University of Zurich^UZH

Dynamic and Distributed Information Systems

Real-time crowd-based subtitle translation

Bachelor Thesis Feb 3, 2013

Nico Rutishauser
of Bülach, Switzerland

Student-ID: 09-706-821
nico.rutishauser@uzh.ch

Advisor: **Patrick Minder**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

For the support of this paper, I want to thank several people. First, thanks to Patrick Minder for the help, the great ideas and all other efforts. He stayed on top of things and guided me to the right direction. Furthermore, I want to thank Professor Abraham Bernstein for the possibility of writing this thesis at his department at the University of Zürich. Special appreciation also to Sandra Pfiffner, Ken Kingsley and Till Salinger for reviewing and other inputs.

Zusammenfassung

Diese Arbeit führt eine Methode zur Echtzeitübersetzung von Sätzen ein. Das wachsende World Wide Web bietet jedem Nutzer die Möglichkeit, aktiv am Internet teilzunehmen und damit zu arbeiten. Millionen von Benutzer können als *Crowd* sogenannte Mikro-Aufgaben erledigen. Die Echtzeitübersetzung besteht aus der Verbindung zwischen einer maschinellen Übersetzung für eine Rohfassung und einer einsprachigen Crowd für Verbesserungen. Es sind sechs Kombinationsverfahren vorgestellt, die die besten Teile der Resultate der Crowd-Arbeiter zusammenfügen. Die neuen Verfahren wurden mit 100 Testsätzen in einer Echtzeitsimulation getestet, jedoch schaffte es keines der Verfahren, die Qualität der Rohfassung deutlich zu übertreffen. Mit dem *Rapid-Refinement*-Prinzip kann schon nach 30s bis 40s eine Einigung zwischen den Crowd-Arbeitern gefunden werden. Es wird gezeigt, dass mit einer optimalen Kombination der Sätze die METEOR-Punktzahl von 43% auf 49% gehoben werden kann. Mit einem guten Umgang mit der Crowd könnte das Potenzial weiter gesteigert werden. Die vorgestellten Kombinationsmethoden können für weitere Anwendungen von Textverarbeitungen in natürlicher Sprache eingesetzt werden.

Abstract

This thesis introduces a method to translate streamed sentences in near real-time. The ability for participation and interaction arises with the growing World Wide Web. Millions of users form a crowd and can process so-called micro tasks. A combination of a machine translation system for the pre-translation and a monolingual synchronous crowd for improvements is used. Six sentence alignment methods are introduced that search a good combination of the crowd-workers results. The merging algorithms are tested with 100 sentences in a simulation of a real-time crowd. With any of the combining methods, the output does not improve compared to the pre-translation. With *rapid refinement*, an agreement on the combined output can be found after 30 to 40 seconds of working time. It is shown that an *optimal alignment algorithm* can improve the METEOR score of the pre-translation from 43% to 49%. However, a good handling of the crowd could raise the potential even more. The proposed merging methods can be used for future natural language processing, for example in multi-engine machine translation.

Table of Contents

1

Introduction

Over the years, computers have become an important part of our society. Our daily lives are influenced dramatically not only by personal computers in our homes, but also by machines we never see. Hardware power is increasing. Demanding problems of yesterday are solved faster, but new problems arise which need even more computing power. As computers do only what the programmers command, the need for humans in the equation becomes even more critical. There are challenges that cannot be solved by increasing the power of a machine. One field in which this is true is handwriting recognition.

There is a rapid progress of communication and collaboration techniques over the Internet. Computers can communicate among each other, and human beings can collaborate on a large scale. As the World Wide Web expands, information becomes more connected, and opportunities for human interaction increase dramatically. In a distributed manner, it is possible to work on tasks over the Internet with uncountable other users. This Internet community, or what we refer to as a *crowd* [Malone et al., 2010], is growing fast. The crowd provides collaboration and teamwork over the network. Not only hundreds, but millions of users can participate. Wikipedia[1] is a good example of collaboration by a crowd. Many individuals working together as a crowd can produce content in large quantities and of good quality. Each user contributes only a small part, but the total is a large encyclopedia. A single company could have never done this work alone [Malone et al., 2010]. Many other companies incorporate the crowd deeply into their business model or daily work, as seen for example at Google[2]. This decentralized resource is also known by the terms *human computation* or *social computing* [Von Ahn and Dabbish, 2008].

Natural language processing cannot be completely replaced by powerful machines. Computers can learn rules, create word nets, store databases and statistics about words, but still not understand the natural language. In language processing where understanding the communication is required, humans are irreplaceable.

---

[1] http://www.wikipedia.org

[2] http://www.google.com

1.1 Motivation

Subtitles, invented for integrating the deaf into the world of television have become important for other audiences as well. They are used, for example, in localities that do not allow audio or places where the ambient noise is too loud, like on TV at the airport as a pastime. Another aspect of transcripts is the capability to store the content easier and more efficiently. The storage of motion pictures needs a lot of disk space. To make videos detectable in archives, one needs sophisticated techniques. Usually, people producing or watching the video tag the content with keywords. With transcripts, videos can be queried with proven search engines for full-text search.

Having written texts, language could act as a barrier. With advancing globalization, we need techniques that help us to comprehend multiple languages. In case of digital or satellite television reception, households receive many foreign-language TV programs that are comprehended by only a fraction of the potential audience.

Much research about generating transcripts of spoken language in real-time has been done [Lasecki et al., 2012]. Speech-to-text applications are used for creating subtitles or protocols in lectures, parliament and many other fields. With the assumption that these subtitles are in place, we present a technique to translate them using the crowd. Subtitles belong with videos and videos are often broadcast live. The goal of this work is to translate as fast as possible using a real-time crowd and a *rapid refinement merge algorithm*. A real-time crowd consists of workers available on call. It is also known as *synchronous crowd*. [Bernstein et al., 2011] showed that the crowd can be recruited within two seconds. *Rapid refinement* is a design pattern for a synchronous crowd that detects early consensus of the crowd [Bernstein et al., 2011].

1.2 Problems

The traditional approach for text translation is to contact a professional translator or a company offering such a service. The quality of professional translations is excellent, but there are some drawbacks. Professional translators are rare and therefore costly. Generating a good translation requires multiple iterations of translating and auditing, sometimes done by several people. This leads to the first problem because we aim at translations in real-time:

Problem 1. *Translations made by skilled translators can be expensive and these translators may not always be available. Depending on the content of the text, there are specialists needed who have the correct jargon. The translation process is done in several iterations and in the best case with a delay of multiple minutes.*

Making life difficult is not what we want, so why not just use machine translation (MT) from a free web service? This would solve the above problem. MT systems are fast and always available. Statistical MT services have witnessed a great deal of progress with sophisticated learning algorithms in the passed few years. They perform well on uncomplicated sentences, but are oblivious to the context because they do not understand the content.

German (original)	”Das Haus der Schnecke ist zerbrochen. Das Schneckenhaus muss repariert werden.”
Statistical machine translation to English (Google Translate)	”The house of the screw is broken. The shell needs to be repaired.”
Professional translation to English	”The snail’s shell is broken. The shell needs to be repaired.”

Table 1.1: Example sentence translated with Google translate

Table 1.1 shows an example for a poorly machine-translated sentence using Google Translate[3]. Although both sentences are about a shell, the statistical machine translation literally screwed up the first sentence. The MT engine detected the word ’Haus’, which is translated to ’house’ in most cases. Similar, ’Schnecke’ is translated with the technical term ’screw’ instead of the biological counterpart ’snail’. In contrast, humans know that for ’Haus der Schnecke’, it is better to use ’shell’ because they understand the content. The second sentence with MT matches with the professional translation and is suitable. This brings us to the second problem:

Problem 2. *Using only statistical machine translation results in bad translations.*

[Minder and Bernstein, 2012] recruited an asynchronous crowd for text translation. It can be assumed that a crowd offers a high availability, thus the translation is faster than with professional translators. The process is also done in multiple steps causing a high latency until completion. The crowd offers good parallelization and multiple parts of a text can be partitioned and translated separately. Similar to MT, the use of a crowd is scalable, but an asynchronous crowd is still slower than MT.

Problem 3. *With asynchronous crowd-sourcing, the translation is not fast enough for real-time applications.*

---

[3] http://translate.google.com

1.3 Hypotheses

With a new approach, we try to solve all three problems above. The combination of translation and human computation poses a new research area. This thesis tries to resolve three hypotheses. First, we ask ourselves, how the crowd can be used to improve erroneous sentences. Does a crowd perform better if it has to modify only wrong parts than if it has to retype the whole text? This leads to the first hypothesis:

Hypothesis 1. *The quality of an erroneous translated sentence, when corrected by a crowd, is better if the crowd-workers have to retype the whole phrase than if they only have to improve the wrong parts.*

Having a crowd that generates data, there has to be a *decision phase* to have a final result [Malone et al., 2010]. The decision is limited not only to selecting the best and pruning poor sentences, but also on merging them. This thesis tries to find an alignment method for the crowd-corrected sentences to improve bad translations. The generated output merges the best parts of the proposed improvements.

Hypothesis 2. *Multiple versions of a crowd-corrected sentence can be aligned so that a better phrase than its original emerges. An early agreement among the workers can be found with a rapid refinement algorithm.*

With a professionally translated reference, different alignment methods can be evaluated. The offline analysis shows the potential of crowd-based translations. There is an *optimal alignment algorithm* that finds the best combination of multiple versions of a sentence.

Hypothesis 3. *Combining multiple versions of a sentence leads to a higher quality than the quality of the best individual-produced sentence.*

1.4 Contribution

In this thesis, we present a concept to translate subtitles in real-time using a synchronous crowd of humans to improve a machine-translated draft and a rapid refinement algorithm that recognizes agreement. Subtitles of movies, TV shows, lectures or any other kind of recording arrive sentence-by-sentence. The process starts with a statistical machine translation system. As the machine translation does contain errors of different kinds, the crowd is used to revise it. After the pre-translation phase, each sentence is given to multiple crowd-workers. A rapid refinement algorithm merges the current state of all workers regularly, until satisfaction. For that, the crowd-corrected versions are first decoded to a graph and a tree, which are traversed using multiple techniques. It is not possible to consider all paths at calculation because the search space is too large. With the use of heuristics, the computation time can be reduced radically, but does not guarantee to find an optimal solution. To detect early agreement, the traversal is executed periodically while the synchronous crowd is modifying the pre-translation. Experiments demonstrated that the entire translation, improvement and alignment process has a minimal delay of 30 seconds.

This application of human computation demonstrates how a real-time crowd can be deployed. Synchronous crowds offer a short wait time until task completion. Moreover, an efficient alignment algorithm that can be used for further language processing methods is proposed.

1.5 Overview

The related work of this paper can be found at chapter two. In the third chapter, the functional principle of real-time translation is presented. Furthermore, the proposed solution contains a detailed description of the rapid refinement merge algorithm. The different alignment techniques are empirically tested and discussed in the experiments in chapter 4. We draw a conclusion in the fifth and last chapter.

2

Related Work

Three research areas are relevant to translating real-time subtitles using a crowd. First, it is important to understand the properties and features of human computation. Second, machine translation has been investigated for many years. Studies about coupling a crowd with MT systems have been published recently, where this thesis is based on. Third, the last section concentrates on the alignment of multiple phrases in natural language processing.

2.1 Human Computation

The power of a crowd was first presented by [Von Ahn and Dabbish, 2008]. They pooled human brainpower through computer games. While a mass of humans play *games with a purpose* (GWAP), side-effects are used for computation tasks. [Malone et al., 2010] showed that it is necessary to analyze a task from four perspectives: What, Who, Why and How. With these four building blocks, it is possible to have a crowd solving a complete task. Splitting the task into small, independent sub-tasks allows the crowd to work concurrently without interaction among each worker. Multiple results of workers are *collected* for each sub-task. The *decision*-phase determines a final solution. The decision phase in this paper will not be crowd-based, but a combining-algorithm of all collected results. This reduces the costs and time needed. [Malone et al., 2010] also focus the crowd workers from a psychological point of view. Why do crowd workers participate in the task? Obviously, money sometimes calls the shots, but there also could be a less-obvious reason. The Open-Source community is a possible example. Programmers do not receive money for fixing bugs in Linux[1] - they do it because they have fun doing it and because they can improve their own coding skills. This might lead to future monetary benefits. [Malone et al., 2010] reason the intrinsic motivation with the people's desire for love and glory. The task of improving bad texts is arid and needs more than the intrinsic motivation of probably improving the language skills.

In *Programming the global brain*, [Bernstein et al., 2012] pointed out that computation problems need to be adapted to the crowd. The diversity among crowd workers can be taken as an opportunity to achieve results that (super-)computers would never have

---

[1] http://www.linux.org

come up with. Cognitive diversity defines the strengths and weaknesses of individual workers. We try to use the variety for an alignment method, so that the best parts of the proposed solutions are merged.

