a more extensive comparison, more Dutch examples are given to both these translation applications as presented in (2):

(2) (i) Ik hoor jou niet.
I can't hear you.
a. I can not hear you. (Google Translate™)
b. I hear you not. (Microsoft Translator™)
(ii) Ik kan jou niet horen.
I can't hear you.
c. I can not hear you. (Google Translate™)
d. I can not hear you. (Microsoft Translator™)
(iii) Ik ken de buurman niet.
I do not know the neighbor.
e. I know the neighbor not. (Google Translate™)
f. I know the neighbor not. (Microsoft Translator™)
(iv) Ik werk in het weekend niet.

[4] Koehn, P. (2010). *Statistical Machine Translation.* Cambridge: Cambridge University Press. (PAGE?)
[5] These two were specifically used due to access to these two applications is free when using a web browser. Google Translate™: http://translate.google.com/, Microsoft Translator™: http://www.bing.com/translator/
[6] At times, Google Translate™ can also produce the translation "I know the man you did not see yesterday".

I don't work on weekends.

g. I do not work on weekends. (Google Translate™)

h. I work in the weekend is not. (Microsoft Translator™)

(v) Ik geef mijn cadeau niet.

I will not give my gift.

i. I give my gift not. (Google Translate™)

j. I give my gift not. (Microsoft Translator™)

As some of these examples show, not every translation is completed as successful as someone would expect. A curious phenomenon these examples show, is the translation and the placement of the negation – in these examples, this is the Dutch word '*niet*'.

The purpose of this thesis is to examine why Statistical Machine Translators, such as the examples given above, have such difficulty coping with the translation and placement of negation in Dutch when translating to English. Secondly, this thesis will examine various linguistic theories regarding negation and try to suggest aspects from these theories which could be added to the methodology used in these applications in order to improve their translation of negations in Dutch to English.

1.2 The Structure and Scope of this Paper

First, in chapter 2, the methodology of Statistical Machine Translation will be discussed. At the end of this chapter, the examples discussed in the introductory paragraph above will be analyzed. By doing so, a clear understanding of how Statistical Machine Translation can be established before it can be critically analyzed. The following chapter 3 will focus on theories on negation from different areas within linguistics. This way, when the examination on the mechanics behind the methodology of Statistical Machine Translation arises, the right tools are in place in order to notice any possible faults in how these applications work. The critical analysis on Statistical Machine Translation will be presented in chapter 4, alongside commentaries on Machine Translation found in other literature. Finally, in the concluding chapter 5, suggestions on improving Statistical Machine Translators from the literature discussed will be presented.

It is important to note that the scope of this paper is on Statistical Machine Translation, and will leave other types of Machine Translation (such as Rule-Based Machine Translation)

outside its scope. Also, the purpose of this thesis is not to present a new algorithm, develop new software or improve existing software through additional programming. Instead, the purpose is to analyze an approach to translating languages which is popular is use from the perspective of linguistic frameworks.

Chapter 2: Statistical Machine Translation

2.1 Introduction to Statistical Machine Translation

According to the *Encyclopedia of Language & Linguistics*, a corpus-based methodology to translation became popular during the nineties of the previous century. This was due to increasing simplicity through which various corpora could be accessed as well as the increasing speed through which these corpora could be established.[7] It comes to no surprise that this was due to the vast improvements made in computer technology – amongst which the introduction of the World Wide Web and the Internet played crucial roles.

It varies between the Statistical Machine Translation programs what 'chunks' of sentences or phrases are selected for the lexical choice. In the purest and simplest form, the lexical choice revolves around the selection of separate words which are to be paired and compared to see what the highest probable translation is. Whether it is due to experience with users, the users' demand, or the knowledge that the mere translation of separate words does not suffice when translating, a range of Statistical Machine Translators have been developed which each make their own choice in their lexical selection. Most new Machine Translators use statistical models of a more phrase-based selection. Clusters of words from the input sentences are selected and are then compared to similar clusters to come to the most probable translation. The use of the word 'cluster' here is due to the logic behind the selection of these so-called 'phrases'. Most of these phrases do not really share a syntactically or semantically logical relation, and are then translated based on their occurrences from the corpora. Other phrase-based Machine Translators have found a pragmatically clever way to optimize their translation results. Namely to re-cluster the phrase various times and to, at a later stage, use the overlapping results to produce more-probable translations.

2.2 How Statistical Machine Translators work

According to their article, Hearne and Way (2011) state that Statistical Machine Translation is one form of two popular corpus-based machine translation methods. The other method

[7] Bernardini, S. (2006). Machine Readable Corpora. In *Encyclopedia of Language & Linguistics.* (Second Edition), 358-375. Retrieved from http://www.sciencedirect.com/science/article/pii/B0080448542004764. This text also provides insight into various corpus-translation specific research done in the nineties. For this paper, however, these researches lay outside the scope.

being that of Example-Based Machine Translation.[8] The main difference between the two, according to their article, is that the latter translates words or phrases based on how the machine has previously translated sentences which seem similar to the ones the machine is presented with, whereas the former uses a slightly more complex and more effective method which will be discussed in more detail below.

Nonetheless, both methods are still corpus-based methods. This entails, as the name might suggest, that the translation machines make use of two sets of corpora: one set of the source language's corpora, and another set belonging to the target language. These corpora are not random corpora, but are corpora containing parallel documents. That is to say, that documents are the same, except that these have been translated by human translators. Often used documents in these corpora are (translated) novels or minutes made during international meetings (such as those of the European Union or the United Nations). In the simplest of forms, these corpus-based translation machines link words and phrases found several times in the corpora of the source language, and makes links to words in the corpora of the target language. This is where the process between Example-Based Machine Translation and Statistical Machine Translation seem to split ways.[9] From here on, only Statistical Machine Translation will be explained further.

In their paper, Farzi et al (2015) describe the tasks of Statistical Machine Translation as follows:

"The Machine Translation task is made of two sub-tasks: collecting the list of words in a translation, which is called the lexical choice, and determining the order of the translated words, which is called reordering."

However, according to Hearne & Way (2011), the two tasks of Statistical Machine Translation are actually "training" and "decoding". It can only be assumed that the difference in terminology for the procedural choice lies in that there is no 'pure' authority in how

[8] Hearne, M. &Way, A. (2011) Statistical Machine Translation: A Guide for Linguists and Translators. *Language and Linguistics Compass, 5,* 205-226.

[9] However, Hearne & Way (2011) make a note that the term 'Statistical Machine Translation' is *not* a 'proper' generic term, due to various so-called Statistical Machine Translators can use widely different methods. However, as Hearne & Way also acknowledge, these variations still come down to the same basic principle. They briefly make the suggestion that terms, such as "Probabilistic Machine Translation", would be more appropriate. Even wider terms, such as "Data-Driven Machine Translation", they suggest as an alternative. However, just as they seem to compromise on themselves, Statistical Machine Translation will be the used term in this paper.

Statistical Machine Translation is supposed to work,[10] however, when looking more closely, their two approaches are not that different from one and other. It rather seems that Hearne & Way (2011) seem to focus more on the translational-aspect of the process, rather than the procedural-aspect, as Farzi et al (2015) do. Their two approaches can, however, be combined to present to the full four-point general procedure of Statistical Machine Translation:[11]

(3) (i) **Training**: the Machine Translator 'learns' what corresponding words and phrases there are (in general) within the source language corpora and the target language corpora, including a probability for each corresponding set being a proper translation.
(ii) **Decoding**: once a translation query has been entered, the Machine Translator finds all corresponding sets which suit the given query (also including the sets with the lowest translation probability).
(iiia) **Lexical Choice**: The Machine Translator selects the most probable translations from the set presented after step (ii).
(iiib) **Reordering**: The Machine Translator finally places the selected words in an order which is suits the grammar of the target language as best it can.

As mentioned above these translation methods use statistical formulae in order to calculate what the most probable translation can be. This is where most of the individual Statistical Machine Translators try to distinguish themselves in the most. The general approach used in these Translation Machines is virtually the same. They all use a variation of the Bayes' Theorem. In statistics, this is a much-used theorem to predict the probability of an event. In its purest form, the theorem's formula is as follows:

(4) $$P(A|B) = \frac{P(A)\cdot P(B|A)}{P(B)}$$

This formula states that the probability (P) of the event A occurring if the event B is true equals the probability of event A multiplied by the probability of B if A is true divided by the probability of event B. In the case of Statistical Machine Translation, this implies that this

[10] See the footnote above.
[11] Based on Hearne & Way (2011) and Farzi et al (2015). Note that steps (ii) Decoding and (iii) Lexical Choice are very similar. It is possible that Farzi et al and Hearne & Way are actually referring to the same procedural step here. However, due to its ambiguity, they are presented as two separate steps here. Also, the reason why Hearne & Way's steps precede those of Farzi et al is because Hearne & Way seem to present an even more general approach as to how Statistical Machine Translators work – also the steps which seem to be taken *before* a translation query is offered. Farzi et al present steps which seem to occur whilst a query is being presented.

theorem is used to calculate the probability of a correct translation (P (A|B) depending on probabilities, or frequencies, of the corresponding words paired from the databases from both languages.

According to Mukesh et al (2010), this formula can be simplified for the purposes of Statistical Machine Translations as follows:[12]

(5) $$P(A|B) = P(A)P(B|A)$$

The extra division with P(B) can be ignored, due to the probability of every (B) is the same of every (A).

Hearne & Way (2011) present more elaborate formulae used in Statistical Machine Translation. The two formulae they present are as follows:[13]

(6) (i) Noisy-channel model

$$\text{Translation} = \text{argmax}_T\, P(S|T) \cdot P(T)$$

(ii) Log-linear model

$$\text{Translation} = \text{argmax}_T \sum_{m=1}^{M} \lambda_m \cdot h_m(T, S)$$

The first formula, which is called the 'Noisy-channel' model, is relatively similar to the Bayesian method described above. The difference is that this formula includes the notion of a maximum value for a probable translation (argmax_T, where *T* stands for 'translation'). Otherwise, this formula is exactly the same as Mukesh et al's (2010) formula. The second formula, however, requires some more explanation. Firstly, the formula uses logarithmic probabilities. Thus, the *(T, S)* at the end refers to a logarithmic probability of *P(T)* and *P(S)* respectively. Furthermore, the *M* refers to features given to parts of the translation process, followed by λ_m, which is the weight of those particular features, and it are these features which are taken in the summation, multiplied by their weight as well as the 'score' of that feature multiplied by the logarithmic probabilities within *T* and *S* respectively. Finally, the maximum scoring probable translation ($argmax_T$) is seen as the most probable translation in the target language of the source language's input.

In their paper, they point out that the second type of formula (the Log-linear model), can be preferred over a Noisy-channel model, mostly due to the fact that within the formula, the

[12] Mukesh, G.S. & Vatsa, N. J & Goswami, S. (July 2010). Statistical Machine Translation. *DESIDOC Journal of Library & Information Technology*, *30 (4)*, 25-32.
[13] Hearne & Way (2011), pp. 206-208.

value of the features can be adjusted per translation. Whereas the Noisy-channel model is an approach where, at first, the Machine Translator is most likely not at all good at translating at all. However, the Noisy-channel model allows for the Machine Translator to gradually become more successful after an evaluation process involving a scoring mechanism. As described by Hearne & Way (2011), a simplified evaluation of the translation can be seen as follows:[14]

(7) In the Source Language (French), we have the following sentence:

Le chat entre dans la chambre. See Target Language (English) translations :

(i) The cat enters the room. **Adequate Fluent translation**

(ii) The cat enters in the bedroom. **Adequate Disfluent translation**

(iii) My granny plays piano. **Fluent Inadequate translation**

(iv) piano granny the piano My. **Disfluent Inadequate translation**

Here, (i) represents a well-translated sentence[15] and (ii) represents a translation which has a reasonably good lexical translation with a poor grammatical choice. Followed by (iii) which has a good sentence structure, but is far from being a good lexical translation. Finally (iv) represents a translation which is lacking in both lexical as grammatical sense. Using these four examples, they would each score differently based on the success of their lexical and grammatical features.[16] The scores reflecting on the lexical translation would be added to the value of $P(S|T)$ from the Noisy-channel model as presented in (6). Scores reflecting on the grammatical aspects of the translation would be added to the value of $P(T)$, as presented in (6). This implies that, gradually, the Translation Machine using this formula will improve and will present more successful translations more often.

2.3 Training in Statistical Machine Translation

Now that the basics of the statistical formulae used in Statistical Machine Translation have been discussed, it becomes clear that these formulae all use a probability of a word or phrase-combinations found in parallel corpora. However, the next question is how these probabilities are established. Outside of human interaction through the use of scoring evaluations, what defines the probability of such combinations?

[14] As presented in Hearne & Way (2011), p. 207).
[15] A well-presented translation in a context-free setting.
[16] These scores would be given by users of the Statistical Machine Translator. See Hearne & Way (2011), p. 207.

As described above in (3), the general procedure of how Statistical Machine Translators work has been defined into four steps, through the examples provided by Hearne & Way (2011) and Farzi et al. (2015). In the previous chapter, the general basic formulae used in most of these Machine Translators have been explained. However, the process of these individual steps have not been covered in full detail yet. First, the *Training*-phase will be covered.

During the Training-phase, the Statistical Machine Translator creates two models, as follows:[17]

(8) (i) The Language Model: the likelihood that the output sentence is a valid sentence in the target language. *P(T).*[18]
(ii) The Translation Model: the likelihood that the output sentence corresponds to the meaning provided in the source language's input. *P(S|T).*

The first model looks only at the target language's corpora, and therefore only contains words and phrases which appear in those corpora. What this model does, is assign values to separate words or phrases, based on the frequency of their appearance in those corpora. For logical efficiency reasons, not each separate word or separate phrase is literally valued, but rather within the context of particular strings which occur in the corpora.[19] Following the examples given by Hearne & Way (2011), a simplistic Language Model would be one which assigns a value to each separate word in following string, "*I need to go to Berlin*". This would result in a unigram model as follows in (i), followed by example calculations from possible translation queries thereafter:[20]

(9) (i) P(I) = $\frac{1}{6}$ P(need) = $\frac{1}{6}$ P(to) = $\frac{2}{6}$ P(go) = $\frac{1}{6}$
(ii) P(I need to go to [_]) = $\frac{1}{6} \cdot \frac{1}{6} \cdot \frac{2}{6} \cdot \frac{1}{6}$
(iii) P(I need to go to [_]) = $\frac{2}{1296}$
(iv) P(go [_] to to need I) = P(I need to go to [_])
(v) P (to to to to to to) = $\frac{32}{1296}$

[17] Hearne & Way (2011). pp. 207-221.
[18] These formulaic entities refer back to those presented in (27), as the P(A), P(B) and P(A|B) as presented in (25) and (26).
[19] By saying 'string', I mean a sequence consisting of a set length of words and/or phrases.
[20] Taken from Hearne & Way (2011) page 209. Although they use the sentence *I need to fly to London tomorrow*. Also take into account that for this example, this string represents the entire corpus – it does not know any other words or phrases than this particular string.

