Online Incremental Machine Translation

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Kay Rottmann
aus Halle in Westfalen

Tag der mündlichen Prüfung:	19. Juni 2015
Erster Gutachter:	Prof. Dr. Alexander Waibel
Zweiter Gutachter:	Prof. Dr. Tamim Asfour
Dritter Gutachter:	Prof. Dr. Joy Ying Zhang

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe, sowie dass ich die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des KIT, ehem. Universität Karlsruhe (TH), zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Stuttgart, 18. März 2015

. . . . . .

(Kay Rottmann)

Abstract

The main idea behind the statistical machine translation (SMT) is to use a big bilingual training corpus to learn statistics which put two languages into relation. Such a training corpus however is never able to cover all possible scenarios where the trained system will be used in. Especially in cases where the system is going to be used for translation of different domains.

This becomes a problem when applying a machine translation system in the context of presentations like they are done at universities. For supporting classes with speech translation you cannot expect that you had training examples for every topic you will encounter in these classes. Especially not with regard to advanced classes which use rich special purpose vocabularies and technical terms. On the other hand however it is reasonable to assume, that during a single lecture the content won't change too much and the domain will probably be the same throughout the whole presentation.

Investigating the use case of a lecture translation system it also becomes obvious, that such a system needs to be designed in a distributed architecture to handle lectures in different lecture halls, and it also has to be dynamic enough to deal with changing audiences across lectures or even during a single lecture. Looking at the problem of domain mismatch, it is also desirable, that such a system can improve and use crowd sourced improvements immediately in presentations held in another location. This becomes obvious thinking of an introductory robotics seminar being held in one location and an advanced robotics lecture in another. It is desirable, that improvements to the translation quality made in the seminar will be used in the advanced lecture as well.

This thesis investigated the problem how to support this situation. In a first step, we developed the algorithms and techniques which allowed a fast modification of the model

parameters to correct translation errors and prevent them from happening in the future again allowing individual customization of machine translation. This enabled us to improve the overall translation quality, but it also allowed us to improve the quality of translations for the rest of a lecture covering the same domain.

In a second step we researched the possibility of making speech translation available in a distributed environment and developed a distributed architecture which allowed crowd sourcing feedback which immediately was usable to improve the underlying translation systems using above mentioned techniques. We took great care in making this system dynamic to handle different at runtime changing requirements, like a changing audience in a lecture with changing language needs, but we also focused on making the system robust to be able to deal with failing components, in the best case without end users noticing any problem.

The developed architecture was put into regular use in 2012 at the Karlsruhe Institute of Technology and its use is constantly extended since then. Since we dealt with crowd sourcing it was also to expect that some people try to sabotage the system by providing bad translations, for that reason we also investigated the identification of bad translations and a user accounting was developed which allowed the identification and even the exclusion of people trying to sabotage the system.

We also asked and evaluated the question how to support users who are willing to provide better translations by the use of automatic machine translation for them.

In this thesis we present extensive experimental results performed on TED lectures, which show that our developed algorithms were able to improve a system in a fast way, without introducing delays for the end user. We also showed that the translation quality improved in ways beyond the simple addition of previously unknown words to the translation dictionary, and that even improvements to the overall translation quality for a single lecture were possible. We proposed a training schema which would allow the new developed setup to be run as a hybrid model with continuous improvements and rare complete retraining to achieve the best overall translation quality over time.