2.2 Translation

The coincidence of a translation task and a crowd has already been shown by [Minder and Bernstein, 2012]. Their goal was to translate a book using a crowd in less time than a professional translator would have done it. The whole text has been translated with machine translation to the foreign language. They first let crowd workers correct the text sentence-by-sentence. In a second iteration, whole paragraphs have been re-merged to specify them more precisely within the context. As a third and last step, crowd workers corrected whole chapters to improve the golden thread. Ultimatively, they showed that the result is of good quality, but not ready for the press. Nevertheless, it is possible to translate whole books with a small budget, such that the reader can easily catch the meaning. They were able to translate 30 pages per hour. This cannot be broken down to calculate the time needed for one sentence because the whole text was processed in parallel. All parts must run through multiple stages, which causes major delay. In real-time applications, such a delay is not tolerable, thus this thesis tries to minimize the time used to for translation at sentence level.

2.3 Natural language processing

An impressive application of crowd-sourcing and natural language processing is made by [Liem et al., 2011]. They developed a speech-to-text engine that reaches 96.6% accuracy. As proposed in the *How*-block by [Malone et al., 2010], the spoken input was split into chunks of ten seconds. Similar to an A/B-test, the crowd was divided into two groups. Both groups processed the same parts redundantly with the result that their outputs can be compared. Iteratively, the transcript was improved until both clusters came up with the same solution. As already seen in [Minder and Bernstein, 2012], multiple iterations negatively affect the time needed. Other methods than iterations are needed in our case.

[Rosti et al., 2007], [Jayaraman and Lavie, 2005] and [Barzilay and Lee, 2002] tried to combine sentences generated from multiple machines (not only translations) into one better sentence without iterations. With different combining methods, the output did improve between 3% [Rosti et al., 2007] and 7% [Jayaraman and Lavie, 2005]. Instead of using pure machine translated phrases from different engines as an input for an aligning algorithm, we will use the crowd's diversity [Bernstein et al., 2012] to receive several versions. The sentence combination procedure differentiates from what is presented here, but we will use the same quality metric METEOR [Lavie and Denkowski, 2009], such that the results can be compared later.

[Pang et al., 2003] aligned sentences that likewise can be used to evaluate new syntax combinations on syntax-level. The method learns the syntax of each sentence with multiple variations and is not time-relevant. The results need to be further evaluated before they can be used for comparison.

3

Proposed solution

We want to build an engine that translates written text in close to real-time. It should be possible with the combination of a statistical MT system and a well-organized crowd. This section first shows how to solve all three problems introduced in section 1.2. After a brief overview of the processes of the proposed solution, we investigate the process of merging the collected data more precisely. The need for iterations, as proposed in [Minder and Bernstein, 2012] is eliminated. Approved methods to measure properties of a sentence are introduced and extended.

3.1 Approach to solve the stated problems

Translation needs a good knowledge and understanding of both the source and the target language. In a real world, it is difficult to establish a crowd that is able to translate, never mind that it must be in real-time. Therefore, the translation task is completely committed to the MT system. The advantages of a MT system are the high availability and the fast results. The first problem is solved by using MT instead of professional translators. Today's MT systems offer reasonable quality of terminology and vocabulary. Of course, human specialists exceed the quality by far. But this can be ignored because we mostly translate subtitles of television channels. Telecasts are formed such that a large part of the population is able to understand the content. This limits the amount of jargon used.

In regard to problem 2, it is not recommended to fully trust in MT. The crowd comes into play at this point: The crowd workers improve the pre-translated text in a second step. The only requirement for a crowd worker is to understand the target language, thus the crowd can be monolingual. The subtitles arrive as a stream, which is pre-translated with MT and improved by the crowd at sentence level. The workers also see the previous and the subsequent sentence because machine translated sentences may be totally fallacious. It is possible to improve the middle sentence so that it sounds correct and fits into the context. An improved sentence of a worker is called a *proposal*.

When processing the text sentence-by-sentence, the task can be split in many subtasks and becomes scalable for the crowd. Here, scalability means that multiple crowd instances can work on different sentences simultaneously. This tackles part of problem 3. Nonetheless, workers need some time to improve a single sentence. In the professional translation sector, it is conventional to have multiple iterations for a sentence. The first version might only be a rough text, which is further refined until perfection. As the time delay of a single iteration already conflicts with the term 'real-time', iteration is no option for us. The quality of the translation after one improvement step must be acceptable.

Multiple crowd workers concurrently work on the same task. With the use of *rapid refinement* an agreement among them can be recognized early [Bernstein et al., 2011]. The alignment does not wait until all workers submit their improved sentence because results need to be accessible quickly. Workers start writing at the beginning of a sentence. After a few words, snapshots of the current state of each worker can already be aligned. The further the workers proceed in a sentence, the more can be merged. With advanced working time, the aligned sentence becomes finer. [Bernstein et al., 2011] states that with rapid refinement, the alignment of the independent opinions can even lead to a faster result than the fastest worker. They went an extra mile and focused free resources of the crowd to critical sections of the task in real-time. For phrases, this concept is hardly adaptable. The crowd already works at sentence level. We think that further narrowing on critical sections would impair the result because of the lack of comprehension and context. To conclude, the third problem can be solved by using a synchronous crowd and a rapid refinement algorithm.

According to [Bernstein et al., 2012], people's rational and accurate performance varies widely. Proposals of crowd workers differ. The proposition is to use redundancy: multiple workers improve the same phrase at the same time and the results are merged in the end. This purges the need for multiple iterations. Another advantage is the reduction of manipulation or misuse by the crowd. Some workers may try to benefit from solving the task incorrectly or incompletely.

3.2 Overview of the process

Figure 3.1 shows the previously described components and how they play together. Assuming that subtitles arrive sentence-by-sentence, they are pre-translated with MT. This raw version of the sentence in the target language is distributed to a set of crowd workers. For good results, at least three crowd workers should work on the same phrase concurrently. When each worker believes he is done with the improvement, he submits his solution and proceeds with another task. Between the start and the submittal, snapshots are sent to the merge process, so that this component can already start to align. A detailed description of the merge process can be found below. When the rapid refinement algorithm detected an agreement among the workers, the task is stopped. The output is the translated and improved sentence.

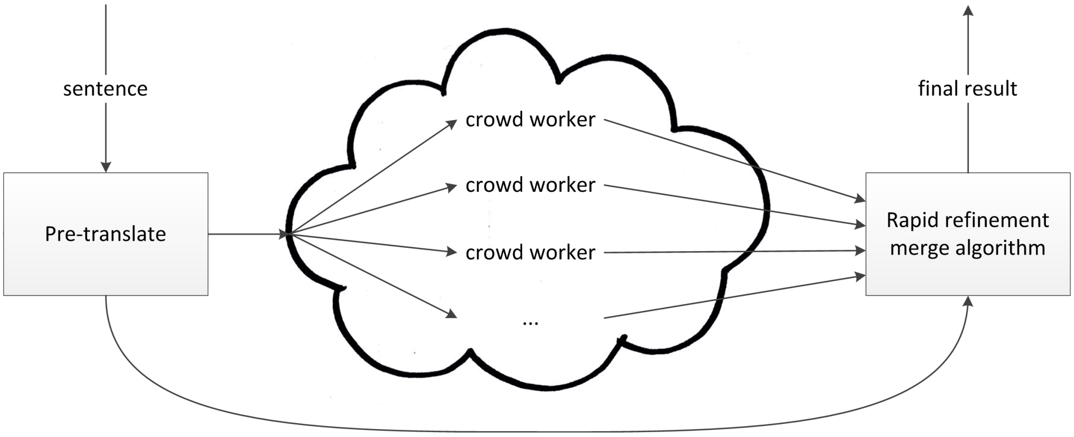


Figure 3.1: Overview of a crowd-based translation

3.3 Rapid refinement merge algorithm

The process of combining multiple proposals into a single result is undoubtedly the most interesting part of the process. According to the second hypothesis, the result has to be of better quality than the pre-translated input. To simplify the alignment, the proposals are converted to two common data structures: a graph [Barzilay and Lee, 2002] and a tree. The nodes contain one or multiple words and the edges hold the weights. First, we introduce the basis, describing the input and these two data structures. To process natural language, three models and their combination are presented. Second, when we have the basic knowledge about the linguistic characteristics of words in a context, multiple traversal algorithms are proposed to find the best path. In the end, the collaboration of the traversal algorithms and rapid refinement is explained in more detail.

3.3.1 Basis

For a common knowledge base, the input of the alignment algorithm is described first. Next, the used data structures, *graph* and *tree*, are introduced in the context of this work. Last but not least, characteristics of natural language are applied to the nodes of the data structures.

Input

Subtitles are bound to a specific timestamp in the movie. With time, they arrive in a data stream sentence-by-sentence. As future subtitles are not known and the delay of translation should be kept as short as possible, the subtitles are translated sentence-wise. The crowd corrects each sentence separately. A working example that has been pre-translated and corrected by three workers can be found in table 3.1.

original (German)	”Ich war vor 21 Jahren in Neuseeland”
pre-translated	”I am 21 years ago in New Zealand.”
worker 1	”I was in New Zealand 21 years ago.”
worker 2	”21 years ago, I was in New Zealand.”
worker 3	”I have been in New Zealand 21 years ago.”

Table 3.1: An erroneous pre-translated sentence has been improved by three crowd workers. Each worker submitted a different proposal.

Data structures

To process the different versions of the corrected sentences, a good organization is needed. The sentences are split up into chunks of words and decoded to a graph and a tree. Both data structures are described below. They store the information redundantly, but can be used to align the proposals with different algorithms. Depending on the alignment method, one of the two structures is used.

Graph At the beginning, the graph contains only a start- and an end-node. The incoming proposals are processed separately and split into chunks of one word. For each word in the sentence, a new node n_g is created. If the graph already contains the word, or a synonym of it, no new node is inserted. When all words of the proposals have been added to the graph, the edges e_g can be formed. Subsequent words in proposals are connected with directed edges. The weight of an edge is increased if multiple workers agree to the word sequence. The end-node in the graph is an *absorber*, meaning that it has no outgoing edges. The creation process is also known as *confusion network decoding* [Rosti et al., 2007] and the result is a directed weighted graph.

The edge weighting strategy can be further expanded. The crowd is given a pre-translated sentence and makes modifications. These changes can be seen in the graph when the original sentence is overlaid. Drifting edges can be either devalued or enhanced with an *enhancement factor*. If an alternating edge is devalued, the weight decreases by a factor, such that weights of the pre-translated sentence appear stronger. Depending on the extent of the factor, alternative paths may disappear completely. On the other hand, variation can be rewarded by enhancing the weight of differing edges.

Table 3.1 shows an input sentence and multiple proposals. The resulting graph for this exemplary input is shown in figure 3.2. Note that no devaluation or enhancement is applied. For simplicity, multiple words are pooled to one node, if all proposals agreed on their sequence. For example, the workers agreed with the order of words ’21’, ’years’ and ’ago’. Instead of chaining three nodes, the words can be aggregated in a single node ’21 years ago’.

Tree The second data structure to illustrate the proposals is a tree. Cycles in the directed graph of section 3.3.1 are unavoidable. They arise when a proposal starts with a part of the phrase that another proposal ends with. The example graph in figure 3.2

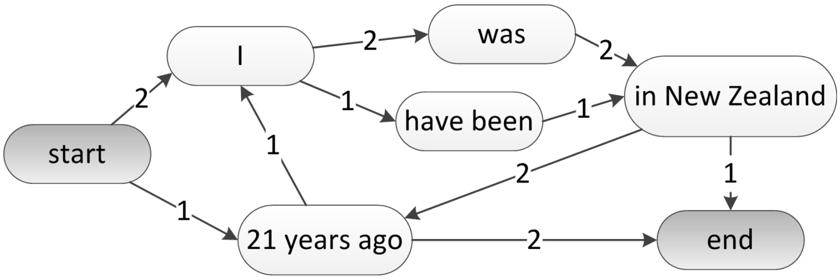


Figure 3.2: Directed weighted graph of the example.

also contains multiple cycles. Finding the optimal path in a tree is easier than it would be in a graph because of that characteristic. The mapping begins with the start node as the root of the tree. The children, which have not already been visited, are successors of the current node n_t. It is allowed that a node of the graph appears multiple times in the tree, but they have to be in different branches. The weights of the edges are taken over from the graph. In the tree, an end-node is always a leaf node. The number of leaves equals to the number of paths in the graph. A tree node has, in contrast to a graph node, not necessarily the same total incoming and outgoing weight. An entangled graph leads to a large tree. When the number of graph nodes increases linearly, the tree grows exponentially.