(vi) P (I need to swim to) = $\frac{1}{6} \cdot \frac{1}{6} \cdot \frac{2}{6} \cdot \frac{0}{6} = 0$

In the examples presented in (9), each word is given a value based on their frequency within this string. All except for 'Berlin', for Berlin is not a true lexical item. Due to the word '*to*' appearing twice in the sentence, it of course gets a higher value than any of the other words. These values are multiplied, and thus the final probability of translating this sentence is $\frac{2}{1296}$. However, as presented in (iv-vi), there are some problems with this system as it is. For example, in (iv) we can see that word order does not play a part in this probability calculation. Also, what (v) demonstrates, is that such a string (which does not make any sense nor is grammatical) will have a higher probability of being a requested translation than the original example sentence, because of the high frequency of high-valued words. And finally in (vi), presenting the Language Model with new words, such as 'swim', will result in a probability of 0.

The problem within this model as presented in (vi) can be solved in one of two ways: increasing the size of the corpus with more words, or by reserving a set value for unknown words. As long as that value is larger than 0, a zero-probability will not occur any more. These are both methods used in order to increase the power of Statistical Translation Machines. Resolving the issue as presented in (v), however, requires the need of making the n-gram larger. As mentioned above in (9), this was a representation of a unigram model – a model where the '*n*-value' is set to one. This means that in the example as presented above, all the separate words were compared to the relation between themselves, rather with the other words in the string. This method will take into account the probability of a select number of words appearing after one and other in the string.[21] For example, when using the same example corpus of "*I need to go to Berlin*" as above, but now with a bigram model, we would come to the following:

(10) (i) P(need|I) = $\frac{1}{1}$ P(to|need) = $\frac{1}{1}$ P(get|to) = $\frac{1}{2}$ P(to|get) = $\frac{1}{1}$ P([_]|to) = $\frac{1}{2}$

(ii) P(I need to get to [_]) = $\frac{1}{4}$

(iii) P (to to to to to to) = $\frac{1}{64}$

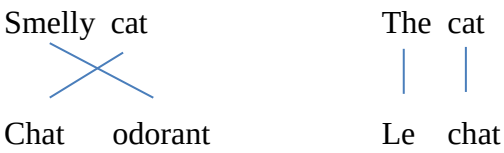
[21] The basic formula for *n*-grams is $\frac{n}{n-1}$. With here, '*n*' representing the amount sequential words in a string.

By using a bigram model, we can be sure that the sentence "*to to to to to to*" has a lower chance of being produced. Increasing the n-gram even further, would give other results. However, the length of the 'n' has its practical limitations when it comes to Statistical Translation – for a too high a value of '*n*' would lead to only allowing very long strings. This would return to the problem we had earlier with unknown words leading to a value of 0. By using these methods, the Language Model can generate probability values to words and phrases from the target language's corpora.

The second model that needs to be generated during the training-phase, is the Translation Model. As described in (8), the Translation Model's task is to calculate the probability that what is presented as at the translated output of the Translation Machine is a supposed translation of the source language's input. Contra-intuitively, the Translation Model does not calculate the probability that of what the produced translation from the source language should be, but instead the probability of the source language's input corresponds with the target language's output. It is a reversed model for translation.[22]

Another part of the Translation Model's task is to ensure a likely word alignment in the translated output. Here, word alignment refers to the alignment of word pairs from the two different data sets. That is to say, the corresponding word from the source language to the target language. According to Hearne & Way (2011), a much used method for statistically deducting word alignment is the Expectations-Maximization algorithm by Dempster et al. (1977).[23] This algorithm requires that strings from one set of data (e.g. the corpora of the source language) and strings from another set of data (e.g. the corpora of the target language) are compared for the probability that these correspond to each other by iterating their probabilities in two separate steps: a so-called Expectations-step and a Maximization-step. Below in (11) a hypothetical example set of data is presented to clarify this algorithm.[24]

(11) (i) **Word alignment**



[22] Hearne & Way (2011). p. 211.
[23] Hearne & Way (2011). p. 214.
[24] These examples are based on the examples given in Hearne & Way (2011). p. 215.

(ii) **Probabilities**

		(Init)	*0*	*1*	*2*
P(smelly|chat)	=		$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{6}$
P(cat|chat)	=		$\frac{1}{3}$	$\frac{1}{2}$	$\frac{4}{6}$
P(the|chat)	=		$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{6}$
P(cat|odorant)	=		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
P(smelly|odorant)	=		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{2}{3}$
P(cat|la)	=		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
P(the|la)	=		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{2}{3}$

In this example, we have the source language data in English consisting out of two strings, "*Smelly cat*" and "*The cat"*. Next to this data, we also have the corresponding French data "*Chat odorant*" and "*Le chat*". In (i), the two pairs have been placed together, with lines representing the proper translations. Following in (ii), probabilities of the French translation corresponding with the English source per word pair are presented, as well as the probabilities after the first so-called *initialization* (*Init* in (11)) below *1* and the second initialization below *2*.

As mentioned above, these iterations happen in two steps, the Expectations-step and the Maximisation-step. During the Expectations-step, we multiply the probabilities of each word's possible word pair given in a string, and divide this by the total sum of probable word pair-combination. So from the data presented in (11), this would mean that from the string "Chat odorant", we have the word pairs <smelly|chat, cat|odorant>, which would produce $\frac{1}{3} \cdot \frac{1}{2} = \frac{1}{6}$. The same goes for the other possible <cat|chat, smelly|odorant>, which also produces $\frac{1}{6}$. The final exercise in the Expectations-step would be to then divide the probability of the single word pairs by the total probabilities, resulting in: $\frac{1}{6} / \frac{2}{6} = \frac{1}{2}$. Following the

Expectations-step, closely follows the Maximisation-step which focusses on the frequency of these possible outcomes. This is achieved by adding up all the probabilities of possible word pairs for a particular translation. In the case of *chat* from (11), this total amount of probabilities is 2.[25] This number is then used to divide the outcome calculated during the Expectations-step. This results in $\frac{1}{4}$, as presented in (11) under *1*, finishing the first iteration. The more often these two iteration-steps are done, the more accurate the model will eventually become. Although, in every new iteration step, the probability values from the previous iteration are used, instead of the original values as presented under *0* in (11). As demonstrated in (11), after two iterations, we can see that the probability for translation of *cat* being *chat* is significantly higher than other possible translations. The same can be said about the other words, although these might require more iterations to become more likely.

After word-alignment, the next step is to order the words properly in an output phrase. This is done by aligning the words to their position in the phrase from both the source language and the target language in both directions. That is to say that these alignments are configured both from source-to-target as well as from target-to-source. A method to do so is by using Phrase-alignment heuristics.[26] This method uses four steps: the word alignment in the target language, the alignment in the source language, the intersecting word alignments and finally a suggested outcome which is built around the intersecting word alignment.

Part of the training process, is the earlier-mentioned use of evaluation of the Translation Machine. Generally speaking there is one widely-used technique for evaluating Machine Translators: the MERT (Minimum Error Rate Training) technique.[27] Often, this technique uses what is known as the BLEU (Bilingual Evaluation Understudy) metric. Depending on what the outcome is of the BLEU metric on the delivered translations by the Translation Machine, the λ-values can be changed in order to possibly improve various features which the Translation Machine can focus on (see the log-linear model in (6)). These features are not sentence-related features, but rather the numeric values from the various calculations as have been presented in the paragraphs above.

The BLEU metric is a metric which compares the output of the Translation Machines to that of other translated work, preferably of the same input text. This translation work is usually

[25] $(\frac{1}{3}+\frac{1}{3}+\frac{1}{3})+(\frac{1}{2}+\frac{1}{2})=2$
[26] Hearne & Way (2011). p. 217.
[27] Hearne & Way (2011). p. 219.

done by trained translators rather than by other Translation Machines. This BLEU metric has its own set of formulae used in order to score the output of a Translation Machine compared to its reference translations. However, an extensive explanation hereof is not needed for this paper and will therefore be left out. In brief, the BLEU metric compares an *n*-set of words from the Machine Translation with its reference translations in order to calculate whether or not these correspond to each other. The higher the outcome, the more precise the Machine Translation's work is. The higher the *n*-amount of words compared, the more probable it becomes for the score to lower. Thus a BLEU result for a translation with a high *n*-amount implies a very precise Translation Machine.

The training-phase of the Translation Machine should now be at an end, allowing to continue to the decoding-phase of a Translation Machine. As long as the training-process has been successful, it should not be a difficult task for a Translation Machine to translate a given string of words to the target language for its lexical choice-phase. However, the difficulty lays in selecting the right (most probable) outcome from the calculations the Translation Machine has made. Alongside this selection problem, there is also the issue of processing abilities. To ensure that the Translation Machine does not overproduce the probable translation. For example, using the data in (11), it is apparent that *odorant* is the least likely option for *cat*. Even so, the word is still an option to be a translation of *cat*, how improbable it might be. In order to ensure that the Translation Machine works as efficient and effective as possible, it is preferable that the Machine chooses to ignore the option of *odorant* whenever it is asked to produce a translation of *cat*. Of course, in this particular example, it is evident that this translation is faulty. However, there can of course be other examples where a less probable option is in fact the proper translation due to its rare or obscure context.

A solution for this is by translating the input source text in steps. Starting out with a small n-gram value, and then start over again, using a higher *n*-gram value and so on. Until the increasing of the *n*-gram value does not deliver any new results any longer from the Translation Model. A variation of this is so-called "beam-search decoder".[28] This method starts out with a select number of so-called hypotheses (possible translations) from the target language. The number of these hypotheses are very high, and not all of these have to actually correspond with the source language's input. It is just a 'ready' selection of possible translations from the target language for any sentence. Next, when the Translation Machine is

[28] Hearne & Way (2011). p.222.

actually translating, it produces a new hypothesized translation based on the calculated probabilities from the input. However, a new hypothesis can only be added if the probability-score of that hypothesis is higher than the previous one. The ending result is, that it could be possible that even before the actual translation process starts, the Translation Machine already has a correct hypothesis at the ready in its 'beam' (initial selection of hypotheses) for the input. If not, there is a likely chance that parts of the input sentence can be translated from (parts of) the hypotheses in the beam. From that starting position, the Translation Machine continues on to try and create a better hypothesis, until it cannot produce a hypothesis with a higher probability.

Although this process could already cover the step of reordering, Farzi et al. (2015) state that, as the name slightly suggests, reordering involves the task which should ensure the most grammatically accurate sentences post-translation. It is within this sub-task where the most difficulties with Machine Translation lies, and where the highest differentiation between various Machine Translation-methods occurs. As with translation, this task is mostly dealt with from a statistical standpoint. The phrase-based model described by Mukesh et al. (2010) called 'The Moses translation toolkit', for example, uses the language model-approach as described above from Hearne & Way (2011) to re-order the words into a grammatical order.[29] Other models, such as the model suggested by Carter & Monz (2010), use a basic form of a syntactic tree in order to help reorder the translated phrases better. They manage to do so by formalizing an algorithm which tests the various translated word orders to see which order is the most probable. The algorithm makes use of so-called POS tags which have been assigned to words in order to test the word order's probability.[30] Even though their methodology is indeed able to reorder words based on their linguistically-inspired algorithm, they fail to produce correct sentences due to their approach to syntax is fairly incomplete and still rely a lot on statistics.

Farzi et al (2015) also add syntax to their reordering methodology. Their approach relies more on configuring something which seems to resemble a basis outline of a general syntactic tree upon which the translations are placed. Their approach is based on the criticism they have

[29] Mukesh et al. (2010)

[30] Carter, S & Christof Monz. (2011). Syntactic discriminative language model rerankers for statistical machine translation. *Mach Translat, 25*, 317-339.

on other Machine Translations – namely that too few Machine Translation models make any use of the internal constructions; the syntactic constraints. They do point out that newer models do, but these tend to be especially unsuccessful when it comes to translating sentences which rely on long-distance syntax. Costa-Jussa & Farrus (2014) share this opinion; (Statistical) Machine Translation relies too little on linguistic properties and rules when translating, which causes most models to quite often deliver fairly poor translations. In the following section I will address these issues further.

2.4 An Analysis of Statistical Machine Translation Applications

In the introduction of this thesis, several examples of translations from Dutch to English were presented. With the general approach of Statistical Machine Translation now being established, it is possible to analyze these example sentences and the translations generated with the help of the applications and point out why the translated sentences were generated in such a fashion. Below, in (12), some of these example sentences have been reproduced:

(12) (i) Ik ken de man die jij gisteren zag niet.

I do not know the man you saw yesterday.

a. I know the man you saw yesterday. (Google Translate™)

b. I know the guy you know yesterday saw not. (Microsoft Translator™)

(ii) Ik hoor jou niet.

I can't hear you.

c. I can not hear you. (Google Translate™)

d. I hear you not. (Microsoft Translator™)

(iii) Ik kan jou niet horen.

I can't hear you.

e. I can not hear you. (Google Translate™)

f. I can not hear you. (Microsoft Translator™)

(iv) Ik ken de buurman niet.

I do not know the neighbor.

g. I know the neighbor not. (Google Translate™)

h. I know the neighbor not. (Microsoft Translator™)

(v) Ik werk in het weekend niet.