Zusammenfassung

In der statistischen maschinellen Übersetzung (SMT), ist einer der Hauptansätze, dass aus einem zu Grunde liegenden Trainingskorpus Statistiken gelernt werden. Ein solcher Trainingskorpus kann jedoch nie alle späteren Anwendungsfälle abdecken, insbesondere wenn nach dem Training das System bei der Übersetzung unterschiedlicher Domänen helfen soll. Dieses ist insbesondere ein Problem wenn man an die Anwendung von maschinellen Übersetzern als Unterstützung von Präsentationen wie zum Beispiel bei Vorlesungen an Universitäten denkt. Bei einer generellen Unterstützung von Vorlesungen ist damit zu rechnen, dass nicht jede Vorlesungsdomäne der an der Universität gehaltenen Vorträge abgedeckt ist, jedoch im Verlauf einer einzelnen Vorlesung die Domäne selbst nicht groß wechselt. Wenn man den Anwendungsfall eines Vorlesungsübersetzers weiter betrachtet, so fällt auch auf, dass an einer Universität mehrere Vorlesungen an unterschiedlichen Orten gleichzeitig gehalten werden. Auch das Publikum in diesen Vorlesungen ist nicht immer das selbe. Dennoch gilt auch hier, dass es häufig vorkommt, dass in einem Seminar zu Robotik ähnliche Korrekturen am Übersetzungssystem nötig sind, wie in einer parallel laufenden Vorlesung zu Robotik. Um eine solche Umgebung optimal zu unterstützen, ist es nötig, dass auf der einen Seite eine maschinelle Übersetzung schnell angepasst werden kann, so dass Übersetzungsfehler am Anfang einer Vorlesung korrigiert werden können damit sie im weiteren Verlauf nicht mehr auftreten. Auf der anderen Seite muss ein Vorlesungsübersetzer aber auch flexibel genug sein, um sich an die wechselnden (Sprach-) Bedürfnisse anzupassen und auch Korrekturen die in einer Vorlesung gemacht wurden in einer anderen Vorlesung direkt zu benutzen.

Im Rahmen dieser Arbeit wurden in Hinblick darauf zunächst Algorithmen und Methoden entwickelt, die eine schnelle Anpassung der Modellparameter eines statistischen Übersetzungssystems nach einem initialen Training erlauben. Damit können zusätzliche Trainingsdaten die erst mit oder während der Benutzung des Systems entstehen, für zukünftige Übersetzungen benutzt werden. Nach einer Einleitung im ersten Kapitel und der Bildung der mathematischen Grundlagen die für diese Entwicklungen notwendig sind in Kapitel 2, beschreibt das vierte Kapitel die in dieser Arbeit entwickelten Methoden zur Veränderung der Modellparameter auf Basis von Korrekturen die zur Laufzeit gemacht werden können. Um diese Veränderungen zur Laufzeit vornehmen zu können, haben wir neue Modelle für die automatische Übersetzung entwickelt, die die nötigen Informationen zur Änderung der bestehenden Modellparameter bereithalten. Ferner greifen wir auf Heuristiken zurück die eine effiziente Berechnung von Wort-Alignments zur Laufzeit ermöglichen.

Um diese Entwickelten Methoden Nutzbar zu machen, haben wir ferner eine verteilte Architektur für Sprachübersetzungssysteme entwickelt, die mittels Crowd-Sourcing von Studenten gemachte Verbesserungen direkt an die Komponenten weiterleitet und die in Kapitel vier beschriebenen Berechnungen anstößt.
Um den zukünftigen Einsatz als Vorlesungsübersetzer zu ermöglichen, entwickelten wir die Architektur so, dass Anforderungen an die benötigten Sprachen dynamisch auch zur Laufzeit einer Vorlesung geändert werden konnten. Dabei legten wir besonderen Wert auf die Fähigkeit des Systems automatisch die beste Verarbeitung für die gewünschte Ausgabe zu finden, und auch im Falle einer fehlerhaften Komponente war unser Anspruch das System so robust zu gestalten, dass diese Komponente nach Möglichkeit für Endanwender unsichtbar ersetzt wird.
Die in dieser Arbeit entwickelte verteilte Struktur des Systems erlaubte dann auch den parallelen Einsatz eines automatischen Vorlesungsübersetzungssystems an verschiedenen Orten - sprich in verschiedenen Vorlesungssälen - und wurde 2012 am Karlsruher Institut für Technologie in Betrieb genommen. Die dafür entwickelte Architektur ist in Kapitel drei beschrieben und die in 2012/13 benutzte Installation in Kapitel 6.
Da im Crowd-Sourcing auch immer wieder mit Saboteuren zu rechnen ist, haben wir zusätzlich Maßnahmen untersucht um schlechte Übersetzungen automatisch zu identifizieren und vor der automatischen Verbesserung herauszufiltern um sie automatisch auszuschließen. Eine in die verteilte Architektur integrierte Benutzerverwaltung ermöglichte damit auch die Identifikation von Personen die dauerhaft schlechte Übersetzungen generieren und erlaubte damit fehlerhafte oder schlechte Korrekturen von weiteren Verbesserungen zu entfernen.