The full tree of the proposals in table 3.1 is shown in figure 3.3. The tree has seven leaves, which equals exactly the number of different paths (without visiting a node twice) in the graph 3.2.

Properties of a node

The weights of the edges can already be used to find the best path. For a more sophisticated traversal, additional characteristics have to be found. In the following, basic concepts of computational linguistics are introduced. They are widely used in MT.

Three model components Statistical MT engines work with three different models to generate the best output. [Koehn, 2010] introduced them to calculate the probability of a translation. Multiple machine translated versions of a text can be compared with this model. All three models yield a certain probability, which can be combined.

Language Model p_{LM} [Koehn, 2010, p. 181-187]: The language model describes the quality of the sentence in terms of fluency. To measure how likely it is that one word follows others, we use n-gram as the metric. N-grams describe the probability p_{LM} that a specific word follows a chain of other words. Any number can replace the 'n' in n-gram. Unigrams only describe the probability that a single word occurs. For example, common words like 'and' or 'the' have a higher probability than 'intergalactic'. But typically, bigrams or trigrams are taken to measure

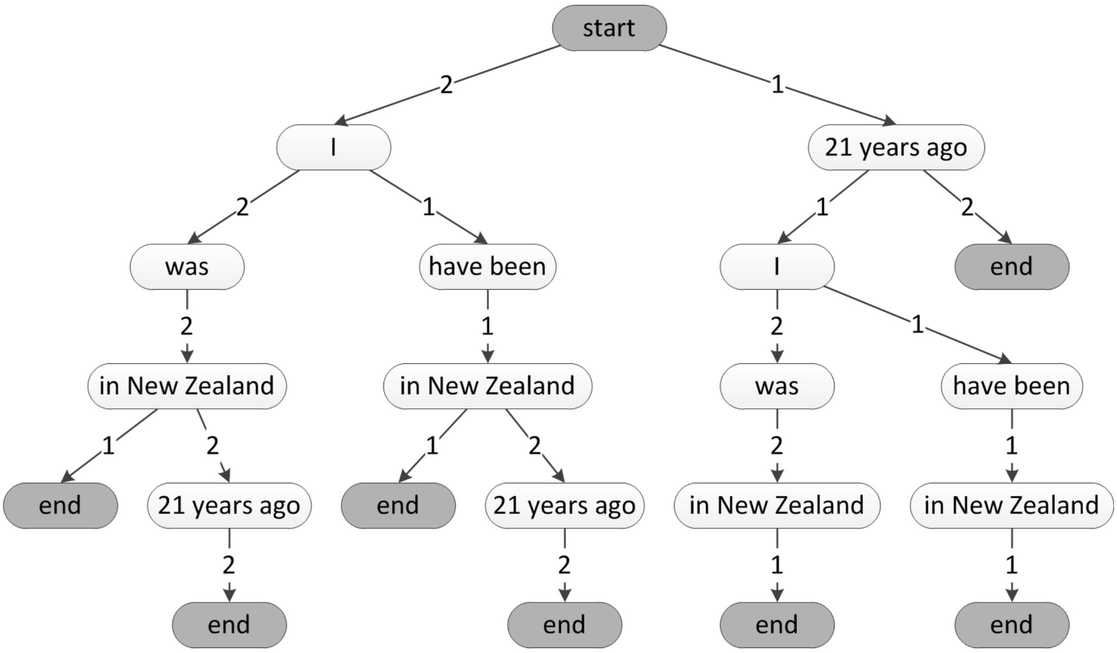


Figure 3.3: Full tree of the example

the language quality. Bigram p(w2 | w1) shows the possibility that w2 follows w1. Trigrams consider two preceding words and so on. Correct word orders are preferred to incorrect word orders:

$$p_{LM}(\text{I feel good}) > p_{LM}(\text{Feel I good})$$

Reordering Model ϕ [Koehn, 2010, p. 129-130]: A sentence in a wrong order is difficult to understand, and can even have a completely different meaning. The order of the input can differ from the order of the output. MT systems perform the reordering also by learning about word orders. The order may be statistically correct, but sometimes, the phrase sounds unnatural for the readers. Therefore, the crowd corrects it. Let's consider the example of table 3.1: Two of three workers think that the *where* should be before the *when*:

$$\phi(\text{I was in New Zealand 21 years ago}) > \phi(\text{I was 21 years ago in New Zealand})$$

In this thesis, the reordering probability of the crowd's proposals is not based on learning. The reordering probability from node n_{i-1} to n_i is calculated with the weights of the outgoing edges. S is the number of succeeding nodes.

$$\phi(i|i-1) = \frac{weight_{i-1 \to i}}{\sum_{n=1}^{S} weight_{i-1 \to n}} \tag{3.1}$$

Translation Model d [Koehn, 2010, p. 81-87]: A bilingual dictionary often contains multiple translations for a word. How does a machine know which translation fits the best? The computer's answer is based on statistics. In contrast to the language model, where the focus is on the combination of multiple words, the focus of the translation model is on the individual words. Depending on the context, one translation may be more suitable than another. A statistical MT has a learning algorithm that collects the data.

How can these three components be combined? With the theory about joint probability of independent events, we have

$$p(a, b) = p(a) * p(b) \tag{3.2}$$

meaning that both a and b happen, but are independent of each other. Applied to a sentence, the probability can be calculated word-by-word. According to equation (3.2), going through all words I and multiplying the probabilities results in the joint probability of the sentence. [Koehn, 2010, p. 157-158] combined the three components multiplicatively (3.3). The joint probability for each possible translation e is calculated; the maximum probability is the best result.

$$e_{best} = \max_e \prod_{i=1}^{I} p_{LM}(e) * \phi(i|i-1) * d(e) \tag{3.3}$$

Since the translation has already been done by a MT, the *translation model* is already covered. Workers can still remove unsuitable words and replace them with words that are more adequate in the context, but this is covered by the reordering probability (3.1). Alternative words are detected at the synonym-check when inserting them to the graph. Thus, the translation model component can be ignored:

$$e_{best} = \max_e \prod_{i=1}^{I} p_{LM}(e) * \phi(i|i-1) \tag{3.4}$$

Equation (3.4) is only suitable for combinations of equal length. With

$$p_{LM} \in [0 \dots 1] \text{ and } \phi \in [0 \dots 1]$$

we can say that

$$\prod_{i=1}^{I} p_{LM}(e) * \phi(i|i-1) \geq \prod_{i=1}^{I+1} p_{LM}(e) * \phi(i|i-1) \tag{3.5}$$

In other words, longer translations have smaller probabilities. In regard to the trees and graphs, path lengths can vary widely. The maximization of equation (3.4) does not hold here because it would tend to output the shortest path. Multiple approaches are suggested later in section 3.3.2.

Timestamps While the crowd types the improved phrases, multiple timestamps can be recorded. When is a particular word typed first? Are important modifications made first? But words appearing later could be better because workers corrected them more recently. Timestamps of words depend on the writing speed and the location in the phrase. As this is only guessable and has no reliable metric, it is not considered in further calculations.

3.3.2 Algorithm

We know how to transform the proposals into a graph and a tree and are able to appraise sentences from a linguistic perspective. This section introduces one algorithm for a graph traversal and two algorithms to traverse the tree. In the tree traversal, four quality estimation methods are introduced.

Graph traversal

The graph nodes have to be relinked to form a new, better sentence. To make only local decisions at the current node is a straightforward traversal algorithm. [Russell et al., 1995, p. 95-97] call the algorithm *greedy best-first search*. The probabilities of each successor are computed and the most probable successor is chosen. This procedure is repeated until the end-node is reached. To avoid circles, it is not allowed to visit any node twice.

There are multiple ways to compute the best local successor. If the decision is only based on the reordering probability (3.1), we always follow the path where the most workers agree, thus the edge with the highest weight. Coming back to the example, the applied traversal algorithm is marked in figure 3.4. The output is 'I was in New Zealand 21 years ago'.

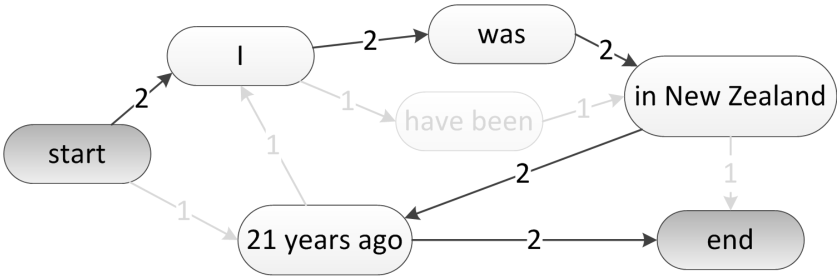


Figure 3.4: Greedy best-first search (only considering the weights) applied to the example

The computation can be extended by the language model component. Since the two models are independent, equation (3.2) can be applied. The decision falls again on the most probable node:

$$n_{next} = \max_{i=1}^{N}(\phi(i|i-1) * p_{LM}(i|i-1)) \tag{3.6}$$

N-gram	p_{LM}(successor| 'I')
('was' | 'I')	0.15
('have been' | 'I')	0.45

Table 3.2: N-grams for successor nodes of node 'I' in the example.

The decision is now not only based on the input of the workers, but also on n-gram statistics. Applied to the example, we focus on the decision of the node with the text 'I'. With the equation (3.1), the best successor is 'was'. This changes with equation (3.6) and the relevant n-grams in table 3.2:

$$\begin{aligned} p_{was} &= \frac{2}{3} * 0.15 = 0.1 \\ p_{\text{have been}} &= \frac{1}{3} * 0.45 = 0.15 \end{aligned}$$

As $p_{\text{have been}} > p_{was}$, the chosen successor is 'have been'. The output now starts with 'I have been'.

It is not guaranteed to find the best path. The successor selection process is only based on a local heuristic. It selects the node where the most future profit is expected. Pruning paths early is risky. A path may start with low probability, but gains a lot of quality later. On the other side, making only decisions within a small range of vision affects the complexity in a positive way. The upper limit of nodes to visit is N_g. The complexity $O(N_g)$ is linear to the number of nodes in the graph.

Tree traversal

There are two advantages of a tree over a graph: First, no cycles exist and second, all possible paths are already apparent. However, the tree grows quickly with increased graph complexity. Presuming that it is not possible to hold the whole tree in memory, an algorithm has to be found, which expands the tree at nodes which look promising. After the greedy best-first search of the graph, this section introduces several tree traversal algorithms.

As nodes appear duplicate in the tree, but only once in a path, they can be used to hold additional path-specific attributes. These attributes are directly computed when the tree node is created:

- depth (= path length)
- accumulated weight
- accumulated language probability $\sum p_{LM}$
- accumulated reorder probability $\sum \phi$

The accumulated probabilities are the sums of all probabilities above the current node. The probabilities are combined additively because of the problem detected with equation

(3.5). Summing the probabilities over-tilts the effect. Long paths have a higher probability than short paths. To compensate, a factor is introduced in A*-section. Two possible tree traversal algorithms are presented below. The well-known depth-first search and the A*-search.

Depth-first search The idea behind the depth-first search is to search the best path in the whole tree. We iterate through all possible paths and discard the poor ones. An advantage is that all paths are taken into consideration. Starting at the root node, all successors are determined and further expanded recursively. If an end-node is reached, it is stored and the algorithm backs off one level, so that the next successor can be expanded. Every time a leaf is reached, it is compared to the current best path in memory. If it is worse, the new path is discarded. If it is better, the latter is superseded with the new path. Note that only one path is held in memory at a time. In difference to the *greedy best-first search*, the *depth-first search* continues searching after finding an end-node.

While progressing through the tree, the best path always improves. Especially in real-time applications, it may be necessary to have an upper time limit. The algorithm can be stopped at any time, being able to output the current best path. This path may not yield the best, but at least a good result.

The mentioned attributes of a node can be used to compare two nodes in different combinations. Four different quality estimation methods are presented here and empirically tested in chapter 4.

Total weight w_{total} : The accumulated weight of the traversed edges can be stored at each node. The end node contains the total weight of a path. Maximizing the total weight is also known as *the longest path problem*. The total weight can be written as

$$w_{total} = \sum_{i=1}^{depth} weight_{i-1 \to i} \tag{3.7}$$

The drawback of this method is that longer paths have higher total weights. But in natural language, the longest sentence is not always the best. The maximization tries to cover as many nodes as possible instead of only focusing on the important ones.

Average weight w_{avg} : This approach maximizes the average number of workers covered with a path. It eliminates the drawback of the method *total weight*:

$$w_{avg} = \frac{w_{total}}{depth}$$

Output sentences tend to be short because the chance of fragmentation is higher on long paths than on short paths. Higher fragmentation results in lower weight on each edge, which reduces the average weight.