I don't work on weekends.

i. I do not work on weekends. (Google Translate™)

j. I work in the weekend is not. (Microsoft Translator™)

(vi) Ik geef mijn cadeau niet.

I will not give my gift.

k. I give my gift not. (Google Translate™)

l. I give my gift not. (Microsoft Translator™)

First, when looking at the unsuccessfully translated sentences, we can assume that these are all examples of Adequate Disfluent Translations; the words are mostly properly translated, it is the positioning of the words in those translations which fail the translation. The question is why this happens. Although the exact statistical formulae used by either of these two translation apps is unknown to the general public, it can be assumed that they too use methods similar to those discussed in the previous section. However, both applications have a visual user-feedback mechanism which presents the user what part of the target language's translation is paired with the source language's input. This way, we can assume what n-gram these translation applications could possibly have used, as well as where in the training-phase of the application it might have gone awry.

Below, the visualization of (12(i)) is given in Google Translate™:[31]

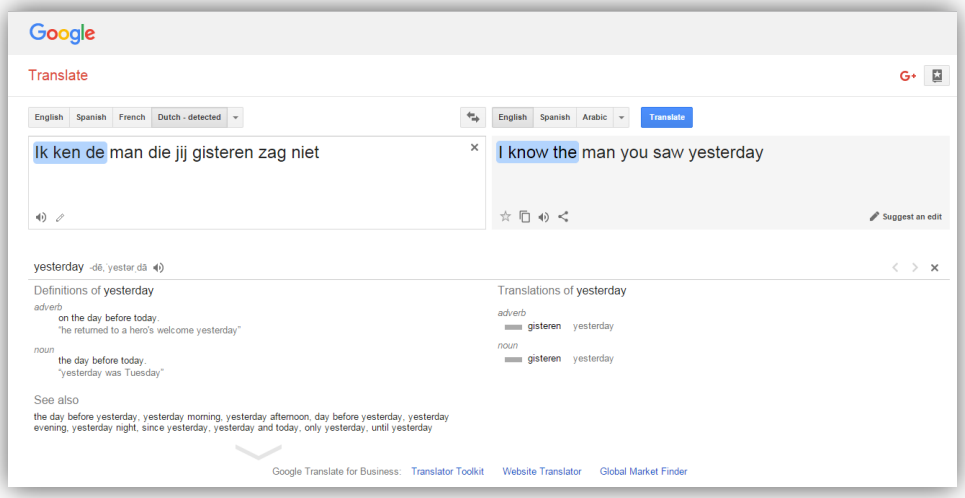


figure 1: sentences (12(i)) and (12a) in Google Translate™, highlighting the first three words

[31] From http://translate.google.com/

The visualization here, where a segment is highlighted in blue when hovering over either words from the source language or the target language shows the corresponding segment. Within the application, this feature is meant to either select possible alternatives for that segment, or to provide the application with alternatives, which it can add to its corpora. For the purposes of this thesis, however, it shows the n-grams the translator used. In figure 1, it is shown that the segment '*Ik ken de*' corresponds to '*I know the*'. What this tells us, is that during the word alignment process, these two chunks of three words seem the most likely to correspond to one and other, in light of the input given. Below in figures 2 and 3, we see the corresponding word segments from the rest of the sentences.

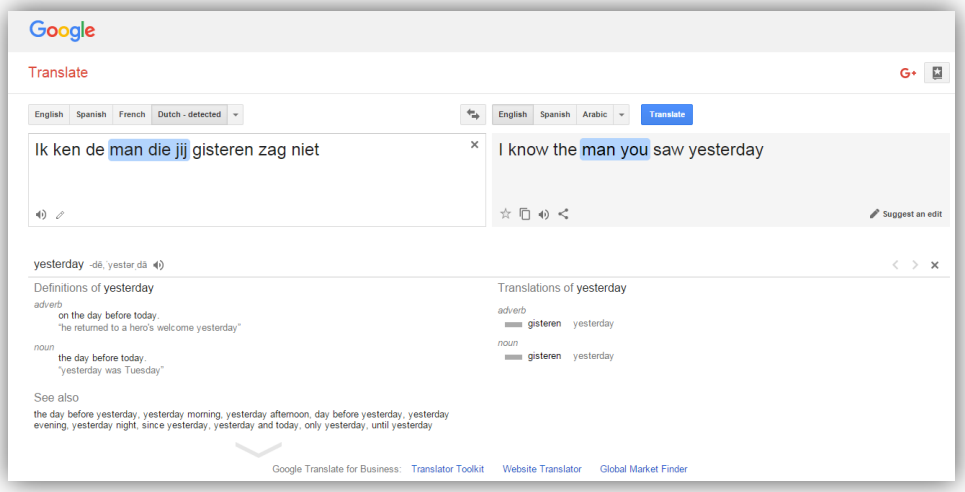


figure 2: sentences (12(i)) and (12a) in Google Translate™, highlighting the second segment.

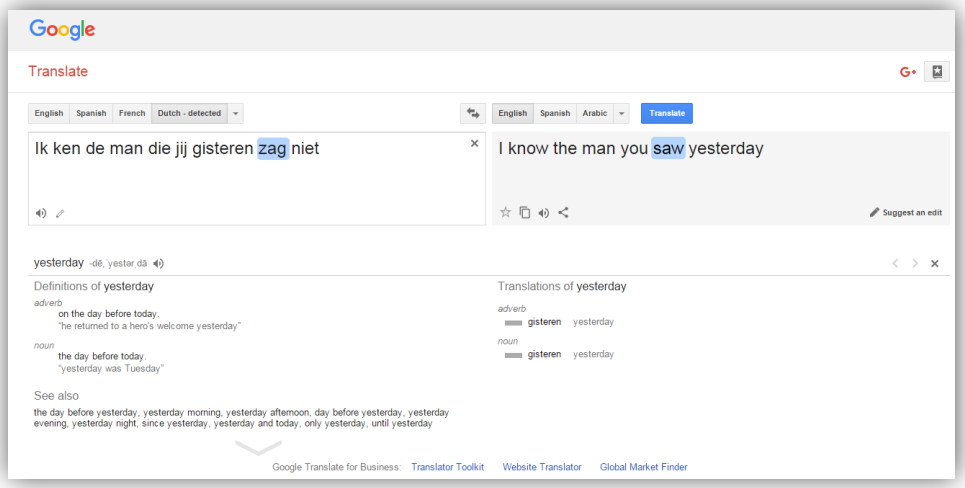


figure 3: sentences (12(i)) and (12a) in

Google Translate™, highlighting the third segment.

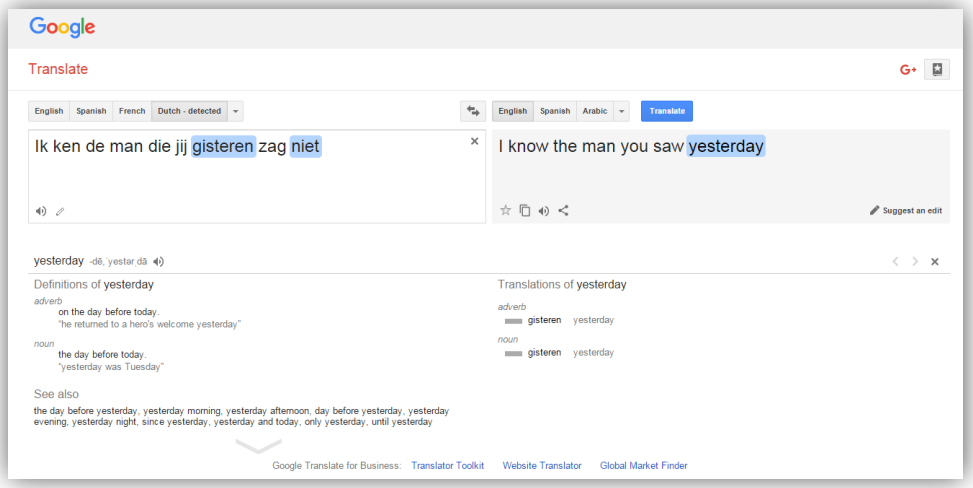


figure 4: sentences (12(i)) and (12a) in
Google Translate™, highlighting the final segment.

What is striking about figure 4, is that Google Translate™ seems to correspond '*gisteren*' and '*niet*' to '*yesterday*'. It could be that when this particular application is faced with a very low possible option for a translation, that it will not produce a translation, or that not including the word '*niet*' in this translation results in a high probable outcome of a correct translation for this application. This is, however, pure speculation, as the full functionality of this application is unknown. However, when using this visual feature for selecting possible alternatives, options as presented in figure 5 are shown which include a translation for the Dutch '*niet*', although these alternatives still do not provide a proper translation of (12(i)).

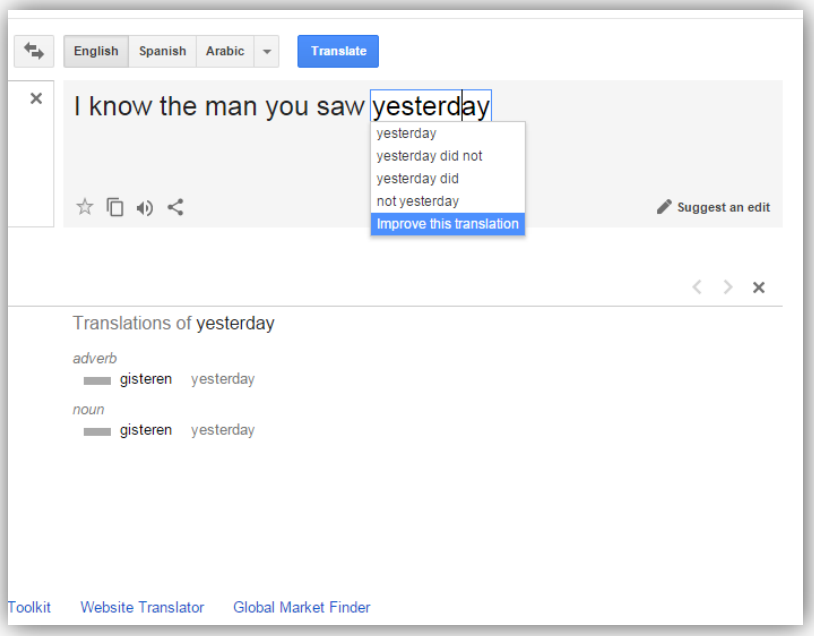


figure 5: alternatives for 'yesterday' in sentence (12(i))

Because the sentence in (39(i)) is one which might not be used frequently in everyday life, or is used in a lot of written works, it is understandable that within large corpora, it is highly unlikely that a large n-gram number could be successful when translating this sentence. Another example when using Google Translate™ presented in figure 11, shows that it is possible for the application to use higher a higher n-gram in order to come to a successful translation.

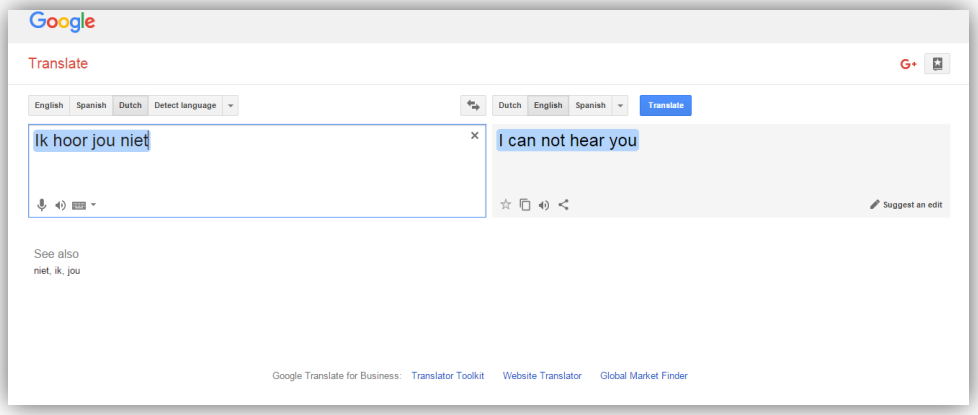


figure 6: a translation of sentence (12(iii)) when using Google Translate™.

The sentence used in figure 6 or (12(ii)), is, however, a sentence which can be expected to be used more regularly. Hence, the odds of finding these exact same words in this exact word order is more likely, as well as finding a correlating translation for this sequence. This explains why the use of a higher n-gram will result in presenting a correct translation due to its high probability.

When comparing these results with the other translation application, Microsoft Translator™, it is clear that the mechanisms used in both applications are indeed fairly similar, and correspond to the description of Statistical Machine Translators. The two figures shown below demonstrate the similarity between the two translation applications used.

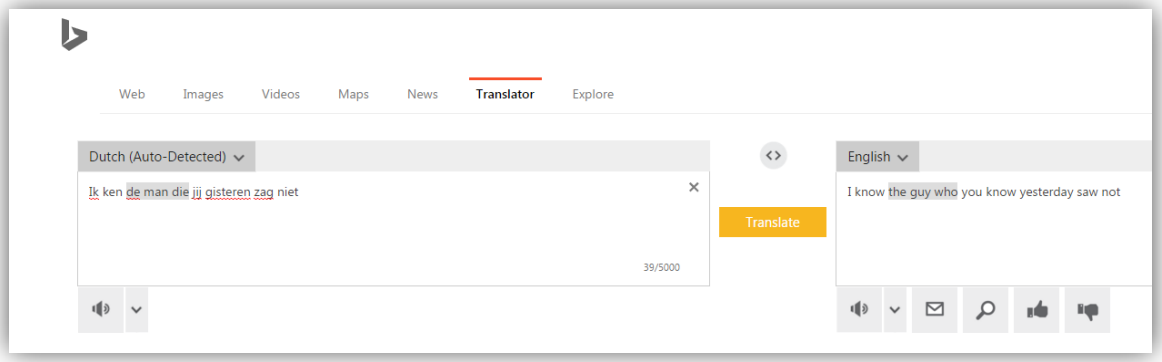


figure 7: Microsoft Translator™ demonstrating what segments from (12(i)) he uses to correlate with its possible translation after beam-search-encoding.