Um den Nutzer bei der Erzeugung verbesserter Übersetzungen zu unterstützen haben wir auch Methoden betrachtet wie ein automatisches Übersetzungssystem den Aufwand bei der Erzeugung von Übersetzungen verringern kann. Die dazu entwickelten Methoden und die integration in den Verbesserungsvorgang ist zusammen mit den anderen Algorithmen in Kapitel 4 beschrieben.

In Kapitel 5 haben wir in der Arbeit ausführlich am Beispiel von Transkripten zu TED Vorträgen - welche mittels Crowd-Sourcing erstellt wurden - gezeigt, dass die von uns entwickelten Methoden geeignet sind ein System zur Laufzeit versteckt vom Benutzer zu verbessern. In den Experimenten gehen wir detailliert auf die Geschwindigkeitsvorteile unserer Methoden ein und untersuchen eingehend die wie siche die Qualität über eine längere Benutzung hinweg verbessert. Wir zeigen auch, dass die Verbesserungen über das einfache Hinzufügen von Wörtern zum Übersetzungsvokabular hinaus gehen.

In den Anhängen beschreiben wir schliesslich die von uns für die verteilte Architektur entwickelten Protokolle, die es Drittanbietern ermöglichen für solch ein System relevante Komponenten einfach zu entwickeln und anzuschließen.

Für

Christina & Zwerg

Acknowledgement

This thesis could not have been done without the help of the many people who supported me during the time I was working on this. I would like to express my special appreciation to my first advisor Professor Dr. Alex Waibel for giving me the opportunity to work on this thesis. His mentoring allowed me to grow as a research scientist and to develop my career.

Furthermore I would like to express my gratitude to Professor Dr. Tamim Asfour and Professor Dr. Joy Zhang for their valuable feedback and discussions and their willingness to help me with this thesis.

I would also like to thank the whole *Mobile Technologies* team, especially: Christian Fügen, Thilo Köhler, Matthias Eck, Maximilian Warsewa, Benjamin Hujer, Tina Milo, Mirjam Mäß and Tim Notari. It was an invaluable experience to see research going into a product used by people around the world coming a step closer to tearing down the language barrier.

A source of always new ideas and valuable input was the team of the *Interactive Systems Laboratories* at the Karlsruhe Institute for Technology. I want to thank: Jan Niehues, Teresa Herrman, Eunah Cho, Thanh-Le Ha, Mohammed Mediani, Yuqi Zhang for the fruitful discussions around machine translation, as well as: Kevin Kilgour, Bastian Krüger, Klaus Joas, Christian Mohr, Markus Müller and Sarah Fünfer. Without them the deployment of the lecture translator would not have been such a success.

My final and greatest thanks go to my whole family for their support and believe in me, and I am especially grateful to my wife. There are no words to convey how much I thank you for all your support and love. Without you, this thesis would not exist.

Contents

1. Introduction

Machine translation remains one of the greatest challenges for todays artificial intelligence systems. The need for accurate translations of text is ever growing and with every passing day the number of available documents as voice transcripts, books, articles or interactions in social media which require a translation is increasing.

Despite of the usefulness of the information, often these documents are only available in just one, rarely in a few more languages. This prevents people not speaking these languages to benefit from the knowledge of those documents. Usually, only when the need is big enough for translation, professional translators are hired for translating these documents then, but this is a very expensive procedure and impossible for every document in the world, restricting this approach only to those documents, where the payoff is big enough to justify the investment.