Sum of probabilities (multiplicative) $\sum p_{LM} * \phi$: Equation (3.2) states that multiple events have to be combined by a multiplication of their probabilities. The reordering model ϕ, known from (3.1), and the language model p_{LM} are combined. Having the joint probability, the concept of maximizing the total weight can be carried on: All joint probabilities of the nodes above are summed:

$$\sum p_{LM} * \phi = \sum_{i=1}^{depth} \phi(i|i-1) * p_{LM}(i|i-1)$$

Similar to *total weight*, the sum increases with growing deepness. In contrast, the weight also depends on the language model.

Sum of probabilities (additive) $\sum \frac{p_{LM}+\phi}{2}$: This method is similar to its multiplicative counterpart with the difference that the probabilities are not multiplied after the rule of joint probabilities but averaged.

$$\begin{aligned}\sum \frac{p_{LM}+\phi}{2} &= \frac{1}{2} * \sum_{i=1}^{depth} \phi(i|i-1) + \frac{1}{2} * \sum_{i=1}^{depth} p_{LM}(i|i-1) \\ &= \frac{1}{2} * \sum_{i=1}^{depth} \phi(i|i-1) + p_{LM}(i|i-1)\end{aligned}$$

Regardless of the traversal algorithm, the number of nodes N_t in the tree is higher than the number of nodes N_g in the graph. In total, more nodes have to be visited. To compute the complexity, we consider a complete graph as the worst case scenario. Starting at the root, $N_g - 1$ nodes have to be visited. Each of the $N_g - 1$ nodes has $N_g - 2$ children. This scheme is continued until the maximum depth, which is $N_g - 1$, has been reached.

$$\begin{aligned}N_t &= (N_g - 1) + (N_g - 1) * (N_g - 2) + (N_g - 1) * (N_g - 2) * (N_g - 3) + ... \\ &= \sum_{i=1}^{N_g-1} \prod_{j=1}^{i} (N_g - j)\end{aligned}$$

No heuristics are applied at the depth-first search. To find the optimal solution, all tree nodes have to be visited. In the worst case, the time complexity is

$$\begin{aligned}O(N_t) &= O(\sum_{i=1}^{N_g-1} \prod_{j=1}^{i} (N_g - j)) \\ &= O(N_g^{depth})\end{aligned} \tag{3.8}$$

The graph in figure 3.2 has seven nodes. If the nodes were fully connected, the tree would have 1957 nodes to process. Compared to the linear complexity of the *greedy best-first search* in the graph, the tree search needs much more computation time.

A* search A* search [Russell et al., 1995, p. 97-105] is an altered form of the best-first search. It is introduced because the runtime complexity of searching the whole tree shown in equation (3.8) is not acceptable. The A* search uses two different functions: the already traversed path $g(i)$ and a heuristic $h(i)$. The heuristic estimates the path from node i to the end-node. The heuristic and the current path form the function $f(i)$:

$$f(i) = g(i) + h(i) \tag{3.9}$$

$f(x)$ is maximized to find out, which node to expand next [Russell et al., 1995]. The heuristic is crucial to find the optimal solution. If the heuristic is too small, the estimation $f(i)$ is also too small. This may give other worse nodes the chance to expand and return a suboptimal solution. The heuristic has always to be higher than estimated.

The effective processed path $g(i)$ can be measured with the *total weight* of equation (3.7). An *admissible* heuristic $h(i)$ guarantees to find the best path [Russell et al., 1995, p. 97], wherefore it must always be greater than the real total weight. To estimate the final weight, the length of the future path has to be known. As the pre-translated version of the MT and multiple proposals are known, a median length $\bar{n}$ can be taken as an estimation. Short paths with small $g(i)$ have a higher $h(i)$ than long paths because there are more nodes which still have to be visited. However, if the current path is already too long, the heuristic should be either small or negative, so that they are not expanded further. The weight of $g(i)$ increases linearly with the depth, $h(i)$ should decrease accordingly. We define a boost function $b(n)$ where n is the depth of the inspected node:

$$b(n) = \begin{cases} \bar{n} - n + 1 & \text{if } n \leq \bar{n} \\ 1 & \text{if } \bar{n} < n \leq \frac{5}{4}\bar{n} \\ \frac{5}{4}\bar{n} - n & \text{if } \frac{5}{4}\bar{n} < n \end{cases} \tag{3.10}$$

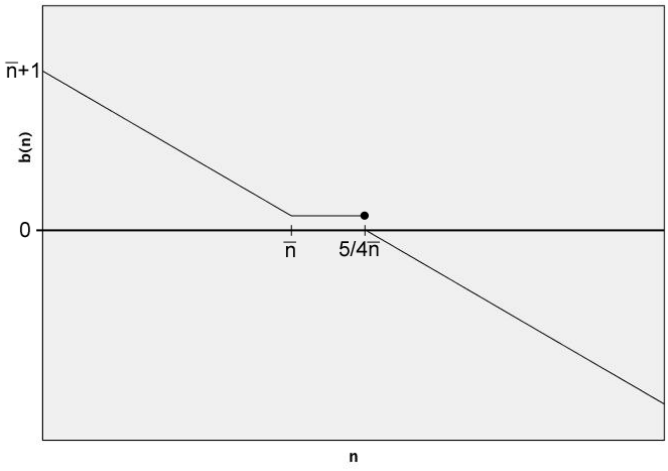


Figure 3.5: Plot of the boost function $b(n)$

If the depth of the inspected node is smaller than the estimated depth, the boost behaves linearly and returns the estimated number of hops until the end of the sentence. If the path is already too long, there is a confidence interval of 25% of the estimation given because slightly longer phrases can also be correct. After surpassing this interval, the boost turns into negative, such that it is not worth expanding this node any further. A plot of $b(n)$ is shown in figure 3.5.

The heuristic function should give a guessed value of the weight that can be gained by expanding the node i. We define a first version of the heuristic:

$$h_1(i) = b(depth_i) * weight_i$$

Not only the current weight, but also the reordering and the language model should be considered. The function is extended by these two probabilities:

$$h_2(i) = b(depth_i) * weight_i * (\phi(i|i-1) + p_{LM}(i|i-1)) \tag{3.11}$$

ϕ and p_{LM} are added (and not multiplied) because it is desirable to overestimate. The component's range analyzed separately helps to find out whether the heuristic is an overestimate in all cases or not:

$$\begin{aligned} b(n) &\in [1 \dots \bar{n}] \text{ (for reasonable phrase length)} \\ weight &\in [1 \dots |hyp|] \\ \phi(i|i-1) &\in [\frac{1}{|hyp|} \dots 1] \\ p_{LM}(i|i-1) &\in [0 \dots 1] \end{aligned} \tag{3.12}$$

If no end-node has been found yet, at least one further node is expanded, so $g(i)$ is increased by the weight of the current node. So we can say that $weight_i$ is the minimum estimation for one hop. Equation (3.11), filled with the minimum values of (3.12) results in

$$\begin{aligned} \min h_2(i) &= 1 * 1 * (\frac{1}{|hyp|} + 0) \\ &= \frac{1}{|hyp|} \end{aligned} \tag{3.13}$$

To prevent that $h(i)$ falls below the minimum estimation $weight_i$, it has to be compensated by the reciprocal value of the detected constant minimum:

$$\begin{aligned} h_3(i) &= h_2(i) * \frac{1}{1/|hyp|} \\ &= b(depth_i) * weight_i * (\phi(i|i-1) + p_{LM}(i|i-1)) * |hyp| \end{aligned} \tag{3.14}$$

The A* search applied on the example in table 3.1 is shown step-by-step in figure 3.6. Initially (a), the tree only contains the root node. $g(x)$ and $h(x)$ of both children are calculated. The node 'I' is more promising because $f(x)$ has to be maximized. After visiting the children of 'I' (c), the three nodes 'was', 'have been' and '21 years ago' are at choice. Since 'was' has the highest $f(x)$, it is visited next. After calculating the heuristics for 'in New Zealand', we see that the previously visited node '21 years ago' now scores the best (d). This procedure goes on (e and f) until the most promising node is an end-node (g). The final path 'I was in New Zealand 21 years ago' is found.

The use of a heuristic reduces the number of nodes visited. The complexity of the A* search is coupled with $h(i)$. In the worst case, all nodes have to be visited, like seen in equation (3.8). Having the *perfect heuristic* $h^*(i)$, [Russell et al., 1995, p. 101] shows that the error of the used heuristic above can be limited with

$$|h(i) - h^*(i)| \leq O(log(h^*(i))) \tag{3.15}$$

It seems pleasant that the error of a heuristic function can be isolated, but the real problem of the A*-search is its memory usage. To find out which node has to be expanded next, all visited nodes have to be held in memory. Pruning bad paths early can solve the problem partly. The issue is not discussed any further because sentences are limited to their length.

3.3.3 Rapid refinement

With rapid refinement, we try to output a solution as quickly as possible. The graph and the tree can be constructed when first snapshots arrive. Following snapshots can be inserted dynamically in the graph. New words (nodes) can be inserted and edges can be connected without affecting other nodes. More nodes and more possibilities lead to a better output.

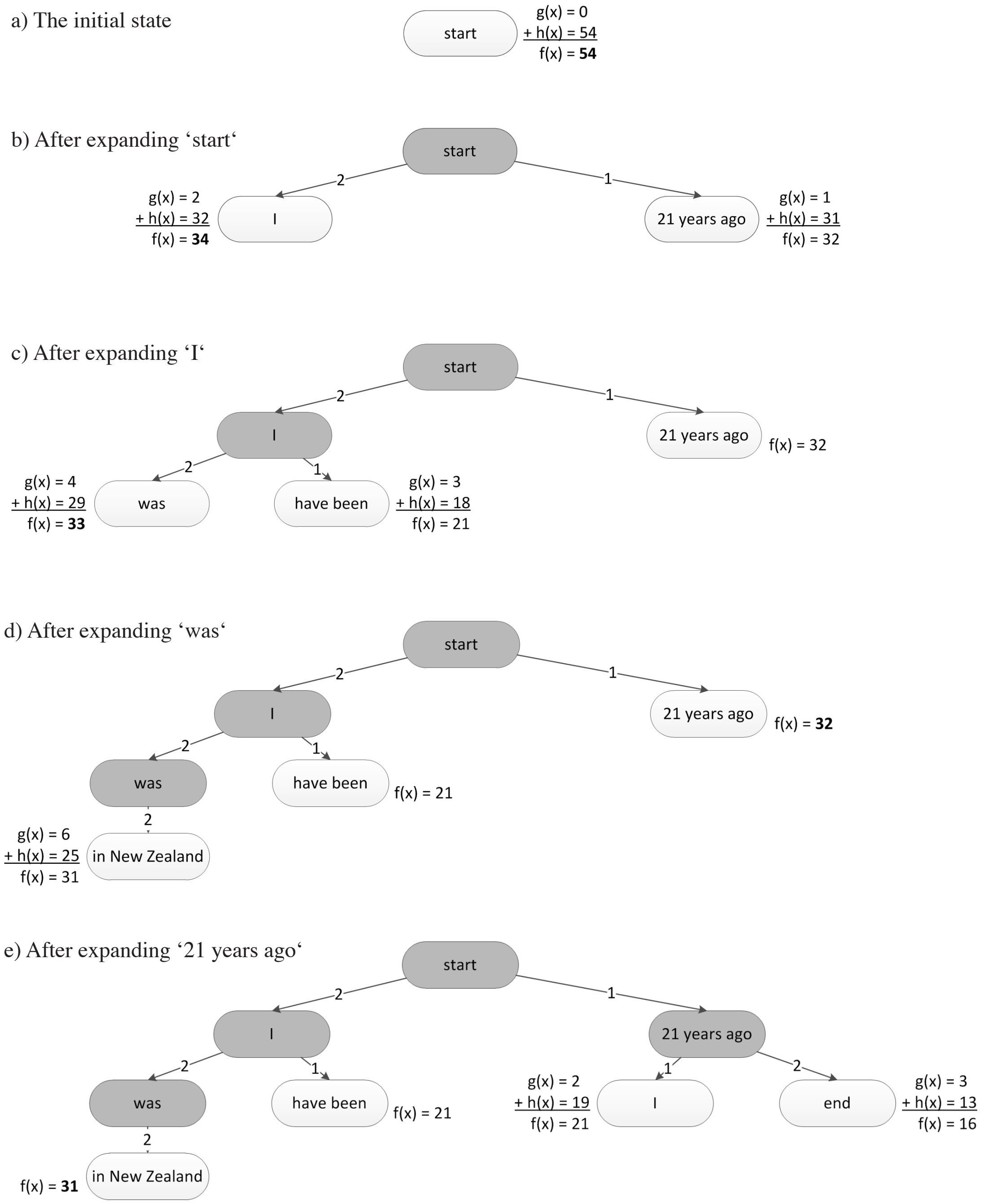
Adding snapshots to the tree successively is slightly more complicated. For one added graph node, multiple new paths emerge which need to be appended to the tree. The predecessors of the new node n_{new} have to be found in the tree, where the n_{new} is inserted as a child. For each instance of n_{new}, the new resulting paths are then added. Moreover, weights of the direct successors of n_{new} have to be adjusted.