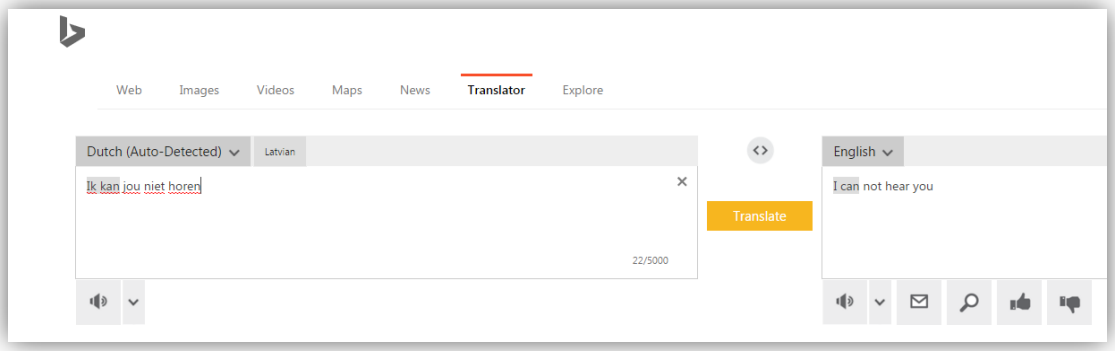


figure 8: unlike Google Translate™, Microsoft Translator still uses a smaller n-gram for sentence (12(iii)).

As figures 7 and 8 show, small *n*-grams are used in both translations in order to come to a final translation. What can be assumed, as figure 8 shows, is that the latter translation application uses a different protocol in its search-beam-encoding, or uses different corpora for its source and target language where the Dutch sentence of "*ik kan jou niet horen*" and the target translation of "*I can not hear you*" do not appear as often alongside each other, resulting lower probabilities when trying to correlating in the entire string as a whole to another full string. The same reasoning can be used to describe the resulting (In)Adequate Disfluent Translation which is presented in figure 7 and (12b)).

This is, however, pure speculation on how the application seems to work, based on the general methodology of Statistical Machine Translation described in the previous chapter. Without knowing the precise specifics behind the programming of these applications, it is impossible to know *exactly* how their translation process works; however, looking at the examples above, it seem highly likely that the mechanisms used do not stray too much from the description used previously. The translation applications both cluster a select number of sequential words from the input sentence, and correlate that string of words with a seemingly corresponding string from the target language's corpus data. Another sense that these two translation applications are either entirely or mostly statistically-driven, can be seen in the translated sentences presented in (12). Whereas nearly all the words are translated correctly, it is the placement and ordering of these words in the translated output sentences which are lacking. Where a large chunk of the sentences can be translated correctly, (12) shows that when a particular word is placed in an ambiguous spot, or outside of a 'standard' sentence – such as the negated '*niet*' at the end of the input sentences – the applications seem to fail to place the word correctly in the translated output. The exceptions are when the application is able to use a large string which includes either most of the words, or the sentence as a whole, as presented in figure 11, or below in figure 14.

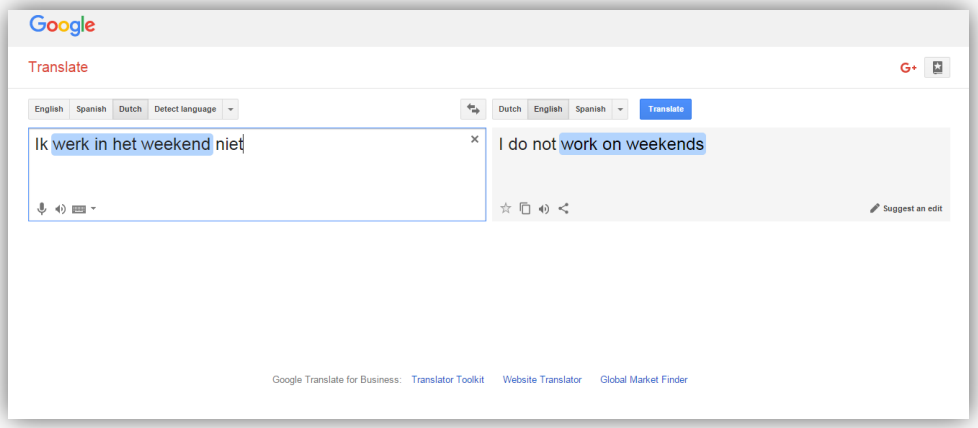


figure 9: Google Translate™ is able to use two-thirds of the sentence as a single string

Finally, one aspect of Statistical Machine Translation remains unknown when analysing these applications from the surface: the weight of λ_m on particular features in the formula presented in (6(ii)) of the Log-linear model. Depending on how the weight λ_m is defined, the output of the applications will be different. Due to analysis of these applications only being possible on a surface-level, it is impossible to determine what the value of λ_m is. Some of the examples presented here could be influenced by this value, and thus creating different output sentences than when solely the probability of the *n*-grams would be used.

Chapter 3: Linguistic Theories on Negation

3.1 English and Dutch sentence structures

When discussing the syntax of two languages, especially with the purpose of systematically translating from the one language to the other, the difference in word order between Dutch and English has to be discussed. Although both languages are historically speaking Germanic languages, their modern forms vary in sentence construction. English is a Subject-Verb-Object language (SVO), whereas Dutch is both a SVO language as well as a Subject-Object-Verb (SOV) language. According to Zwart (2011), Dutch can also be interpreted as having a SVOV word order at times.[32] Unlike English, Dutch is a verb-second (V2) language. This entails that in Dutch, finite verbs can appear in the second position of a main clause. To be more exact, the position which directly follows the first constituent.[33] Zwart's (2011) analysis on Dutch is as follows: Dutch is head-initial, has a SOV word order in embedded clauses and a SVO word order in main clauses.[34]

According to Haegeman (1995), the V2 phenomenon seen in Dutch is caused by V-to-C movement.[35] This is when the V moves from below in the clause to the front ([Spec, C]) to fill a otherwise vacuous C-position. This will be discussed further in section 3.4.

3.2 Negation and Scope

First, for the purpose of this paper, it is important to describe *what is negation* and *what framework* will be used to approach negation from a syntactic (and later semantic) perspective. For what negation is, exactly, is difficult to say. In their introduction, Morante and Sporleder (2012) quote Lawler who state, from a cognitive perspective, "[negation] *involves some comparison between a 'real' situation lacking some particular element and an 'imaginal' situation that does not lack it."*[36] To illustrate what this quote from Lawler means, is that

[32] Zwart, J.W. (2011) *The Syntax of Dutch*. 1st ed. Cambridge: Cambridge University Press, 243-280. Cambridge Books Online. Retrieved from http://dx.doi.org/10.1017/CBO9780511977763.011
[33] Zwart, J.W. (2011). *The Syntax of Dutch*. 1st ed. Cambridge: Cambridge University Press, 281-295. Cambridge Books Online. Retrieved from http://dx.doi.org/10.1017/CBO9780511977763.012
[34] Zwart, J.W. (2011), p. 266.
[35] Haegeman, L. (1995) *The syntax of Negation*. Cambridge: Cambridge University Press. pp. 112-115.
[36] Morante, R. and C. Sporleder. (2012) Modality and Negation: An Introduction to the Special Issue. *Computational Linguistics*, 38(2), 229-231.

negation can only be used in a comparative way to something which is a 'positive'. For example, in (13):

(13) a. John is here.
b. John isn't here.

The negation in (13a) is a statement which implicitly compares itself to the statement made in (13b). That is to say that if the statement made in (13a) is true and real, then such a statement would only make sense if it can compare itself to an opposite, positive situation. In this case, the presence of John.

What the sentences in (13) also demonstrate is what Morante and Sporleder describe as: 'clausal negation'. That is to say that the entire preposition is negated by the influence of a form of negation (what this form is exactly will be described below). Because, in (13a) the preposition of the presence of John is negated, resulting in there not being a presence of John. A different, yet related form of negation that Morante and Sporleder mention is that of 'constituent negation'. An example of this can be found in (14):

(14) a. Mary has got sufficient income.
b. Mary has no sufficient income.

What occurs here is that not the entire clause is negated. The preposition on Mary having some degree of income is not what is negated here – only the constituent 'sufficient' is negated.

The proper terms for what Morante and Sporleder are demonstrating in their introduction, are 'scope' and 'focus'. For when a negative element (or 'negator', or 'negative marker') – the word *not* in the cases of (13b) and (14b) – is added to a sentence, it is the element's scope or focus which determines what and how much is actually negated in the sentence, and how the sentence then is to be interpreted correctly.[37] So the differences between the previously mentioned clausal negation and constituent negation are issues regarding scope. However, as Van der Auwera (2001) exemplifies, there are instances where the scope of a negator may

[37] Van der Auwera, J. (2001) Linguistics of Negation. In *International Encyclopedia of the Social & Behavioral Sciences*. 10462-10467. Retrieved from http://www.sciencedirect.com/science/article/pii/B008043076702965X.

indeed be the entire sentence – its focus might not be. Whereas the scope of negation can be made syntactically clear (see chapter 2.4), the focus can often remain syntactically vague. According to Van der Auwera (2001), focus can often be the subject of constituent negation.

Another form of negation that Morante and Sporleder list in their introduction, is that of Negative Polarity Items (NPIs). These NPIs are, as Morante and Sporleder describe it, terms which 'act differently' around negation. Their description is quite limiting, but in practice, NPIs are grammatical polarity items which can affirm or negate the context of a sentence. Take the examples in (15):

(15) a. I never sleepwalk at all.
b. *I always sleepwalk at all.
c. I haven't ever sleepwalked.
d. *I ever sleepwalked.

The sentences in (15) show that the NPIs ('at all' and 'ever') can only appear in sentences which (already) contain a negative item ('never' and 'not'), and when the negative item is missing, that these cannot appear, as in (15b) and (15d).
Morante and Sporleder go on to discuss other areas of negation, but for the scope of this paper, only the three types of negation matter, for these contain clear negative-markers, rather than being open to interpretation on whether or not these are positives or negatives, nor do these types contain items concerning with oppositions (polarity) or antonyms. In other words, the discipline of pragmatics lay outside the scope of this paper.

Theories on negation are not confined to the realm of linguistics. Negation has existed in the realm of logic long before linguistics became a subject. For in logic, negation is often referred to as a type of opposition - a tool through which contraries and contradictories can be expressed. For example 'unhappy' is the contrary negation of 'happy'. For unhappy is a state which is the exact opposite of happy (namely, sad). The statement 'not happy' is, however, not (necessarily) the same as 'unhappy'. Instead, 'not happy' is the contradictory negation of happy. For 'not happy' is virtually everything else to what happy is.[38]

[38] Example taken from Van der Auwera, J. (2001).

Returning to linguistics, 'unhappy' is a negated form of a 'happy' which is negated through the morphologically added prefix 'un'. From a linguistic point of view, the difference beotween the morphologically negated happy and the constituent negation of 'not happy' is that 'unhappy' is not only semantical in nature,[39] but also syntactical.

This paper will however focus on the use of explicit negative markers (mainly the negator '*not*'), rather than morphological negation or NPIs.

3.3 Negation and Polarity Items

As Haegeman (1995) is quick to note, negation in sentences seem to resemble characteristics of WH-movement and Rizzi's WH-criterion.[40] Namely, interrogative elements license polarity items, and so do negative elements (as discussed above). For example, take the sentences from (15) and compare them with similar interrogative sentences in (6) below:

(16) a. ?Do I never sleepwalk at all?
b. Do I always sleepwalk?
c. Haven't I ever sleepwalked?
d. Have I ever sleepwalked?

Next to the polarity-items now being licensed by the interrogative elements, we can also notice that it does not matter any longer whether or not a negative or positive element is present. Only (16a) seems to require a specific context in order to be acceptable.[41]

Haegeman points out that it is proposed that polarity items are licensed through c-command by a negative or an interrogative element. We can show this through the examples in (17); here, the polarity items are placed in sentences which do contain either a negative or an interrogative element, however these do not c-command the polarity items.

[39] Not happy tends to be interpreted more as a contradictory negation, whereas unhappy tends to be interpreted more as a contrary negation. See Van der Auwera, J. (2001).
[40] Haegeman, L. (1995), p. 71.
[41] This sentence seems to require a follow-up sentence or clause regarding circumstances when one would (in this context) 'always' sleepwalk. For example, after (4a) a sentence such as "Not even when I'm wearing blue pajamas?" seems to be necessary. But it could be that in this particular sentence, the polarity item (at all) is c-commanded by 'never' rather than by the interrogative element (inverted 'do').

(17) a. *Anyone did not kill John.
b. *Anyone did kill John how.
c. *Anything did John buy.

Due to the polarity items ("*Anyone*" and "*Anything*") not being licensed in the examples (17a-c), these sentences are ungrammatical – disregard there being a negative or interrogative element in the sentence. When we place these elements in the proper c-commanding positions for the polarity items, we will see that these sentences become grammatical again:

d. Didn't anyone kill John?
e. How did anyone kill John?
f. Did John buy anything?

What we can also see in (17d) is that, apparently, the two elements (both interrogative and negative) seem to c-command the polarity item 'anyone'. This will, however, be explained later on, for now it is enough to assume that these elements now c-command the polarity item 'anyone'.

The second similarity between interrogative elements and negative elements that Haegeman points out, is that both elements seem to be able to trigger subject-auxiliary inversion. This entails that an auxiliary verb, such as 'to be' or 'to do' will take the subject's position in the (root) sentence. In other words, the auxiliary and subject switch places in the sentence-order. For example, take the sentences in (18):

(18) a. You saw what. (or: "*You did see what")
b. What did you see?
c. You greet the queen like so. (or "You do greet the queen like so.")
d. Never do you greet the queen like so.

Haegeman does point out that in case of negative elements, this subject-auxiliary inversion does not *always* occur. Especially when the negative elements are sentence-initial, as the sentences in (19) demonstrate:

(19) a. Not everyone has a good pair of shoes.

*b. Not has everyone a good pair of shoes.

It is this difference between the triggering of a subject-auxiliary inversion and the lack thereof through which Haegeman makes the distinction for clausal negation and 'local negation'.[42] Clausal negation not only triggers subject-auxiliary inversion, but also, as the name does suggest, invokes that the negative element has the entire sentence as its scope. Unlike the inversion-free local negation which, as is to be expected, only has a fragment the sentence as its scope.