One can take university lectures as an example. With the rise of the internet, more and more lectures held at universities or at public conferences are made available to the public as video recordings. The rise of conferences like TED(TED13) or the growth of so called massive open online courses (MOOCs) are a good example for the trend of providing lectures to an audience with diverse language background.

In the past couple of years, crowdsourcing human translations has proven to be one possible solution for this problem in the lecture setting. The world wide held TED conferences are followed by thousands of people providing transcripts and translations shortly after the publication. But still, even with a large background community doing translations for free, there is always the delay until a whole presentation or document has been translated by a human, and it can't be guaranteed how fast a translation will be available.

A new way to make this content accessible to the broader public is to use machine translation to provide translations for those who do not understand the original content. One of the most promising approaches among the automatic translation is the so called statistical machine translation (*SMT*), which has been shown to be able to provide useful translations for large amounts of documents in a fast and cost effective way.

In this thesis we discuss how to combine crowd sourcing of corrected translations and statistical machine translation, to provide better translation systems, which show fast translation quality gains from user feedback in a distributed lecture translation environment without the need for long training runs. We also developed the techniques needed to make improvements in the translation quality available to different translation engines at once and it also showed that improvements can be partially applied to language pairs, other than those for which user feedback was collected.

1.1. Statistical Machine Translation and Data

Statistical machine translation is based on the main principle to extract statistics over the language usage from large monolingual and bilingual texts. While monolingual data is usually available in large amounts, even this large amount used for training does not cover every possible domain which might be needed in the future.

For bilingual data, the problem is much worse. The problem with bilingual data is, that it is most of the time even not available at all for specific domains. And for some language pairs, the total amount of bilingual data is so small that every little bit of not in domain data would be tremendously helpful.

Therefore it is very important to collect this data to continuously improve machine translation systems. For that reason, there are different projects that created with a lot of effort collections of parallel data. To name a few, there are the translations of sessions in the european parliament (Koe05), manually constructed corpora (Tat12) or collections of data like subtitles, documents by the UN or other freely available data (Tie12). But still, *SMT* is facing problems when translations are needed for topics that were not covered in the original training data.

This all shows the importance for *SMT* systems to continuously grow the size of the bilingual training data and add this additional data to the system for continuos translation quality improvement.

1.2. Update Cycle for Statistical Machine Translation

Solving the obvious need for new training data is however just one side of the problem. Even in an ideal world, where new training data was constantly generated, the underlying models for the machine translation system need to be updated as well.

This retraining of a statistical machine translation system is a very time consuming process. The standard batch processing approach for training statistical machine translation systems can easily take multiple hours and even days to finish, depending on the overall available training data amount. The reason for this very long training time is the fact, that the statistics used by the system are computed over all available data and the model parameters are optimized to maximize the translation quality.

One could argue, that with the continuously improving speed of computer hardware, the training will eventually be fast enough to deal with this problem. However the always increasing number of training samples also comes in hand with an increase in training time. In addition to that, even if the increase in processing speed outperforms the increase in training time, so that the overall time used for training is becoming smaller in the future, it is unlikely that a full retraining and re-estimation of all model parameters is possible within for example two consecutive requests to a machine translation engine, which are often only a few milliseconds apart.

Therefore it is important to find ways how to update the model parameters of a machine translation system in a fast and efficient way.

1.3. Difference Between Training and Real World Usage

Even when assuming that the previously mentioned two problems are solvable - meaning it is possible to continuously find additional training data for improving statistical machine translation systems and there are mechanism, which immediately improve the machine translation models based on this additional data the moment the data is available - another problem still exists.

The problem is the actual mismatch between training data and the translation requests faced in the real world application of the machine translation system.

As mentioned before, it is very important to have training data reflecting the use of language which will be seen during real world usage. Otherwise translations suffer from bad word usage in translations, or even so called unknown words (words which have not

been seen before in training and are therefore not translatable) because of the difference in statistics caused by different word usage between the translation requests and the initial training data.