To have a constant refinement of the best path, the graph or the tree has to be traversed multiple times while the workers are improving the sentence. The use of rapid refinement infers that the outcome may change when new nodes and edges are added dynamically. The traversal is conducted regularly until one of the following points is true:

- The quality of the result does not improve anymore (an agreement can be assumed)
- All assigned workers have submitted their proposals
- A timeout occurs and an instant result is needed

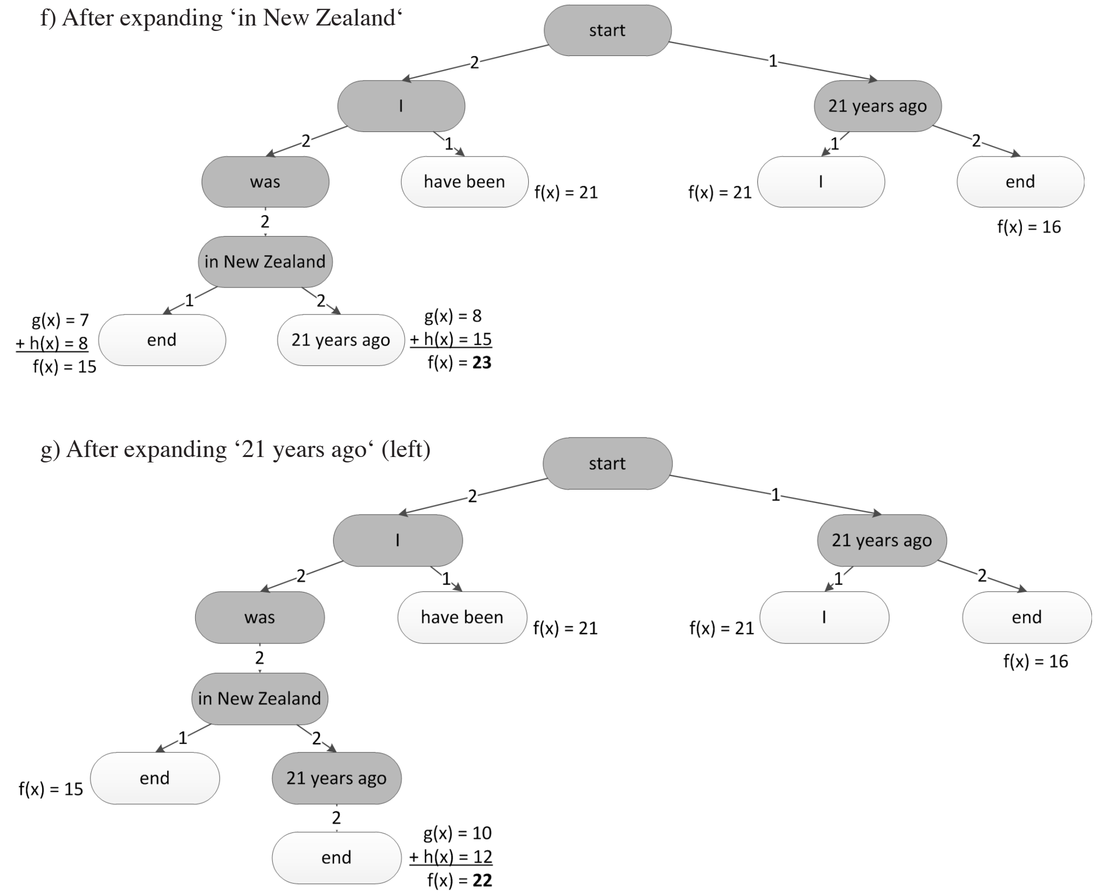
After stopping the merge, the output is assumed to be of good quality. The workers can also finish improving the current sentence and wait until the system needs them for the next tasks. The rapid refinement algorithm requires that enough resources can be bundled for the same phrase at the same time.

The regular execution of the traversal emphasizes the search costs even more. The second hypothesis tries to find out how the quality of the output and the needed time behave. With each repetition of the alignment, an incremental improvement of the quality is expected. Experimental results can be found in section 4.3.2.



(a)

Figure 3.6: A* algorithm applied to the example



(b)

Figure 3.6: A* algorithm applied to the example (continued)

4

Experiments

Chapter 3 introduced the procedure of real-time translations with the use of human computation. Multiple alignment algorithms which are tested in this chapter have been proposed. First, the setup of the experiment is explained. Then, the the results are shown and described. In the end, a discussion clarifies the limitations and describes points for future work.

4.1 Experimental setup

We came up with three hypotheses that need to be verified or rejected. In the first hypothesis, we want to know how the crowd behaves when correcting a pre-translated sentence. The crowd is divided into two groups for an A/B-test. Both groups have to improve the same pre-translated sentences, but with different conditions. The two groups are called 'A' and 'B':

A: The crowd worker sees three sequencing pre-translated sentences and has to improve the middle one using a text box. The text-box is already filled with the sentence to improve, such that only the erroneous parts have to be modified. If finished, the 'Submit' button has to be pressed.

B: The user interface looks exactly as seen in group A. The only difference is that the text-box is not already filled-in. Workers have to retype the whole sentence. This group is not tempted to simply accept the pre-translated, but instead to be courageous to write a different, perhaps better, phrase.

A short description is given at the start, so that the workers know what they have to do. The user interface can be seen in figure A.1. It is designed modestly to make the workers focus on the task.

To be able to simulate the improvement procedure later, a snapshot of the current state is stored every three seconds. To avoid stressing the workers, they are not told about the snapshot storage. The snapshots allow us to test the second hypothesis. The progress of all workers on a specific phrase can be passed to any alignment method to test the rapid refinement merge algorithm. Thus, a real-time crowd can be simulated. In addition to the snapshots, the total working time of each task is recorded. To prevent misuse, a minimum sentence length, dependent on the reference's sentence length, must be typed. For group A, a minimum time is introduced. The workers must obey a minimum waiting time of ten seconds before submitting a proposal. Although a more sophisticated malpractice detection mechanism could have been used, we think that those two restrictions act as sufficient deterrent.

The third hypothesis is tested on the final proposals submitted by the crowd. In an offline analysis, the worker's proposals can be tested against a reference translation with METEOR (see section 4.2). The worker's results are expected to be better than the pre-translation. The proposed alignment algorithms are scored against the reference and compared with the optimal alignment.

In an attempt to simulate a viable application as good as possible, the test set used is a compilation of multiple talks from TED[1]. Spoken language differs from written language in terms of style, syntax, diction and length. Usually, spoken language is easier to understand and to follow, but contains more slang. All TED-talks are held in English and contain transcripts. TED's Open Translation Project lets volunteers around the world translate talks about different topics. At the beginning of the project, TED engaged professional translators to ensure a good quality. A professional German translation from TED is the input for the experiment. We use the original transcripts as a reference. The output of the translations can then be compared with the reference. Figure 4.1 illustrates the process.

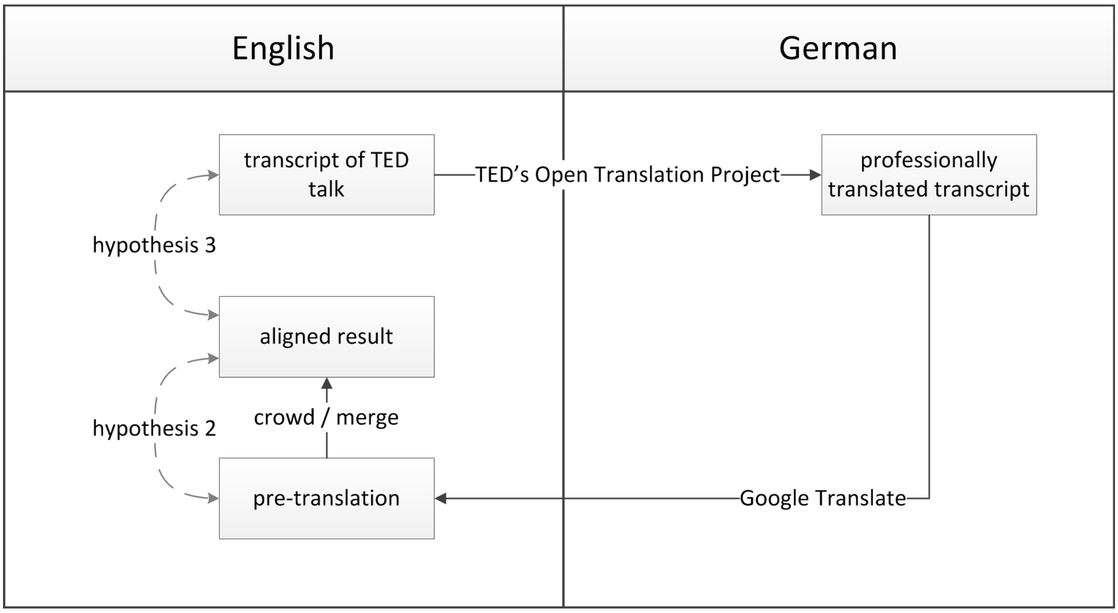


Figure 4.1: Overview of the used transcripts and translation processes. Investigated relations are marked with the corresponding hypothesis.

---

[1] http://www.ted.com

In total, 100 sentences out of five different talks are used. The first 20 sentences are used for the A/B test. The remaining 80 phrases are improved only by group B. This is done for two reasons: Assuming that workers are lazy, the pre-filled text will often be accepted. It is not clear anymore, if the pre-translation is of good quality or not. Besides, group B includes the natural progress of writing that is needed to test the rapid refinement.

Google Translate is used as the MT engine to pre-translate the sentences. It offers a fast translation and good availability.

Many proposed alignment algorithms include the language model. The statistics for the bigrams are taken from the Microsoft Web N-Gram Service[2], which is currently in beta stadium. The statistics are based on nine months of Bing[3] queries.

The experiment is run on Amazons Mechanical Turk (MTurk). A large crowd and therefore a fast task completion are the advantages of this crowd-sourcing platform. Every sentence is improved by ten different workers, but workers are allowed to improve multiple sentences. The task is monolingual and no special requirements are necessary. Each assignment has a time limit of five minutes and is rewarded with 0.05$. The experiment started at the 3^{rd} of December 2012 at 13:15 CET and ended at 20:00.

The alignment algorithms are implemented in Java using Jung[4] as a graph framework. The data are stored with PostgreSQL[5] and evaluated with Microsoft Excel[6] and IBM SPSS[7].

4.2 Quality metric

There is a need for a quality metric for a produced sentence. One possibility is to conduct manual evaluations by human judges [Koehn, 2010]. Bilingual evaluators are expensive and not always available. Moreover, scores are subjective and vary from person to person. Experimenting with this research would take too much time if people evaluated it. An automatic evaluation is used instead. A machine needs a point of reference like a professional translation. The original transcripts of the talks are used as the reference in our experiment. There exist multiple metrics to compare two texts. Two main components for an automated evaluation are *precision* and *recall*:

precision [Koehn, 2010, p. 223]: It measures the number of words that occur in the output and the reference text:

$$precision = \frac{\text{matching words}}{\text{output-length}} \tag{4.1}$$

---

[2] http://web-ngram.research.microsoft.com

[3] http://www.bing.com

[4] http://jung.sourceforge.net

[5] http://www.postgresql.org

[6] http://office.microsoft.com/excel

[7] http://www.ibm.com/software/analytics/spss

recall [Koehn, 2010, p. 223]: If the reference text is a dictionary containing all words, equation (4.1) would return 100% correctness. Precision only regards the length of the output text, not the length of the reference length. Thus, the *recall* is introduced:

$$recall = \frac{\text{matching words}}{\text{reference-length}} \tag{4.2}$$

The order of the words is not integrated in both scores. The evaluation metric BLEU (Bilingual Evaluation Understudy) [Papineni et al., 2002] is based on the precision and considers the word order with the use of n-grams. The n-grams are not based on global statistics, but on the reference text. If an output and the reference have long sequences of words in common, the BLEU score is high.

METEOR [Lavie and Denkowski, 2009] is a recent metric that concentrates more on the recall [Koehn, 2010, p. 228]. The scoring includes also word stemming and synonym detection with WordNet[8]. The scoring with METEOR is much more complicated than with BLEU. We use METEOR in this evaluation because it hones multiple weaknesses of BLEU. All scores are in the range between 0 and 1, where 1 stands for a 100% match.

4.3 Results

The sentences were improved by a total of 177 different crowd workers. Each worker edited an average of 6.8 sentences; the worker with the most assignments submitted 82 phrases. The outcome of the experiment is analyzed separately for each of the three hypotheses.