Referring back to the licensing of polarity-items, Haegeman states that negative sentences without inversion (local negation) cannot have their polarity-items licensed by the negative elements.[43] In the examples presented in (20), this lack of licensing is illustrated:

(20) a. Not often do you say *something/anything to Linda.
b. Not long ago John bought something/*anything for Linda.

What can be suggested from Haegeman's statements so far, is that it seems that whenever a negative element appears a head-position, rather than a specifier-position in the sentences (initially), the sentence has clausal negation. Whenever it is the opposite, and thus the negative element appears in a specifier-position (initially), the sentence has local negation.

3.4 Negation and WH-Movement

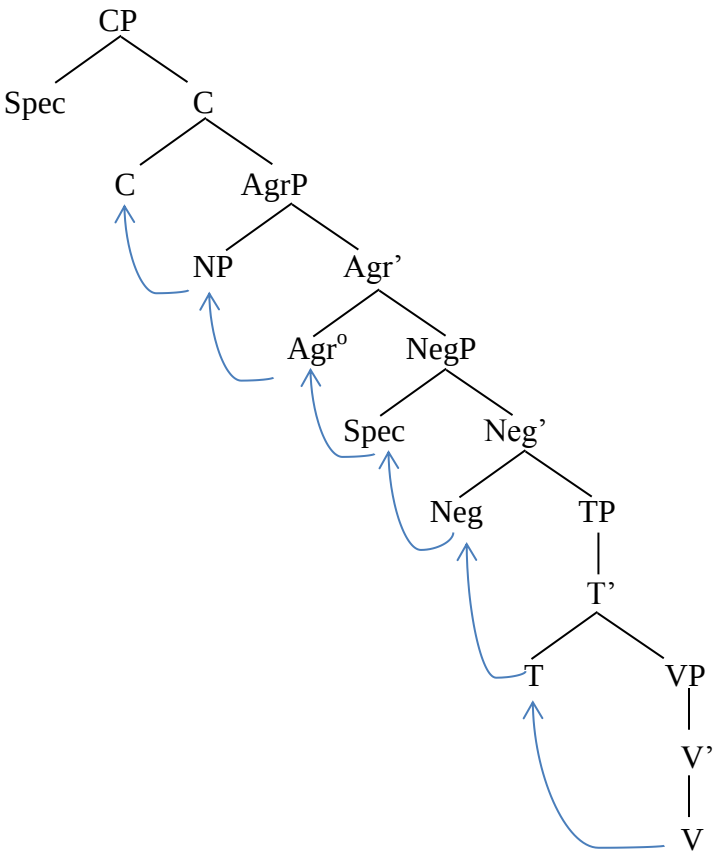
As mentioned above, according to Haegeman (as well as Zanuttini, R. and Rizzi, L.)[44] negation elements and interrogative elements seem to have various syntactic characteristics in common. Next to their behavior with polarity-items, it seems as if the negation elements undergo a movement through the sentence structures which is similar to the movement interrogative elements make, known as WH-movement.

Within the Generative Syntax framework, movement of elements is not uncommon. As briefly discussed in 3.1, Dutch is a V2 language which is caused by V-to-C movement.

[42] Haegeman, L. (1995) p. 72. Although Haegeman does not mention the term 'clausal negation', but rather talks about 'negative sentences' and later on to 'sentential negation'.
[43] Haegeman, L. (1995) p. 73.
[44] This deducted from the fact that not only Haegeman herself refers back to them a lot in her own work, but also due to Haegeman having written papers alongside them on this topic.

Movement of elements (specifically those in head positions) occurs in steps. According to Haegeman (1995), the verb moves up the tree from V to T to Agr to finally set in C.[45] However, NegP dominates TP first. Which means that V will have to move through NegP first, to land in front of the negative marker. For her book on West-Flemish and its Negative Concord[46], this is a key feature on why negation can occur both in front and behind a verb within a sentence. For Dutch, however, this explain why in the examples given in chapter 1 have negation at the end of the sentence – after the verb – whereas English would require the negative marker to occur in front of the verb. Below, in (21), an exaggerated version of how head-to-head movement works: the verb moves up the tree via other head positions (usually vacant) until it reaches the final position.[47]

(21)



In some cases of movement, elements alter due to their movement through specific positions. For example, the moved element can leave 'drops' behind in empty positions, or other

[45] Haegeman (1995), p. 115.
[46] Simply put, double negation.
[47] Configured using Haegeman (1995) p. 115 as an example.

elements occupying the head-positions can latch on to the moving element – as is the case with Haegeman's analysis of West Flemish.[48]

One of the key elements regarding WH-movement is whether or not an element can move. For now, the reason as to *why* elements move will be left for a later stage. First it has to be explained whether or not elements can move. Movement of categorical syntactical elements occurs through head-to-head movement. This entails that the heads of phrases can only move to (and in an extent thereof, can only move via) other head-positions. Related to head-to-head movement is Rizzi's definition Relativized Minimality places some limitations to head-to-head movement to which elements should abide to. In (22) Relativized Minimality is defined:[49]

(22) **Relativized Minimality**

X x-governs Y only if there is no such Z such that:

a. Z is a typical potential x-governor for Y

b. Z c-commands Y and does not c-command X

What Relativized Minimality entails, is that a head X cannot move to a position to govern Y, if there is a head Z between X and Y which can govern Y just as head X can. Neither can the head Z already be c-commanding (x-governing) Y before movement. Consider (23) as an example of this:

(23) a. Who_i do you think [CP t_i [IP they will kill t_i?]]

b. *Who_i did you wonder [CP why [IP they will kill t_i?]][50]

c. Why_e did you wonder t_e [CP who_i [IP they will kill t_i?]]

As the subscripted 'i' demonstrates, in (23a) the interrogative element 'who' moves from t_i position from the IP, to a t_i position in CP to finally take the sentence initial position. However, this becomes impossible when there is another element in between the initial and final position of 'who' which can block its movement due to it being as good a potential x-governor as 'who' is ('why' in (23b)). The sentence in (23c) is acceptable, due to the element of 'why' moving to sentence-initial position, after which it leaves an opening for 'who' to

[48] See Haegeman, L. (1995) for a more detailed analysis.

[49] As defined in Haegeman, L. (1995) p. 43.

[50] Haegeman suggests a similar sentence to (10a), however, she leaves it questionable whether or not such a structure with who and why is grammatical. I am however of the opinion that this is not the case, and is simple ungrammatical.

move to. This movement of the wh-element is what is known as A-bar (or A') movement[51]. Movement of elements in (or to) so-called A-bar positions entail that those positions, and thus also the elements which house those positions, are not associated with a grammatical function. Primarily, this means that these positions generally do not house grammatical argument-holding elements, such as verbs or nouns. In more detail, A-positions can house elements which have to 'agree' with a head element based on such features as gender, number or case (φ-features). Haegeman argues that A-bar-positions rather agree with heads based on their operator functions – functions such as focus or wh, and also neg.[52]

However, as Relativized Minimality explains, the 'freedom' of A-bar movement is limited due to the possibility of government. As movement can leave empty spaces within sentences (see examples in (23), yet as the Empty Category Principle by Rizzi (1990) states: "*An empty category must be properly head-governed*".[53] As Haegeman (1995) explains, there are basically two types of government in place when it concerns moved elements (which have thus created empty spaces): namely, (i) binding and (ii) antecedent-government.[54] For binding, the definition is relatively simple:[55]

(24) **Binding**: X bind Y iff
- (i) X c-commands Y
- (ii) X and Y have the same referential index

This roughly entails that as long as X c-command Y, the empty space (or trace) left by movement will be properly governed. However, with antecedents, this lies somewhat more complicated as presented below in (25):[56]

(25) **Antecedent-governing:** X governs Y iff
- (i) X and Y are non-distinct
- (ii) X c-commands Y
- (iii) No barriers intervene

[51] Hornstein, N. et al (2005) *Understanding* Minimalism. Cambridge: Cambridge University Press. pages 144 and 321-322
[52] Haegeman, L. (1995) pp. 258-259. However, on page 77, Haegeman also points out that according to Rizzi (1990), quantifiers can also house A-bar-positions.
[53] From Haegean, L. (1995), page 41.
[54] Haegeman, L. (1995) p. 42.
[55] As taken from Haegeman, L. (1995) p. 42
[56] As taken from Haegeman, L. (1995) p. 42

(iv) Relativized Minimality is respected

If we were to return to the examples presented in (23), we can see that in (23b) that the sentence is also ungrammatical, for *why* is a (potential) antecedent for t_i, and thus *Who* cannot refer back to t_i. Especially when dealing with extraction-islands (as the examples in (23) were), these rules mentioned above are key to be remembered. According to Haegeman (1995), WH-movement is fairly similar, if not nearly exactly the same as NEG-movement. Which means that the rules described above are also valid for NEG-elements.

A final analysis on the behavior of WH-elements which, according to Haegeman (1995) are similar to that of NEG-elements, is that of scope (something which will be dealt with in more detail in chapter 3.5). Firstly, from Brody (1993), she takes the notion that scope is created through the means of a chain, which, unlike through movement, is created through coindexation.[57] Brody's theory suggests that these coindexation chains for WH-elements are created through non-overt operators which are lost at the Spell-out phase of the sentence, but are still intact during the LF-stage of a sentence. In other languages, with multiple WH-movement, it is possible that all the chains, including those of scope, will move up to adjoin with their operator, and will thus be spelt out at Spell-out as well. However, English and Dutch are neither one of such languages. Therefore, only one moved wh-element is spelt out in the sentence. However, Haegeman (1995) does mention that the scope position must be *"a left-peripheral A-bar-position"*.[58]

Haegeman (1995) lays out all of these rules for WH-movement, is because she needs describe what is called the WH-criterion, so she can develop what she calls the NEG-criterion. For the WH-criterion gives rise to the WH-movement, as she believes that the NEG-criterion should give rise to NEG-movement. She describes the two criterions as follows:[59]

(26) 1. **WH-criterion**

(i) A WH-operator must be in a Spec-Head configuration with an X-[wh].

(ii) An X-[wh] must be in a Spec-Head configuration with a WH-operator.

[57] Haegeman, L. (1995) pp. 49-50.

[58] Haegeman, L. (1995) p. 93-94; As part of the definition for the WH-criterion, following her so-called Affects-criterion which defines the agreements the WH-features and NEG-features.

[59] Haegeman, L. (2000). Negative Proposing, Negative Inversion and the Split CP. In L. Horn & Y. Kato (Eds.), *Negation and Polarity. Syntactic and Semantic Perspectives.* (21-61). New York: Oxford University Press.

2. **NEG-criterion**

(i) A NEG-operator must be in a Spec-Head Configuration with an X-[NEG].

(ii) An X-[NEG] must be in a Spec-Head configuration with a NEG-operator.

What these bracketed [NEG] and [wh] entail, are the semantic features of either an negative or interrogative nature to which operators, which contain a scope, wish to gain a relationship with. Therefore, according to Haegeman, movement is triggered in general, and the above rules discussed in the previous sub-chapter are to be followed when either a wh-element or a neg-element is to move leftward towards a Spec-Head position.[60]

Although Haegeman might explain to certain detail how and why negation is syntactically placed in the sentence as it is through a Minimalist Program methodology, it does not explain what the scope is of the negation. As Moscati (2010) points out, the position of the negative operator in a sentence, on a Surface level at least, does not always reflect its scope.[61] Moscati does seem to agree with Haegeman (and so also with Brody) on the sense of there being some sort negation-chain 'hidden' in the sentence which should link the negation marker with its lower position. Unlike the work from Haegeman (and Zanuttini), Moscati proposes the following regarding scope and the raising of the negative element in order to do so:

(27) **Optional Negation Raising** is possible when

(i) X c-commands Y at PF, but Y scopes over X.

(ii) Where X is a logic operator and Y is the logic operator '¬'

The reason why Moscati (2010) suggests this 'optionality' in Negation Raising (and therefore also the widening of the negative scope) is because just that: there can often be more than one interpretation of a sentence's negation – especially when comparing (translating) languages which, comparatively, have an inversed placing of the negation (such as English and Dutch).[62]

[60] Haegeman, L. (2000). p. 23.

[61] Moscati, V. (2010). Negaton Raising: Logical Form and Linguistic Variation. New Castle upon Tyne: Cambridge Scholars Publishing. page 43.

[62] See Moscati, V. (2010) pages 44-50 on the ambiguity on negation, through translation from German to English. Here he shows that a single German sentence can be translated in two different ways, placing the negative marker differently in both translations. However, the interpretation of the translations is virtually the same, although they do seem to formally negate other segments of the sentence.

3.5 Semantic Features of Negation

In his introduction, Zeijlstra (2013) states that there are, to an extent counter intuitively, languages where negation is not limited to phrase-initial position.[63] Instead, some languages actually 'ban' negation from the phrase-initial position. He refers back to Payne (1985) and Horn (1989) who have coined for a so-called 'middle field' of negation. For example, Dutch, allows for negation to occur in sentence-initial position if, and only if, the negation does not contain a single negation marker. For example, see (27):

(28) a. Niemand vindt Kees een leuke man.
No one finds Kees a nice man.
"No one thinks Kees is nice man."
b. Niet iedereen kan goed koken.
NEG everyone can good cook.
"Not everyone can cook well."
c. Nooit heb ik iets gestolen.
Never have I something stolen.
"Never have I stolen something."
d. *Hans niet loopt op het gras.
Hans NEG walks on the grass.
"Hans doesn't walk on the grass" (or "Hans isn't walking on the grass").

In these examples, the contrast lays in the negative markers being joined in spell-out with a polarity-item or by having an XP as its compliment as in (22c), whereas in (22d), the negative marker 'niet' is standing on its own. Here, the negative marker does not have a XP as a compliment and thus is a 'single marker'. This, as Zeijlstra (2013) points out, is ungrammatical in most V2-languages; specifically V-to-C languages (such as Dutch). Single negative markers cannot occur sentence-initial.

One such example where V-to-C languages do not allow for the negation to occur sentence-initial, is when these languages allow for True Negative Imperatives (TNIs). What this entails can be exemplified as follows, in (29):

[63] Zeijlstra, H. (2013) Not in the first place. *National Language Linguist Theory*, *31*, 865-900.