4.3.1 Hypothesis 1

The first hypothesis is tested with the A/B test with a sample size of 20 phrases. As presented in the experimental setup, group A has to only improve parts of the sentence, while group B has to retype it completely. The difference of quality between A, B and the pre-translated sentence from Google Translate are compared first. Then, the average working time and delay are addressed. Note that no alignment algorithm is applied here.

Quality of the improved sentences

Figure 4.2 shows the METEOR scores of the input (Google Translate, 29%), group A (27%) and B (22%) plotted against each other. The scores of all workers and sentences are averaged, thus each bar represents the mean of 200 meteor scores. Surprisingly, the scores decreased after modifying the version of Google Translate. In particular, the results of group B dropped deeply. The 95%-confidence interval shows that the scores of group B vary heavily. In contrast, with the improvement of group A, the variance has been reduced. In total, a METEOR score of only 30% is low, but since we are

---

[8] http://wordnet.princeton.edu

only comparing separate sentences, it is acceptable. Longer texts would lead to a higher score.

At closer sight on the proposals of group B, the lower score in figure 4.2 can be partly reasoned with the corrupt orthography. Typing errors are not detected by METEOR and lead to worse scores. Besides, workers often wrote in lower case only. Since METEOR uses a *normalization*, the score is not affected by wrong capitalization [Lavie and Denkowski, 2009]. Other workers did improve the first sentence instead of the middle one. Figure A.2 shows the median scores. In contrast to the average value, the median is not biased by outliers. The median scores have the same trend, but are more balanced than the average values.

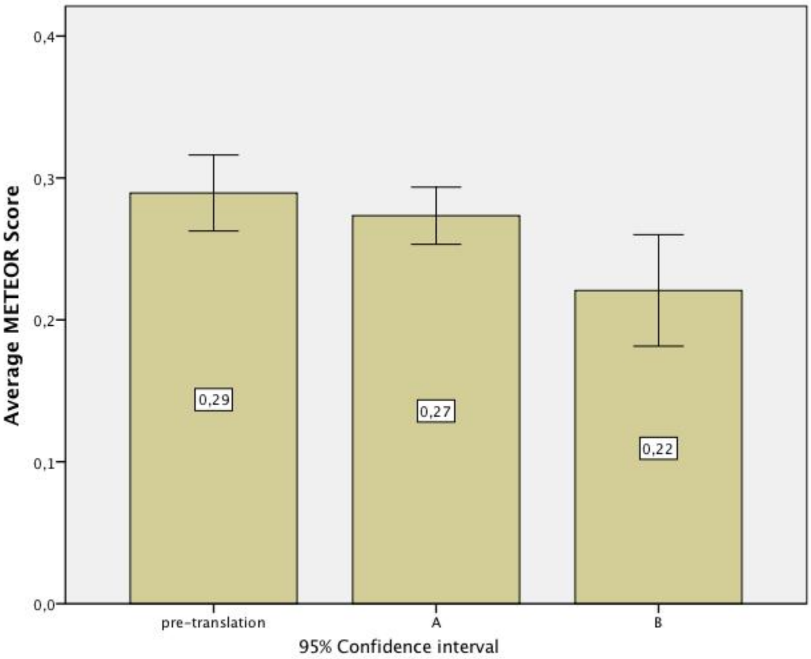


Figure 4.2: Average METEOR scores of the pre-translation, group A and B

Time analysis

Group B had to retype the whole sentence, while group A only corrected parts of it. As expected, the average working time of Group B (89 seconds) is higher than of group A (60 seconds). Figure A.3 shows the average time needed until the 'Submit' button was hit. One minute delay is unfortunately far from real-time. Further analysis of the recorded snapshots revealed that the workers show an initial delay of 22s (A), respectively 27s (B), before they start to type. Figure A.4 shows the comparison graphically. The real average handling time therefore is 38s (A) and 62s (B).

To conclude the first hypothesis, we can say that the average quality is lower if the workers have to retype the phrase. Not only is the quality of group A higher, also the working time is better. However, the diversity of group B is higher, what could boost the alignment efficiency.

4.3.2 Hypothesis 2

This section tests how the alignment methods performed in comparison to the pre-translation, and how the score evolves over time with rapid refinement. Due to the large complexity of the depth-first search, 12% of the test cases could not be resolved. In these cases, the trees end up with over ten million paths. We set the maximum number of paths for the traversal to 30'000. To preserve a reasonable comparison, these cases are skipped by all methods.

With the stored snapshots, a step-by-step simulation has been done. Figure 4.3 shows the progress of the METEOR scores with increasing work time for the proposed alignment algorithms. The abscissa is the time, measured in seconds from the start, and the ordinate is the measured METEOR score. The sampling rate used is 3 seconds. The simulation was run until five minutes elapsed. The figure only shows the first 66s because after one minute the evolution of the scores is not surprising anymore.

The scores surge in the first six seconds. *A**, *total weight* and *greedy best-first* climb higher, while the other methods stagnate. After about 30 seconds, all methods stabilize on their final average score. After the stabilization, the quality difference of each traversal algorithm is apparent. Only the *A** and *total weight* did catch up with the pre-translation of Google Translate (42%). The METEOR scores of all alignment methods after stabilization can be found in table 4.1 in the column 'unfiltered'.

The time analysis of the first hypothesis shows that workers' average wait (in the inspected group B) is 27s before they type the first word. Figure 4.3 reveals that stabilization is accomplished just shortly after 30s. This occurrence is unexpected. Further investigations brought out that 16% of the workers did 'cheat' and just copy-pasted the pre-translated sentence. Out of these cheaters, 98% did not modify the copy and committed the pre-translated sentence. This leads to high scores early, but was not in the intent of the authors. Figure 4.4 shows the scores, where the cheaters have been filtered out. With the applied filter, each sentence was improved by an average of 8.6 workers. Without the early copy-pasted proposals, the needed time until stabilization is raised. When the sentences are filtered, the maximum METEOR scores drop slightly. Table 4.1 shows the stabilized scores without cheaters in the column 'filtered'.

For an ongoing refinement of the output, the traversal algorithms have to be executed repeatedly. The alignment algorithm should be fast so that it can output results in as close as possible to real-time. In chapter 3, the complexity is already analyzed theoretically for worst cases. To learn the behavior in practice, each algorithm is run over all sentences recording the number of visited nodes. The average count of visited nodes for each algorithm can be found in table 4.2. The numbers are heavily dependent on the test set, but it serves as a comparison between the used methods. Graph traversal visits the minimum number of nodes. In contrast, all tree nodes have to be visited at

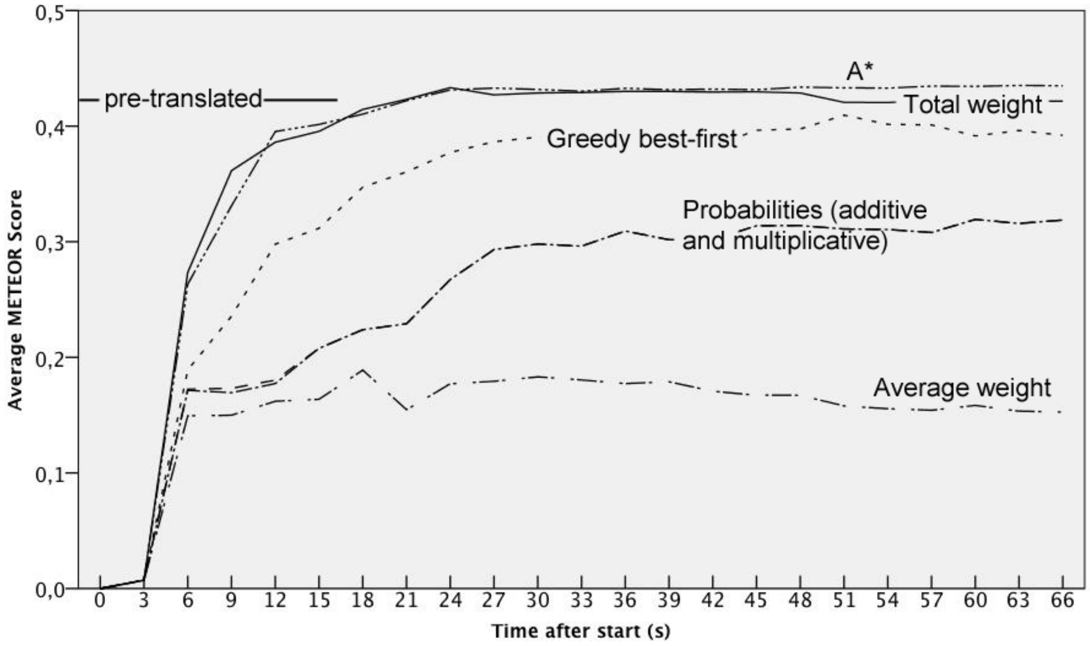


Figure 4.3: Progress of the score over time

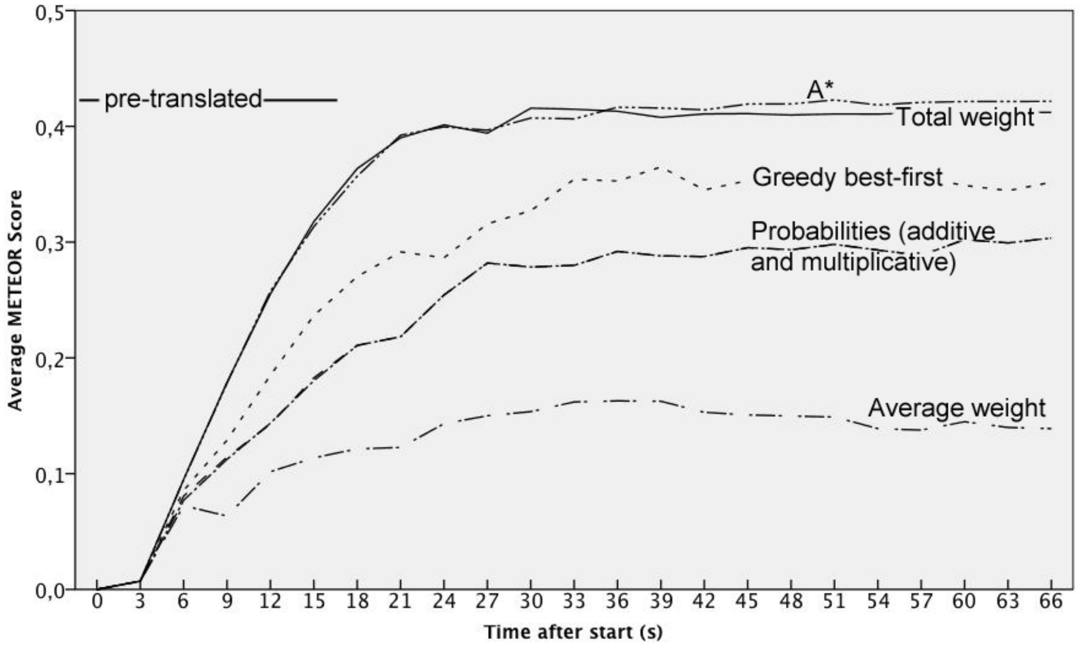


Figure 4.4: Progress of the score over time with cheaters filtered out

Alignment method	**unfiltered**	**filtered**
Total weight	0.423	0.421
Average weight	0.143	0.126
Probabilities (multiplicative)	0.338	0.323
Probabilities (additive)	0.338	0.323
Greedy best-first	0.397	0.358
A*	0.437	0.423
pre-translated	0.421	

Table 4.1: Final METEOR scores of the alignment method with and without filtering.

the depth-first search, independent of the methods used. The performance of A* is by far better than the best-first search and reaches almost the theoretical minimum. In regard to the fact that A* did output the highest METEOR scores and did converge the fastest, it is the best algorithm proposed in this thesis.

Traversal method	**∅ visited nodes**
Greedy best-first	12
Best-first	146890
A*	22

Table 4.2: Average number of visited nodes

The rapid refinement strategy can only be applied to the graph traversal and the A* search. Constructing a full tree is inefficient and a today's computer cannot handle the workload in real-time.

The second hypothesis can be corroborated. The minor enhancement of the score is not outstanding, but regarding the degradation of the average scores after editing the pre-translation, it shows the cardinality of sentence alignment. The A* search can again catch up with the pre-translation.

We proposed an *enhancement factor* for the weights in section 3.3.1. If the factor is larger than 1, edges that differ from the pre-translation are reinforced. The idea is to strengthen new words and word combinations. If the factor is less than 1, the alternating edges are devalorized and paths similar to the pre-translation are enhanced. Figure A.5 shows the METEOR scores over the whole test set for different enhancement factors. A reference line at the peak of factor 2 is drawn. Higher enhancement factors have a negative influence on the score. Factors below 1 show a more constant behavior. With smaller factor, the output resembles more the pre-translation. An enhancement score of 1 has been used in all other analyses, which means that no enhancement or devaluation has taken place.

4.3.3 Hypothesis 3

The offline analysis allows scoring the sentences against a reference translation. It reveals the real potential of a crowd-based translation. Two more alignment methods are introduced:

Best-worker: According to figure 4.2, the average quality after the improvement decreased. Nevertheless, some workers submitted better sentences than the pre-translation. Instead of merging the proposals, the output is simply the best proposal.

Oracle: The proposals are mapped to a tree, as explained in chapter 3. All paths are scored with METEOR against the reference. The oracle picks the best-scoring path as the output. The result is the best possible combination of the proposals and cannot be topped.

Since a reference translation must be available, these methods are only used to demonstrate the potential of translation with human computing. Figure 4.5 shows the average scores plotted together with the pre-translation and the A* traversal. The best workers scored almost 46%. The bar chart unveils that the best proposals (best-worker) can be further improved with the use of a combining method. With an *optimal alignment algorithm*, the output has a METEOR score of 48.7%. Again, not all sentences could be scored by the oracle because of their complexity.

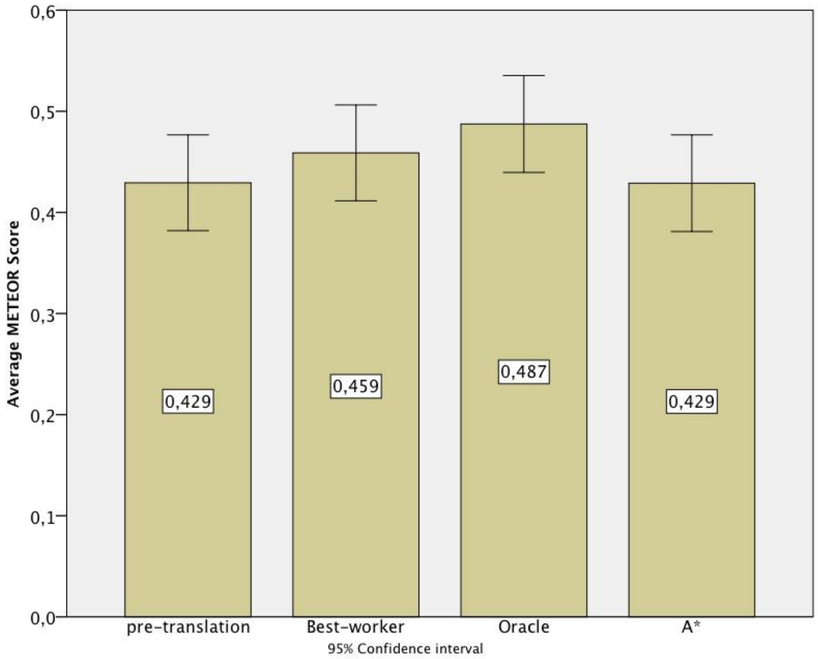


Figure 4.5: METEOR scores for the best-worker and the oracle

These results prove the third hypothesis. Combining multiple versions of a sentence can generate a better sentence. In this experiment, the quality gain is in the region of 3%, which is in the same range of the experience of [Rosti et al., 2007] with multi-engine machine translation.

4.4 Discussion, Limitations and Future Work

4.4.1 Hypothesis 1

The A/B-test points out that not only the alignment is important, but also the correct handling of the crowd. As [Bernstein et al., 2012] stated, *programming the global brain* is completely different from programming a computer. In the first hypothesis, the focus is not on the rapid refinement merge algorithm, but on the individual proposals of the crowd. The results disprove hypothesis 1 and show that the quality of the crowd-corrected sentences is lower, if the crowd workers have to retype the phrase. The crowd was not integrated well enough in the whole process. It turned out to be a weak link in the chain. Some workers did not understand the task, others did cheat deliberately. These factors have been underestimated in the design of the experiment. Many proposals were similar or equal to the pre-translation. Future work could be done to encourage the workers to use more imagination, creativity and activity to produce more-heterogeneous proposals of better quality. The task could be presented as a *game with a purpose* [Von Ahn and Dabbish, 2008], as a *multi-user game* [Bernstein et al., 2012], or to allow more interaction between the crowd-workers improving the same phrase. Another idea for better comprehension of the sentence is to also include the visual component of the film.

A limitation on the reliability of the result of hypothesis 1 is the small test set used. Only 20 sentences have been improved by both groups. A total of 200 sentences per group is not enough with respect to the fact that workers improved an average of seven phrases.

4.4.2 Hypothesis 2

We detected that the cheat-rate is exceptional high [Eickhoff and de Vries, 2012]. Workers in group B decided to copy-paste the pre-translation instead of retyping the sentence. This could have been avoided by converting the text to an image, not allowing the text to be copied. This mistake cost us 16% of unusable data. Instead of 10 workers per sentence, just over eight workers improved a pre-translation faithfully.

The results show how the proposed alignment algorithms performed in the real world. The final scores of the algorithms do not outperform the pre-translation. With cheaters filtered out, the scores did not go up, thus the bad results are not caused by the cheating workers. To test the hypotheses 2 and 3, group B has been used, which performed worse than group A. With low input quality, the score could only be raised back to the original value. However, by picking group B, a larger diversity among the proposals helps to generate a better sentence because the potential raises. The higher the diversity, the more difficult is the alignment.

All alignment methods are tested independently and some scored better than others. In the following, every method is discussed separately.

Average weight: The *average weight* method scores the worst because the output is generally too short. This search method often finds the shortest path. As mentioned in section 3.3.2, longer paths lead to a higher division of the workers, which would lower the average. Short paths contain too little information and are worse than too long paths.

Probabilities (additive and multiplicative): It is not surprising that these two methods score almost equal, because the quality prediction formulae are very similar. The outputs are long enough since they are maximizing the sums. The problem is the equal weight of the language and the reordering probability. The statistical n-gram values influence the sum too much. Results tend to use conventional words instead of the correct, rare words. The used data set are talks about specific topics, containing non-trivial words. The n-gram statistic from a search engine favors typical words and combinations. METEOR devalues this because the messages are not specific enough.

Total weight: In contrast to the other methods above, the *total weight* method performs better. The aligned sentences of are usually longer than the input and contain many needless or redundant words. METEOR does not punish too long sentences much because more words match with the reference. This method has a low *precision* but a high *recall*.

A* search: Unlike the output of the total weight method, the output of the A* search is always similar in length to the pre-translation. With 25% of confidence interval for the length of the A* output, longer sentences are also allowed. The precision of A* is higher than the precision of *total weight*, because the same information is stated in shorter phrases. However, more condensed phrases that are even shorter as the pre-translation, which may score well, are hardly considered. This limitation can be tackled by using a non-linear boost function $b(n)$ that is more tolerant towards shorter solutions.

Greedy best-first: The greedy best-first search is the only algorithm which does not require a tree. The final score is almost as high as the total weight algorithm, but it needs more time until convergence. The heuristics used might lead to wrong results if there are only few results to base the decision on. When more crowd workers start to type, the best path becomes more and more apparent. This alignment method shows the largest difference when the cheaters are filtered out. With only local decisions, the outcome of this method depends heavily on the consensus among the workers. Cheaters all submitted the same result and have a high consensus and thus a high influence on this algorithm. The output does not vary greatly from the pre-translation. With cheaters filtered out, the resemblance lowers.

There is a tradeoff between performance and the output quality. A faster algorithm uses stricter heuristics to be able to visit only the minimal number of nodes. The greedy best-first search makes local decisions only and has no back-off mechanism. The A* search visits almost twice as many nodes, but results in higher quality. Following this rule, all breath-first search methods should score the highest. They did not in this experiment, because the quality metrics of the paths are not good enough. The correct combination of averaging or summing of the weight, ϕ and p_{LM} is not yet found. Additional work has to be done to find better metrics. After finding this metric for the breath-first search, it can be mapped to the A* algorithm to reduce the search cost.

The proposed *enhancement* and *devaluation* weighting strategy of section 3.3.1 turned out to be a fine line. On one hand, the pre-translation often is a good phrase. When punishing these edges by devaluating them (enhancing the differing paths), other worse paths (maybe containing misspelled words or wrong sentences) are preferred. On the other hand, enhancing edges that are present in the pre-translation even reinforces the low diversity. The weighting factor can be tuned depending to the quality of the MT system and the input. If the pre-translation is of bad quality, drifting paths should be considered more. In the experiment, several factors have been tested. Factors below 1 reduce the chance for outstanding alternative phrases. If alternative paths are enhanced too much, good chunks of the pre-translation are avoided as well. This is the reason why the METEOR score drops with higher enhancement factors. Further analysis of this factor could be done to fine-tune the weights of the edges.

Good results emerge already after 30 to 40 seconds with rapid refinement. Late improvements do not preponderate much. The second hypothesis can be confirmed. All things considered, a delay of 30 seconds for a subtitle translation is reasonable. The duration of saying one sentence is typically in the range of 3-7 seconds in the talks used. This leads to the need for 5-10 worker groups for ongoing real-time translations. With 600 phrases per hour, having 10 workers improving a single phrase (0.05$/assignment), the costs for real-time translations amount to 300$/hour.

4.4.3 Hypothesis 3

With help of the oracle, the real potential of crowd-based translation is demonstrated. The maximum quality of the test set could be improved by 6%. A statistical significance is not given because figure 4.5 shows that the confidence intervals of the pre-translation and the oracle overlap strongly. The ranking of the pre-translation, the best-worker and the oracle is consistent. With a high chance, one of the ten workers is able to write a sentence of better quality than the pre-translation. If not, at least one cheater only copies and pastes the pre-translation. Thus, the score of *best-worker* has to be at least as high as the pre-translation. The oracle must score at least as good as the best worker does, because it could waive the combining and simply output the result of the best worker. By always taking the best parts of the proposals, an even better phrase can be output. However, an optimal alignment method is not found yet. This has two reasons. First, the proposals have a limited diversity. Most workers tried to improve the pre-translation instead of writing a complete new phrase with the same meaning. The orientation on the pre-translation limits the power of the crowd. Significant differing translations could lead to new interesting combinations that score better. Second, the proposed algorithms use wrong metrics or weight them unfavorably. Alternative metrics for quality predictions have to be found to compare the quality of two sentences. The breath-first search considers all paths but unfortunately prunes also the best-scoring results because of a wrong metric.

4.4.4 General discussion points

All scores are based on METEOR. Further appraisal with non-automated scoring has to be done since human evaluations are more trusted. The *adequacy* of a result is evaluated preferentially by humans. METEOR does not understand the content of the text and thus we cannot fully trust on this automated evaluation [Koehn, 2010, p. 222].

As the alignment process is the last component before the final output, it is not guaranteed to have a good solution. We have no control whether the merged sentence is correct and fits into the context or not. Additionally, it is not assured that words are used consistently over multiple sentences.

Last but not least, this thesis tested the concept only with German to English translations. The universal validity has to be shown by transferring the idea to other language pairs. The used talks are all about specific research areas. Other fields like movies or TV News could lead to different findings.

5

Conclusions

This thesis proposes a concept to translate a stream of sentences, as occurring in subtitles, with the help of a crowd. A MT engine serves as a pre-processor. With human computing and a sophisticated alignment method, the pre-translated sentence is improved. A monolingual synchronous crowd improves the text sentence-wise and redundantly, so that multiple proposals correspond to each pre-translation. Experiments show that the crowd-corrected proposals are worse than the pre-translation, but can be combined to generate a better phrase than its original.

The different versions of a phrase form a graph or a tree, where each word is a node. The graph and the tree are searched for the path forming the best sentence. Among multiple combining algorithms, the A* search turned out to be the best. With an A* search, it is possible to traverse the tree efficiently to find a good recombination. The rapid refinement merge algorithm already detects an agreement among the workers after 30 to 40 seconds of working time.

Real-time translations are not limited to subtitles. Language acts as a barrier in other situations of our daily lives, too. When it comes to natural language processing, we do not place our trust in machine translations, but in human brainpower. For example, online support chat rooms could use real-time crowd-based translation as well. Up to now, international companies have to hire and educate specialists in multiple languages, which is very costly. Instead, these companies could use the idea proposed in this thesis.

We believe that the coupling of MT and crowd-sourcing has a high potential. With a well-designed task for the crowd, the potential can be much higher than the 6% of quality gain we showed in the experiment. Additionally, compared to professional translations, the expenses are reduced, as a monolingual crowd is cheap and professional translators would no longer be needed. The high availability of the crowd saves time and reduces delays. A synchronous crowd could become a valuable resource for tasks that cannot be solved by potent machines and require the smallest possible latencies.

References

[Barzilay and Lee, 2002] Barzilay, R. and Lee, L. (2002). Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 164–171. Association for Computational Linguistics.