(29) a. Hij fietst niet
He cycles not
"He doesn't cycle".
b. Fiets!
Cycle!
"Cycle!"
c. Fiets niet!
Cycle not!
"Don't Cycle!"

In sentence (29a) it is shown that the negative marker in Dutch (*niet*) follows the finite verb (*fietst*). In other words, the main verb occurs phrase-initial. When looking at imperative sentences such as (29b-c), we notice that here too the negative marker follows the verb, also when the verb is in V1 position. What TNIs prescribe is that negation can follow a finite imperative verb as it follows verbs in an indicative sentence. Other languages, however, such as Spanish do not allow for TNIs; there the negative markers in imperative sentences cannot use the same form as their indicative counter parts.

Zeijlstra (2013) notices this stark difference between negative-markers in V-to-C languages being 'excluded' from sentence-initial position when there is no XP, whereas negative quantifiers, such as '*nooit*', can exist in sentence-initial position without having an XP compliment to its side. His analysis exists of comparing various previous research done on various types of movement concerning both negation and imperatives. One of the positions Zeijlstra takes is originally from Frege (1892) and Lee (1988), namely that negation can only operate on the sentence's propositional content. That is to say that when negation is added to a sentence (one of any (semantical) kind) – be it imperatives, questions, or assertions – the sentence does not become a negation-sentence, but rather a negated imperative, a negated question or a negated assertion. In context of his analysis, this means that the primary scope of a sentence will hardly ever be the negation itself. When this assertion is placed within semantics and syntax, this means that negation cannot c-command the main scope of the sentence. In his paper, Zeijlstra takes an example from Rivero and Terzi (1995) on Slavic languages (in this case, Polish):[64]

[64] From Zeijlstra, H. (2013), page 870.

(30) Nie pracuj

[$_{CP}$ [$_{NegP}$[$_{Neg}$ ^{o}Nie][$_{IP}$[$_{I}$ ^{o}pracuj$_{[IMP]i}$] [$_{VP}$t$_{i}$]]]]

NEG work.2SG.IMP

"Don't work!"

According to this example, the negative marker 'Nie' c-commands the imperative verb (and thus also the imperative operator which contains the sentence's scope), which according to Zeijlstra (taken from Han) cannot be the correct construction. Building further on Han's preposition on negation and scope, Zeijlstra (2013) explains that the negation's scope is to be avoided by the imperative operator's scope. Han's solution to (29) would be that the IMP operator feature continues on from the I^{o} position (dropping the verb), and continues on to the C^{o} position instead. This would leave the NEGP being c-command by the (phonologically silent) IMP in C^{o}, instead of having IMP being c-commanded by NEG^{o}.

To this explanation, Zeijlstra adds the idea that in some languages negative markers are semantically vacuous. This preposition would give room for examples as sentence (20) to occur without having to suggest the movement of a phonologically silent semantic feature IMP. Namely, the structure would remain very similar as it was proposed by Rivero and Terzi, for now the scope of the negative operator would not c-command the scope of the imperative operator from NEG^{o} due to the negative marker not having a semantic operator contain a negative feature. The imperative's scope remains untouched in (30).

This idea of negative operators being semantically vacuous comes from an earlier paper by Zeijlstra on negative concord from 2004. Here, Zeijlstra argues that there are two types of Negative Concord (NC) languages. That is to say, languages which allow (or require) double negation to occur in sentences. The distinction between the two types of NC-languages are so-called Strict NC-languages and Non-strict NC-languages. The difference between the two is that Strict NC-languages contain negative words (n-words) which require another negative marker, whereas Non-strict NC-languages also have n-words which require negative markers, but these markers can only accompany post-verbal n-words. What occurs, is that both these NC-languages need two elements which each carry different negative features in order to produce negation: one element which contains an interpretable negative feature ([iNEG]) and one element which contains an uninterpretable negative feature ([uNEG]). These two features

then check each other in order to create negation ([iNEG] checks [uNEG]). In some cases within these languages, the [iNEG] feature cannot be realized in a phonologically present element. So instead, Zeijlstra introduces the abstract element *OP*¬ which serves simply as a semantic node, in order to replicate the features of [iNEG] in order to check the [uNEG] features in the n-word.[65]

The distinction between the various negative feature carrying elements a described above is what Zeijlstra uses to explain the positioning of negative markers in both V-to-C languages (as Dutch), as well as the allowance or refusal of the before-mentioned NTIs across languages. To do so, he makes a difference between three classes of languages: Class I, Class II and Class III. The distinction of these three classes can be described as follows:[66]

(31) a. **Class I**: Languages which contain negative markers containing the [iNEG] feature (a semantically active negative marker). Negative markers also occupy NEG^{O}-positions
b. **Class II**: Languages which also contain semantically-void negative markers (which carry [uNEG] features).
c. **Class III**: Languages which only contain semantically-void negative markers.

Virtually all three of these Classes described by Zeijlstra follow the same syntactic rules to for a similar construction, but are limited by either semantic scopes, or by other syntactic-constraints belonging to that particular (type of) language. An example of a Class I language is Spanish, with its semantically active '*no*'. French, with its '*ne… pas*' is an example of a Class II language, where one negative marker is semantically active and the other is semantically void. Finally, examples of Class III languages are English and Dutch. Both languages contain mere semantically void negative markers.

In addition, Merchant (2001) describes, certain languages have a phrasal negative marker, whereas others do not.[67] The syntactic difference lies in languages of the first kind will have the negative marker as a phrase (as an XP), whereas the latter will have their negative marker

[65] This abstract element OP¬ seems to correspond with Moscati's mentioned before.
[66] From Zeijlstra, H. (2013), page 883.
[67] Merchant, J. (2006). Why no(t)? Style, 40 (1&2), pages 20-23.

more as a head (X^o).[68] At the start of his paper we learn that English, in this case, is of the first kind: phrasal. Following Zeijstra's Class system, it confirms the idea that English is, as Dutch, a Class III language. This implies that in case of negation, semantic features can be ignored which have to be checked within sentences as well as the issue regarding scope.

All in all, what Zeijlstra shows in his papers are two things: first, there are languages which handle the semantic features of negative markers differently. Second, the semantics of the negative features can influence the position of other words in the sentence, depending on their own semantic features. To be more exact, the element which holds the scope of the sentence, and the elements which holds the negative scope of the sentence are to be placed thus in sentences that the negative element is submissive to the elements which holds the sentence's primary scope. For the purpose of this paper's analysis on Machine Translation of Dutch into English, this means that no additional effort is required to convert negation from the one language into the other. However, it is worthwhile to note that, according to Zeijlstra's analysis on negation, more effort would be in order when translating negation across different classed languages – such as from French to Spanish. Albeit outside the scope of this paper, it could be argued that Class I and II languages could be easier to translate than Class III languages, due to a more explicit negative scope.

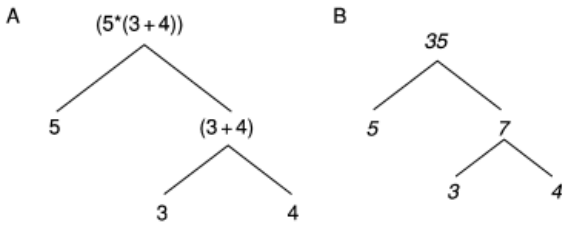
3.6 Merging Semantics with Syntax

Thus far, perspectives on negation in general have concerned with either semantics or syntax as separate approaches. In order to come to a final and successful conclusion on the matter of the use of negation in Statistical Machine Translation, it seems that a link between semantics and syntax has to be made. Luckily, many theories and approaches to this matter have been discussed by others, avoiding the need to re-invent the wheel. As Koenig (2006) states, the idea that syntax can be mapped into semantics in a relatively easy manner is often rather difficult, for the syntax of many natural languages seem to violate different hypotheses on one note or another.[69] A more accessible approach would be to first analyze how semantics and syntax can correlate to each other in artificial languages.

[68] Merchant notes here that there are other languages, such as Turkish, where negation is more word-internal and morphological of nature, but does not discuss these further.
[69] Koenig, J. P. (2006). Syntax-Semantics Interface. In *Encyclopedia of Language & Linguistics*. (Second Edition), 427–438. Retrieved from http://www.sciencedirect.com/science/article/pii/B0080448542019891.

In artificial languages, word orders are set, and thus it becomes a simple task to, for example, determine the scope of a particular semantic entity. Take the example as presented in (32), with the use of numerals instead of words or phrases:[70]

(32)



This example is of an artificial language which follows the following linguistic rules, with (33a-b) being syntactical rules, and (32c-d) being semantical rules:[71]

(33) a. If *a* and *b* are syntactically well-formed expressions, then so is ⌜ (*a* + *b*) ⌝.
b. If *a* and *b* are syntactically well-formed expressions, then so is ⌜ (*a* * *b*) ⌝.
c. A formula of the form (*a* + *b*) refers to the sum of the referent of *a* and the referent of *b*.
d. A formula of the form (*a* * *b*) refers to the product of the referent of *a* and the referent of *b*.

By combining both the syntactic (a-b) and semantic (c-d) rules in (33), we can come to the arithmetic answer as presented in (32). Because one-to-one mapping between semantics and syntax is relatively easily possible in this artificial language, we can speak of what is known as homomorphs.[72] As a result, we can say that in example (32), *5* takes *(3+4)* as its scope; for the joined phrase is its sister to its right. This has as an affect that the numbers are multiplied in order to create the number *35*. Therefore, within this artificial language, it is very clear what the (semantic) scope of a particular element is due to its syntactical position within the sentence.

As stated before, however, in natural languages, rules are not always as straight-forward as these are. For example, have a look at the following sentences expressed in (34):

(34) a. John began the journey.

[70] Example taken from Koenig (2006), figure 1, page 428.
[71] Also taken from the example by Koenig, J. P. (2006).
[72] See Koenig, J. P. (2006). A variation to homomorphs are 'isomorphs'. The difference being that homomorphs are literally one-to-one between semantics and syntax (that is to say, one syntactical rule can have one semantical input), and isomorphs are less one-to-one (i.e. multiple syntactical rules can have multiple semantical rules).

b. John emptied the glass.

In the example above, the two sentences contain verbs of the same morpho-syntactic type: past perfect tense. However, semantically speaking, we can assume that the event prescribed to (34a) has yet to come to an end (that John is on a journey), whereas the event in (34b) has come to a close. This is a simple, yet effective example as to how natural languages (in this case, English) are difficult to map homomorphs in.[73]

In his chapter, Koenig (2006) presents two general hypotheses which are widely-used by various linguists in order to address the issue of merging syntax with semantics in natural languages.[74] The first hypothesis is called 'Deep Split Structural Isomorphism' (DSSI). This hypothesis takes the stand that the reason why natural languages are too imperfect for one-to-one mapping of semantics and syntax is because of the (wrong) focus on the surface level of the sentence structure. In other words, sentences are too large in order to properly map out the homomorphs. Instead, a type of micro-management on the various individual strings is needed in order to properly map out the homomorphs. For at some syntactic level or semantic level, there is bound to be a level where micro-syntactic rules can be mapped out to correspond with micro-semantic rules.

According to the General Binding theory, the structure of a sentence can have different forms. Namely, a Logical Form (LF) and a Phonological Form (PF) which both derive from the same Deep Structure (DS).[75] Starting at DS, grammaticality of θ-roles are prescribed to the corresponding actors and patients. Later on, the sentences are split to the PF and the LF forms, whilst being assigned case and various operators are checked. At LF, the sentences gets it logical (semantical) meaning, and the PF it can be pronounced.

What the DSSI proposes is that the mapping between syntax and semantics happens on the different phases between the DS, LF and PF. Mainly, on the DS-level, that is to say, before spell-out, it is a lot clearer as to what syntactic element were to correspond to what semantic

[73] For other examples of how natural languages are difficult to map homomorphs in, see Koenig (2006).
[74] In truth, Koenig, J. P. (2006) offers three hypotheses. However, the third one, the Imperfections Reflect the Architecture of Grammars hypothesis, is too conceptual of nature to suit this paper's needs and is therefore discarded from the analysis presented here.
[75] See Hornstein, N. et al (2005), chapter 2. At the second half of that chapter also the issues the Minimalist Program has with this methodology are explained, as well as its alternatives.

element and what the consequences of such correspondence is. Below, three representations of the same sentence (“*Everyone loves someone”)* are shown. These are one S-level representation (as the sentence is at Spell Out) and two logical representations as to how this sentence could be structured at the LF-level.[76] The verb “*loves*” is capable of taking on two quantifier nouns (as it has two θ-roles, one for the actor, and one for the patient). However, it can be ambiguous as to which NP is given which θ-role (represented in the figures as ‘*Q*’)

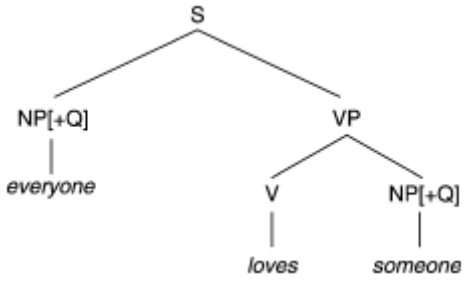


figure 10: S-level representation

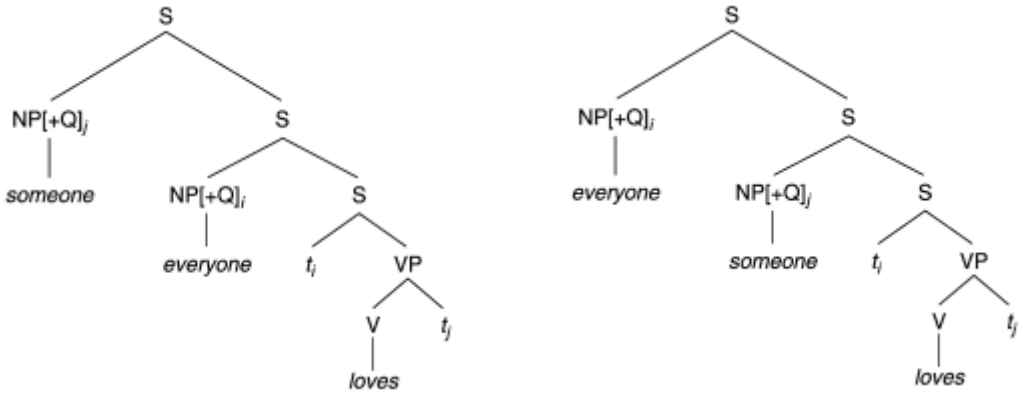


figure 11: two possible LF representations of the same sentence

Without knowing for certain which θ-role is given to which NP, the sentence will remain ambiguous. The only moment in the process one can be certain about its interpretation, according to the DSSI, is at D-level, as presented below, using the sentences “*Mary played/Mary will play*”.

[76] As taken from Koenig, J. P. (2006) pages 430-432.

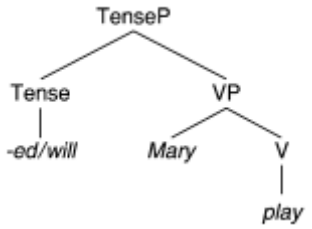


figure 12: D-level representation

At this level, the otherwise present ambiguity has left, due to nothing having moved yet, and every element (including tense) c-commands what it initially should. In this particular case, the Tense c-commands the entire event of *Mary* acting out the verb *play*.

However, there is also critique on this hypothesis; namely that in order for the hypothesis to work, it must be assumed that the sentences are indeed constructed precisely through the various steps of the different levels. As well as the assumption that at different levels different elements of the syntactic-semantic relations can be deducted.[77] Also, this hypothesis seems to be outdated, especially for those who follow the Minimalist Program.[78]

A second hypothesis comes from Categorial Grammar; the 'Natural Language Perfection Hypothesis'(NLPH). The first contrast with this approach compared to the DSSI hypothesis above is that it discards the idea of a rule-to-rule relationship between syntax and semantics.[79] Especially the implicit conclusion that therefore only constituents can contain semantic information. Unlike generative grammar, Categorial Grammar uses the Lambek Calculus instead of syntactical trees to express the structure of a sentence. According to Morrill et al. (1990), Categorial Grammar can best be described as being a directional system.[80] This directional system uses '/' and '\' in order to express how combinatory expressions are formed. The simplest example is that the sentence *' he walks*' is only a sentence because a

[77] Koenig, J. P. (2006), pages 431-432.
[78] See Hornstein, N. et al (2005), chapter 2.
[79] Steedman, M. (2000) The Syntactic Process. Cambridge, Mass.: MIT Press. Retrieved from http://quod.lib.umich.edu/cgi/t/text/pageviewer-idx?c=acls;cc=acls;rgn=full%20text;idno=heb08464.0001.001;didno=heb08464.0001.001;view=image;seq=27;node=heb08464.0001.001%3A4.1;page=root;size=100
[80] Morrill, G., Leslie, N., Hepple, M. & G. Barry. (1990) Categorial Deductions and Structural Operations. In G. Barry & G. Morrill (Eds) *Edinburgh Working Papers in Cognitive Science. Volume 5: Studies in Categorial Grammar.* 1-22.

noun (NP) precedes its verb 'walks'. Hence, a sentence (S) follows due to a preceding NP. See below in (35) how this is expressed:[81]

(35) i. he := NP, 'he walks' := S
ii. NP\S := walks
iii. 'he walks' := (NP\S)/S

This approach to grammar using the Lamek Calculus takes a more algebraic approach to grammar. This form of grammar, much as Generative Grammar theories, categorizes words in sentences. As presented in (35), the noun is categorized as being a noun phrase, NP, as it would be in some Generative Grammar theories. However, one aspect where Categorial Grammar differs from the other approach, it does not define verbs (or verb phrases) as categorized types of their own. Instead, verbs are defined as how they are related to NPs. As in (35), the verb '*walks*' is defined as the type 'NP\S'. It can seem unintuitive to approach grammar in such a way where verbs are not defined as verbs on their own, or where categories of the one type are defined in expressions of other categories. From relatively simple algebraic point of view, however, this is not the case. For when $\frac{X}{Y} = Z$, then all instances of *Z* can be expressed as $\frac{X}{Y}$. As Morrill et al. (1990) explain:

(36) i. If *X* is a primitive type, then *X* is a type.
ii. If *X* and *Y* are types, then *X/Y* and *Y\X* are types.

Another way of interpreting this approach is by reading it as following: [input] ['to the left' or 'to the right' symbol] [output].[82]

Similar to other algebraic rules of elimination, we can, through the use of this Lambek Calculus, 'deduce' a grammatical string of words:[83]

(37) i. $\frac{4 \cdot 3}{2 \cdot 3} \rightarrow \frac{4 \cdot \cancel{3}}{2 \cdot \cancel{3}} \rightarrow \frac{4}{2} = 2$
ii. Mary likes John → Mary * likes * John → NP * ((NP\S)/NP * NP)

[81] As taken from the example from Morrill, G. et al (1990). p. 2.
[82] Moot, R. & C. Retoré. (2012). The Logic of Categorial Grammars. Berlin Heidelberg: Springer.
[83] Example ii. Taken from Morrill, G. et al. (1990). p.4.

iii. (likes John) := ((NP\S)/~~NP~~ * ~~NP~~) → NP\S

iv. Mary * (likes John) := ~~NP~~ * ~~NP~~\S → S

v. Mary likes John := S

In (37), we can see that the simple calculation in (i) can be applied to Categorial Grammar for the use of elimination.[84] This process can be used to result in another important aspect in deducing the sentence – namely the assumptions the sentence makes. According to Morrill, G. et al. (1990) this can be achieved by looking at the direction of the elimination at various stages. A cluster can be made from the string of words which have the same direction of elimination, which can be used to deduct its semantics through the use of semantics. Namely, in the example presented in (37), it can be concluded through the means of the direction through which the elimination occurs that "*likes John*" is a cluster, because the elimination there took place to the right. "*Mary"* is a separate cluster, because the elimination there occurs to the left.

According to Hendriks' dissertation (1995), Categorial Grammar also allows to express concatenations of different phrasal expressions.[85] Namely, when Z is the product of the expression of an X and an expression of a Y, in that order, this can be represented as being (X·Y). Do note that (X·Y) is not the same as (Y·X). An example of such a feature given by Hendriks (1995) is the Dutch "*leggen*", "to put something somewhere": this verb can be represented as the functor PP\(NP\VP) or as the functor (NP·PP)\VP.[86] The difference between using the · instead of either / or \ depends on the structure of the entire phrase; the first noting a flat structure, whereas the latter noting a binary split.

With respect to the mapping of syntax and semantics, this means that each separate constituent is also represented in its logical form, whilst being placed in a Lambek Calculus formula from which the sentence's complete semantical meaning can be deduced. Hendriks (1995) gives the following:[87]

[84] Do note that in Lambek Calculus refers to the "/" and "\" symbols as "to its right" and "to its left" respectfully, and not "above" or "under". The "·"-symbols used in (24) are purely as a visual support, and do not fulfill any other function than that.

[85] Hendriks, P. (1995) *Comparatives and Categorial Grammar* (doctoral dissertation Rijksuniversiteit Groningen) Groningen Disstertations in Linguistics 14 (80). print.

[86] Hendriks, P. (1995), page 83. Do note that VP is usually written as NP\S.

[87] Hendriks, P. (1995), page 84.

(38) i. X/Y:*f*, Y:a → X:*f*(a)
ii. Y:a, Y\X:*f* → X:*f*(a)
iii. (Z\X)/Y:*f* → Z\(X/Y): $\lambda v_1 \cdot \lambda v_2 \cdot f(v_2)(v_2)$
iv. (Z\(X/Y):*f* → Z\X)/Y: $\lambda v_1 \cdot \lambda v_2 \cdot f(v_2)(v_2)$

To a degree, this is already exemplified in (37), namely (i) and (ii) in (37). Following (i) first, X/Y correlates to *likes* ((NP\S)/NP) and Y correlates with *John* (NP). This results in NP\S: *likes*(*John*). Next, there is (35(ii)) in which X correlates with *likes*(*John*) (NP\S), and *f* with *Mary*. Which will result in S:*likes*(*John*)(*Mary*). The formulas described in (iii) and (iv) in (38) follow the same principle, but for more complex functors; namely functional elements which require more than one argument (for example, ditransitive verbs).

Although different to what Koenig (2006) proposes for homomorphing, the use of Categorial Grammar provides a methodology through which semantic information from formed sentences can be extracted in a deductive manner. By using Categorial Grammar along with the NLPH as mentioned by Koenig (2006), a large advantage towards this approach compared to the DSSI hypothesis, is when following NLPH, a Deep Structure (re)analysis of a sentence is not needed. For the scope of this particular paper, this is a huge advantage. For, as previously shown, Statistical Machine Translators only work on a Surface-level of a sentence. A SMT cannot analyze its source language's Deep Structure first, to then analyze the Deep Structure of the target language second, and after its complete analysis of these two, easily create a complete flawless translation.

If we were to incorporate the theories on negation that we have discussed thus far – how could we best incorporate with this Categorial Grammar method of homomorphing needed to deduct the meaning of negation within a sentence?

3.7 Combining Theories on Negation

The theories regarding negation discussed above give explanations as to where negation is placed within a sentence's structure and how the scope of a sentence is defined. However, these theories rely on an analysis of sentences from a Deep Structure-perspective, and as Koenig (2006) states, mapping syntax and semantics cross-linguistically is an exercise which is bound to end in failure. To analyze sentences from a Surface's Level, Categorial Grammar

appears to be the more beneficial method. For, as long as it can be deducted through what element within a sentence reflex on another element, translations of such elements can also be related to one and other in the same manner. However, as Wansing (2007) examines, the Lambek calculus approach that Categorial Grammar uses does not have a clear method incorporating negation.[88] Whereas the Lambek calculus could be used to understand negation in the sentence by approaching it purely from a logical point-of-view, this does not suffice the purpose of this paper's exercise: translating negation from Dutch to English. So a different approach must be found.

Zeijlstra's preposition on the scope of negation can be expressed through the use of Categorial Grammar. As expressed in (37), a sentence's semantic scope can be deduced through the use of Categorial Grammar. This can then also be true for a sentence's negative scope. Even though both English and Dutch are Class III languages, and thus have semantically-void negative markers, using Categorial Grammar to determine a sentence's negative scope in Class I or Class II languages will also work; as long as negative functors can be defined properly.[89] Haegeman's and Moscati's theories on negation are, however, more difficult to properly incorporate in Categorial Grammar due to their presumptions on negation rely on implicit analyses of the sentence and how sentences are constructed in various steps before Spell Out; such as movement. However, what Haegeman and Moscati do make clear, is that a negative element (or a NEG-criterion) does not exist on its own. That is to say that negation, in a generative grammar approach, appears in the [Spec, Head] position of a so-called NEG-P. How this can be added to the Categorial Grammar, is that a negative functor or category will have to be added to address this feature.

Another issue in which the use of Categorial Grammar seems to fail which is used in Haegeman's approach (as well as in Zeijlstra's) is the principle of movement. Especially when examining a language such as Dutch, in which scrambling is a common feature, movement is an exercise which cannot be ignored.[90] As movement is a DS-exercise, it is impossible to address or analyze this from a pure Spell-Out analysis, which Categorial

[88] Wansing, H. (2007) A Note on Negation in Categorial Grammar. *The Author, 15(3),* 271-286.
Instead, Wansing mentions three different propositions on negation in Lambek Calculus by Buszkowksi. Wansing mostly elaborates on Buzskowsoki's notion of his so-called 'LN'-rule using connexive logic. Wansing however focusses on incorporating and interpreting negation in the Lambek calculus, rather than using it to understand grammatical structures in sentences.

[89] In other words, if there are more than one negative markers in a sentence, eventually the one marker should be able to check the other marker as a functor to its right or its left.

[90] Haegeman, L. (1995), pages 58-69

Grammar essentially is. For example, most sentences presented in (2) are sentences in which some form of movement has to have occurred in order for the negation to be at the end and appear as if it is outside the initial VP. This follows suit to what Moscati states, that the scope of negation at the Spell-Out level can be distorted, due to movement which has occurred during the sentence's creation.

An approach to solving the problem of movement of categories within a sentence's creation is addressed in Cremers (2004) for the Categorial Grammar machine '*Delilah*'.[91] This approach, aimed at a computer program which can deduct semantic interpretations from Dutch sentences through the methodology of Categorial Grammar, takes in account that a lexical item can be represented in a multitude of functors. In other words, a lexical item can be represented in more ways than just one functor. The program *Delilah* deducts the semantic meaning of a sentence by also checking which particular functor-variant of the lexical elements are relevant in the particular given sentence. More precise, *Delilah* is programmed thus that it takes into account possible empty and non-empty positions to a lexical item's left or right. The challenge there lies in correctly representing a lexical item in the correct functor. Thus, when employing this logic to the issue with the movement of negation in Dutch, a negative functor can be developed in order to cope with various positions of the negative operator.

[91] Cremers, C. (2004), Modal Merge and Minimal Move for Dislocation and Verb Clustering. *Research on Language and Computation*, *2(1)*, 87-103

Chapter 4: Critical Analysis and Comparison

4.1 Problems with Machine Translation

As mentioned in chapter 2, Statistical Machine Translation models are efficient, but ineffective. Rather than properly looking into what the sentences are trying to convey, or how these are actually constructed, most Statistical Machine Translation models rely on comparisons rather than actual translation. In their paper, Costa-Jussa & Farrus (2014) list the various areas where, according to them, Machine Translation models are still lacking and where the most progress lies. Syntax and semantics are the two areas which lack the most progress. Instead, it seems as if the focus of the most efforts in Machine Translation lay in achieving the highest BLEU-scores instead of achieving proper translations.