[Bernstein et al., 2012] Bernstein, A., Klein, M., and Malone, T. (2012). Programming the global brain. *Communications of the ACM*, 55(5):41–43.

[Bernstein et al., 2011] Bernstein, M., Brandt, J., Miller, R., and Karger, D. (2011). Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 33–42. ACM.

[Eickhoff and de Vries, 2012] Eickhoff, C. and de Vries, A. (2012). Increasing cheat robustness of crowdsourcing tasks. *Information Retrieval*, pages 1–17.

[Jayaraman and Lavie, 2005] Jayaraman, S. and Lavie, A. (2005). Multi-engine machine translation guided by explicit word matching. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 101–104. Association for Computational Linguistics.

[Koehn, 2010] Koehn, P. (2010). *Statistical machine translation*, volume 11. Cambridge University Press.

[Lasecki et al., 2012] Lasecki, W., Miller, C., Sadilek, A., Abumoussa, A., Borrello, D., Kushalnagar, R., and Bigham, J. (2012). Real-time captioning by groups of non-experts. UIST.

[Lavie and Denkowski, 2009] Lavie, A. and Denkowski, M. (2009). The meteor metric for automatic evaluation of machine translation. *Machine translation*, 23(2):105–115.

[Liem et al., 2011] Liem, B., Zhang, H., and Chen, Y. (2011). An iterative dual pathway structure for speech-to-text transcription. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

[Malone et al., 2010] Malone, T., Laubacher, R., and Dellarocas, C. (2010). The collective intelligence genome. *IEEE Engineering Management Review*, 38(3):38.

[Minder and Bernstein, 2012] Minder, P. and Bernstein, A. (2012). How to translate a book within an hour: towards general purpose programmable human computers with crowdlang. In *Proceedings of the 3rd Annual ACM Web Science Conference*, pages 209–212. ACM.

[Moore et al., 1998] Moore, G. et al. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85.

[Pang et al., 2003] Pang, B., Knight, K., and Marcu, D. (2003). Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 102–109. Association for Computational Linguistics.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

[Rosti et al., 2007] Rosti, A., Ayan, N., Xiang, B., Matsoukas, S., Schwartz, R., and Dorr, B. (2007). Combining outputs from multiple machine translation systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 228–235.

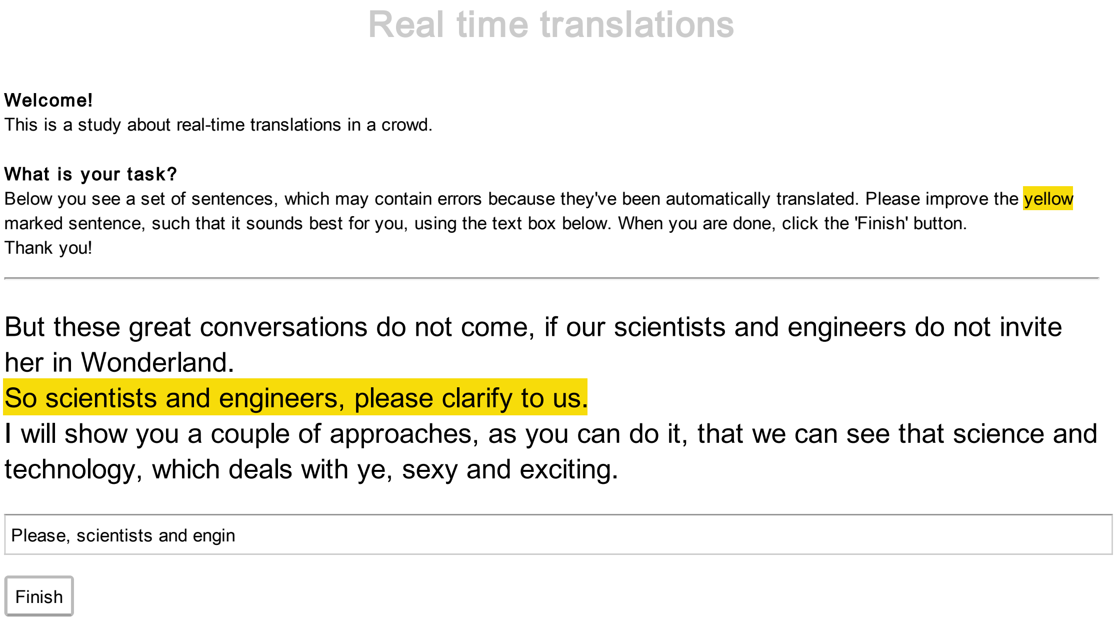
[Russell et al., 1995] Russell, S., Norvig, P., Canny, J., Malik, J., and Edwards, D. (1995). *Artificial intelligence: a modern approach*, volume 2. Prentice hall Englewood Cliffs, NJ.

[Von Ahn and Dabbish, 2008] Von Ahn, L. and Dabbish, L. (2008). Designing games with a purpose. *Communications of the ACM*, 51(8):58–67.

A

Appendix

A.1 Figures



Real time translations

Welcome!
This is a study about real-time translations in a crowd.

What is your task?
Below you see a set of sentences, which may contain errors because they've been automatically translated. Please improve the yellow marked sentence, such that it sounds best for you, using the text box below. When you are done, click the 'Finish' button.
Thank you!

But these great conversations do not come, if our scientists and engineers do not invite her in Wonderland.
So scientists and engineers, please clarify to us.
I will show you a couple of approaches, as you can do it, that we can see that science and technology, which deals with ye, sexy and exciting.

Please, scientists and engin

Figure A.1: The user interface for the experiment

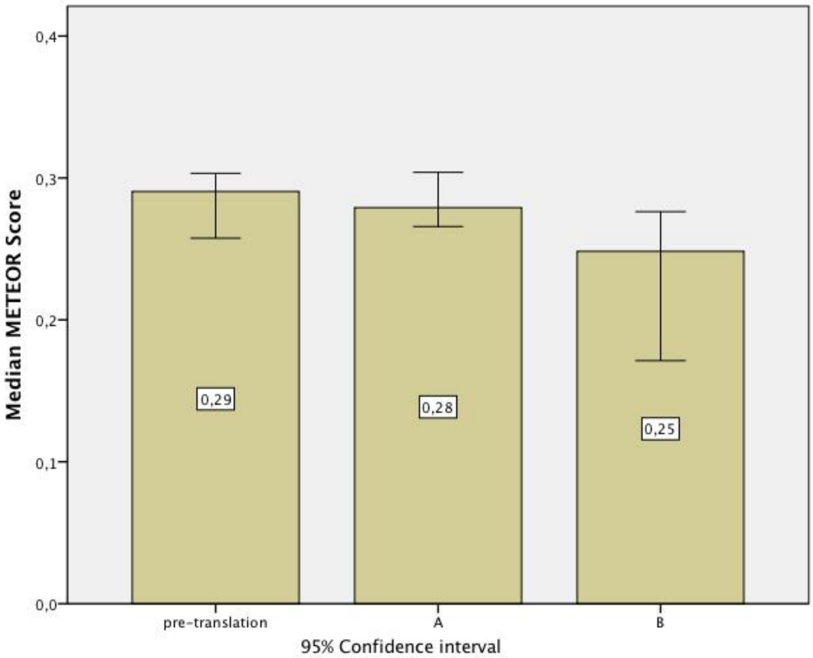


Figure A.2: Median Meteor scores of the pre-translation, group A and B

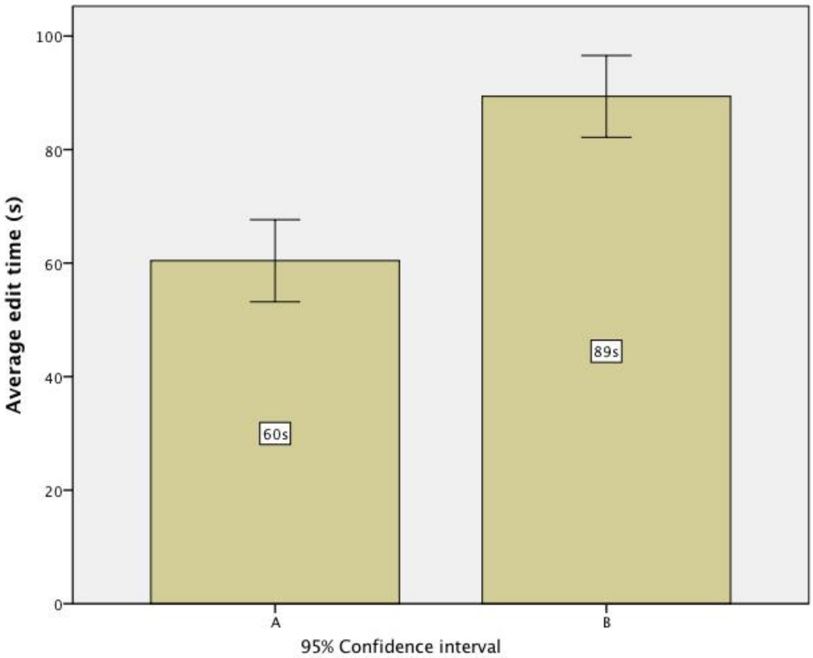


Figure A.3: Working time of group A and B submitting the result (incl. waiting time)

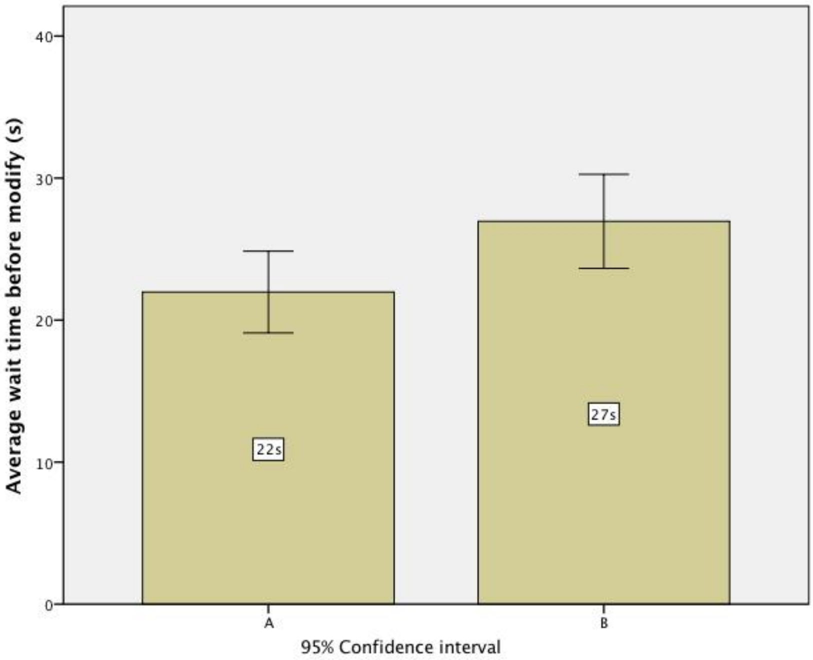


Figure A.4: Average waiting time until the first modification is done

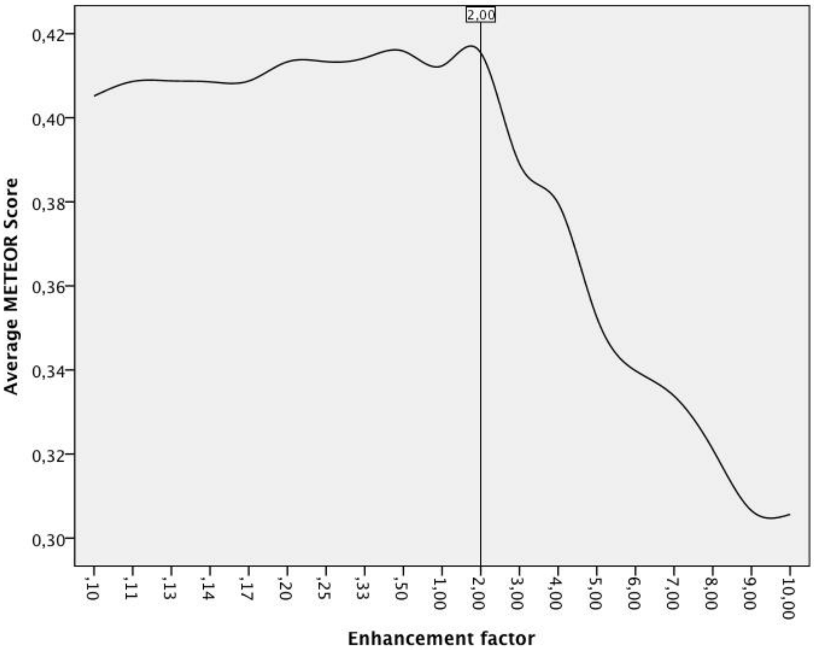


Figure A.5: Evaluation of different enhancement factors for edge weights

List of Figures

List of Tables