The argument that Farzi et al (2015) make is that there is a lack of coping with long-distance syntax in Machine Translations. The examples used in the previous chapter illustrate this problem as well. The negated "*niet*" at the end of some of these sentences fail to be ordered properly in the translated output sentences. As discussed in 3.1 and 3.2, Dutch syntax is freer than English, which allows Dutch to place negation elsewhere in the sentence. Although Farzi et al (2015) probably do not refer to long-distance syntax in this particular way it still does fit the argument. The fault here, of course, lies in the dependency of working with *n*-grams. A particular string of sequential words have to be selected in order for the Statistical Machine Translators to correlate them to other strings. From the point of view of statistics, if a larger string of words, and thus a larger *n*-gram, is able to produce a probable correlation with a high value, it is logical to assume that this is a good translation. However, when using an n-gram approach for translating a string with long-distance syntax, the Translation Machines will have to use a larger *n*-gram. This will result in lower probable translations to be used, due to the lower chance of correlating large strings with other long strings. The beam-search-decoder approach might seem as a possible solution for this issue. Using small strings to incorporate these into larger strings, to compare again, until a final translation has been produced can possibly solve long-distance syntax issues. An example of this can be seen in figure 9 of the previous chapter. However, as other examples in the previous chapter also show, this approach is not perfect and still allows for wrong translations to be produced.

Another flaw with the previously mentioned Machine Translation methods, is not only that they are statistically based, but they are confined to existing corpora, which are always limiting, no matter how large they are. Hearne & Way (2011), for example, state that using corpora for translations is always limiting, for there could always be a word which is queried for translation which simply does not exist in the used corpora. This implies that Translation Machines would be ill-equipped to successfully translate sentences which are grammatically correct, but have never been spoken (or translated) before. As stated in the previous chapter, the decoding-phase of the Machine Translation remains to be difficult to produce proper sentences with. For it is possible, due to the context, that a more obscure and rare translation is required to be produced. Due to the use of probability scores, these obscure translations can sometimes be left out. As Hearne & Way (2011) briefly mention, in some cases, a particular word-pair can be difficult to successfully produce due to the brevity of lexical meanings a particular word in either the source or the target language.[92] The example they refer to, is when translating "*the large cat*" from English to French. This could, amongst other options, result in the translations "*la grosse chat*" or "*la grande chat*".

Perhaps the largest issue in Machine Translations, statistical or otherwise, is that these translation programs are surface sentence bound. In other words, these translation machines can only translate sentences by analyzing and interpreting the sentences as they are. Which means that linguistic syntax theories which use mechanisms such as *move*, *merge* or *checking*, as well as some other mechanisms which require deep structure analyses, cannot be used as they are when using translation machines. Without explicitly programming these machines to correlate particular linguistic patterns within sentences to one and other, as well as to explicitly program these machines to know what properties a particular word has (such as, for example, whether or not a word is a noun or a verb), a completely error-free translation machine is impossible to realize. This does not mean, however, that adding some aspects from various linguistic theories to Statistical Machine Translators is an impossible task, or that such additions could not improve the results these machines produce.

4.2 Improvements for Statistical Machine Translators

Below in (39), the main issues regarding Statistical Machine Translation which are mentioned above are listed in no particular order:

[92] Hearne & Way (2011). p.212.

(39) (i) Limited selected corpora
(ii) Using n-grams is limited when confronted with long, complex strings
(iii) Machine Translators are focused on the surface structure of sentences

When attempting to improve the translations of negation in sentences from Dutch to English when using Statistical Machine Translators, several approaches are possible to come to different solutions solving the problem. Solving or improving any of the three issues described above in (39) could result in Statistical Machine Translators properly translating negation in Dutch sentences to English sentences, or increase the success-rate of their translations.

Starting with issue (i), enlarging the corpora used for both the source language as the target language should increase the probabilities of correlating the right set of strings from the two sets of corpora. The general idea behind this solution is that the Machine Translators are able to properly translate even the longer strings of words, as long as these precise strings occur within the corpora used – even more so when these strings occur frequently in both corpora, increasing the Bayes' P(A|B) value. However, this solution is not a sustainable or a perfect one. As long as it is possible to create new sentences which have never been produced before, this method will remain flawed. Of course, an immensely large corpus could house the elements needed for new, unproduced sentences which could still be statistically be correlated to elements found in a different corpus. This approach would solve current issues with Dutch-to-English negation, but it remains to be flawed.

Solutions for the issues presented in (ii) and (iii) above could, however, provide with more sustainable mechanisms for translating negation as well as other faulty translations Machine Translators produce caused by these issues. These two issues are also related to one and other. Namely that the n-gram method is a method used to analyze the sentences on a surface level. Whilst analyzing sentence purely on a surface level, without any interpretation of the words used, it becomes difficult for these Translation Machines to place, or re-order these words properly after translating the words separately or in a small n-gram string. Of course, interpreting the sentences from a deeper layer such as most linguistic syntax and semantic theories do, can also contribute in interpreting the target language as well, before the translation itself. This could influence the probability of the selection for pairing the strings between the source and the target language.

The linguistic theories on the position, creation and scope of negation as discussed in earlier chapters by Haegeman, Zeijlstra and Moscati do give explanations as to how the semantic meaning or function of a negative element in a sentence can be analyzed. However, the issue here with regard to possible implementation in SMTs, is that these theoretical methodologies analyze sentences further than their mere surface level (or Spell Out). This is, for the purpose of this analysis, the largest pitfall of generative syntacticians: the explanation and interpretation of why sentences are formed as they are and what they are therefore to mean, rely on an assumption of how these sentences are supposedly to have been formed – taking a hypothetical construction as starting point to reach the final state of the sentence. This is neither a criticism on generative syntax or the more modern Minimalist Program – for there are plenty of other articles in different research areas which benefit from such theories. However, a practical solution for translating sentences, especially mechanical translating, must be found elsewhere. Solutions to a mechanized translation process using theories from a generative syntax framework would require a system which first deconstructs the input sentence, only to interpret and reconstruct it into the target language. Not to forget that such a system would also need to have an interpretation of sentences' context.

Such a system would require vast annotated corpora to construct suitable lexicons which would include all the possible constituents which can either accept or distribute θ-roles or semantic features which can (and should) be checked. Followed by an ordering system on how the various movement mechanisms are to be put into place and when whilst reconstructing the sentence to the target language. This is not to say that creating such a system is an impossible operation to fulfill – it is, however, are rather cumbersome one. Also, the methodologies used within Generative Syntax frameworks such as those used in the previous chapters are aimed to deconstruct produced sentences in order to comprehend how these sentences came to be – rather than constructing new sentences from scratch. Finally, throughout this paper, only a selection of theories and approaches from the Generative Syntax framework have been presented, only focusing on the one aspect of negation. There is, however, a vast amount of theories and rules produced on other aspects of sentence construction which each have their own strengths and weaknesses; with the biggest weakness often being loop-holes or exceptions to the rules.[93] This does not mean, of course, that the

[93] For example, see Boeckx' (2012) critique on Chomsky's Phase Impenetrability Condition (PIC). In short, Chomsky's PIC provides an answer to how extraction of, for example, NPs from Phrasal Islands is possible by having each section be constructed in 'phases'. By the end of each phase, all but the head of the created phase is 'set' and cannot be changed any longer. Only the phrase' head can still move to a higher position in the next

Generative Syntax framework does not hold any value, but rather that a different framework is preferred for the applicability of SMTs.

As mentioned before, according to Koenig, a mechanical application of the Generative Syntax framework could be ineffective at times, due to the fluidity of natural languages. Instead, as Koenig suggests, an approach which focuses on mere surface level (or Spell Out) analyses of sentences such can be of more use. Categorial Grammar is a framework which analyzes sentences just so. Although, this approach also has its limits. For the phenomenon of movement, which can distort the initial (and actual) interpretation of the scope and the function of negation in a sentence cannot be analyzed through the use of lambekian calculus. At least, not yet. Although the approach Cremers has with incorporating various possible functors into a single functor which can cope with various possible other functors to both sides of a lexical element, the issue with Statistical Translation Machines will not be solved. For the Categorial Grammar program *Delilah* uses its own lexicon where the lexical items are defined. Categorial Grammar can only properly work when all the lexical items are defined into functors.

Even though a Categorial Grammar-based approach is plausible when solving the issues (ii) and (iii) listed in (39), it will force Statistical Translation Machines to use some sort of functor-based lexicon. Furthermore, it will have to develop $n + 1$ lexicons; "n" being the number of possible target languages one would wish to translate a source sentence into. Even though this could result in semantically accurate translations, it will be a more costly and complex operation to do so successfully, than the current method through statistics is. Perhaps a more feasible solution lies in adding a type of annotated corpus to corpora already being used by the Translation Machines, and, for this particular case regarding negation, is to only configure a functor specifically for negation, whilst using an annotated corpus which can help categorize other elements of the sentence in order to be able to interpret the scope of the negation before translation.

4.3 Conclusion

---

phase, until all phases have passed. Boeckx' critique here is that this movement-system via the head position allows for constituents to always move, as long as they can be moved to the head position during each phase. In other words, rather than expressing limitations in movement, Chomsky's PIC allows for movement to virtually always be possible. From Boeckx (2012). Syntactic Islands. New York: Cambridge University Press. pp. 61-62.

It is clear that current translation apps which are Statistical Translation Machines do not use any, or hardly any, theories on syntax or semantics covered in various linguistic frameworks. This exclusion on sentence construction or sentence interpretation theories seem to be responsible for producing poor translations of especially sentences. In this paper, the focus was on the translation of negation from Dutch to English, as evidence proved that erroneous sentences can be produced – specifically when having negation in sentence-final position in Dutch. Various linguistic theories provide arguments that there are both syntactic and semantic elements to negation which cannot be ignored. However, these theories do not provide room for mechanized applications in translation. Unlike theories from the Generative Syntax framework, the Categorial Grammar framework approaches sentence-analysis in such a way that it does allow for a more mechanized application – as has been practically proven already in Delilah. How this is to be done exactly is, however, uncertain. For unlike Categorial Grammar, generative synacticians have analyzed and formalized negation, whereas categorial grammarians have not. If the area of Statistical Machine Translation wants to improve their output in order to produce translations which are also linguistically sound, they are best to incorporate aspects from Categorial Grammar to minimize awkward translations such as "*I know the man you saw yesterday not"*.

References

Bernardini, S. (2006). Machine Readable Corpora. In *Encyclopedia of Language & Linguistics.* (Second Edition), 358-375. Retrieved from http://www.sciencedirect.com/science/article/pii/B0080448542004764.

Carter, S & Christof Monz. (2011). Syntactic discriminative language model rerankers for statistical machine translation. *Mach Translat, 25*, 317-339.

Cremers, C. (2004), Modal Merge and Minimal Move for Dislocation and Verb Clustering. Research on Language and Computation, 2(1), 87-103

Haegeman, L. (2000). Negative Proposing, Negative Inversion and the Split CP. In L. Horn & Y. Kato (Eds.), *Negation and Polarity. Syntactic and Semantic Perspectives.* (21-61). New York: Oxford University Press.

Haegeman, L. (1995). *The syntax of Negation*. Cambridge: Cambridge University Press.

Hendriks, P. (1995) Comparatives and Categorial Grammar (doctoral dissertation Rijksuniversiteit Groningen) Groningen Disstertations in Linguistics 14 (80). print.

Hornstein, N. et al (2005) Understanding Minimalism. Cambridge: Cambridge University Press.

Hearne, M. &Way, A. (2011) Statistical Machine Translation: A Guide for Linguists and Translators. *Language and Linguistics Compass, 5*, 205-226.

Jacobson, P. (2002) The (dis)organization of the grammar: 25 years. *Linguistics and Philosophy, 25*, 601-626.

Koehn, P. (2010). *Statistical Machine Translation.* Cambridge: Cambridge University Press.

Koenig, J. P. (2006). Syntax-Semantics Interface. In *Encyclopedia of Language & Linguistics.* (Second Edition), 427–438. Retrieved from http://www.sciencedirect.com/science/article/pii/B0080448542019891.

Merchant, J. (2006). Why no(t)? *Style, 40 (1&2)*, 20-23.

Moot, R. & C. Retoré. (2012). *The Logic of Categorial Grammars.* Berlin Heidelberg: Springer.

Morante, R. and C. Sporleder. (2012) Modality and Negation: An Introduction to the Special Issue. *Computational Linguistics, 38(2),* 229-231.

Morrill, G. (2006). Categorial Grammars: Deductive Approaches. In *Encyclopedia of Language and Linguistics*. (Second edition). 242-248. Retrieved from http://www.sciencedirect.com/science/article/pii/B0080448542008919.

Morrill, G., Leslie, N., Hepple, M. & G. Barry. (1990) Categorial Deductions and Structural Operations. In G. Barry & G. Morrill (Eds) *Edinburgh Working Papers in Cognitive Science. Volume 5: Studies in Categorial Grammar.* 1-22.

Moscati, V. (2010). *Negaton Raising: Logical Form and Linguistic Variation*. New Castle upon Tyne: Cambridge Scholars Publishing.

Mukesh, G.S. & Vatsa, N. J & Goswami, S. (July 2010). Statistical Machine Translation. *DESIDOC Journal of Library & Information Technology, 30 (4),* 25-32.

Steedman, M. (2000) *The Syntactic Process.* Cambridge, Mass.: MIT Press.

Van der Auwerwa, J. (2001) Linguistics of Negation. In *International Encyclopedia of the Social & Behavioral Sciences*. 10462-10467. Retrieved from http://www.sciencedirect.com/science/article/pii/B008043076702965X.

Wansing, H. (2007) A Note on Negation in Categorial Grammar. *The Author, 15(3),* 271-286.

Zeijlstra, H. (2013) Not in the first place. *National Language Linguist Theory, 31,* 865-900.

Zwart, J.W. (2011) The Syntax of Dutch. 1st ed. Cambridge: Cambridge University Press, 243-280. Cambridge Books Online. Retrieved from http://dx.doi.org/10.1017/CBO9780511977763.011

Zwart, J.W. (2011). The Syntax of Dutch. 1st ed. Cambridge: Cambridge University Press, 281-295. Cambridge Books Online. Retrieved from http://dx.doi.org/10.1017/CBO9780511977763.012

Zwarts, F. (1986) *Categoriale Grammatica en Algebraische Semantiek: Een onderzoek naar negatie en polariteit in het Nederlands*. (doctoral dissertation, Rijksuniversiteit Groningen), Groningen Dissertations in Linguistics 10 (n/a). print.