One approach to overcome this problem to some extend is to generate artificial examples - sentences, which might show up in the future real world application - and let those be translated by professional translators. The problem with this approach however is that there is still the mismatch between requests and this additional data, because the artificial generated data is based on assumptions and unable to cover the real distribution and variations one might see in the different documents.

Especially when it comes to the translation of university lectures, this is a very big problem, because these lectures are usually dealing with topics not covered by the available data, forcing the machine translation to translate sentences only with its background knowledge from other domains. It is obvious, that the translation of a math lecture dealing with advanced calculus will not be covered well by language learned from sessions of the european parliament.

This adds an additional layer to the previously mentioned problems for improving a statistical machine translation system, which is to find or generate training data that is useful for the application of the translation system.

1.4. User Feedback

A solution to build additional training data which is similar to the real usage of the translation system is to have the inputs to such a translation system manually translated afterwards. This would ensure that past utterances which resulted in a bad translation will generate better translation results in the future. One approach for doing this is to have an external work force of professional translators who are paid for the task of translating and correcting past translation requests.

This approach is usually done in an offline manner and the task can be sped up by techniques which select those sentences which will help most in improving the translation quality, as described in (ES08) or related to that in (Amb11). This allows a cost reduction and a much faster improvement of the translation systems, by focusing on those sentences with the most impact.

However the problem with this approach is still that the integration of external human translators introduces a time overhead of generating the translations and then later including the data into the models again. In addition to the time overhead the earlier discussed

costs also play a role in this case, because professional translators are expensive. Putting the costs aside, the users who see problems in translations won't benefit from the professionally generated translations because of the introduced time delay, which also means, the user is unable to get the immediate feeling of an improving system which might affect the adoption of machine translation systems and the willingness to provide translation corrections.

As an alternative to the costly generation of correct translations, crowd sourcing techniques at the moment of the generation of the content can be seen as an alternative. Here users who observed a problem in a translation can directly give feedback on how a better translation would have looked like. This in combination with a fast improvement of a machine translation system, where new training data is immediately integrated into the models would leave the users with an improved system and an immediate gratification in the form of better translations of similar sentences the user just corrected. The user would be able to see, that improvements he made are directly resulting in improved translations in the future, where in the optimal case previously made translation errors won't happen again.

In a large scale system this goes even further and ends in an environment where multiple users use a distributed system and user A corrects a translation, while user B uses the same system at a point in time after A gave feedback for a different sentence which benefits from user A's correction.

This thesis developed the techniques and algorithms, to make this task possible.

1.4.1. Harmful User Feedback

When crowdsourcing translations or even when using professional translators for the generation of translations, a problem that often occurs is the mixed quality of feedback. Sometimes the manually generated translations are of very low quality, and sometimes the translations are also completely wrong. There are multiple reasons for that.

While in some cases translators are simply not able to translate sentences correct due to lack of knowledge - this is especially the case with crowd sourced translations - there are other cases, where the translators also translated the wrong sentence.

But in addition to this, there is also the problem of bad translations caused by users who try to sabotage the system. This might be for "fun", or other unknown reasons, but it is hard to prevent this in the case of crowd sourcing.

For that reason it is an important task to identify good training examples while removing the bad training examples to improve the overall machine translation quality.

1.4.2. Supporting Users in Machine Translation Tasks

Another part in the crowd sourcing for translations is also to make it as easy as possible for the users to provide feedback.

By supporting users the moment they provide feedback, they will be faster, and in addition to that they also notice, that their work is valued. This in return will make them more willing to provide feedback and show higher retention and adoption of the crowd sourcing process as a whole.

1.5. Distributed Architecture for Learning Speech Translation Systems

While the previous sections concentrated alone on the question of how to improve the machine translation system and how crowd sourcing could be a part of this process, the crowd sourcing approach in a lecture translation environment also opens up another side to look at.

The crowd sourcing in university lectures goes far beyond the updating of models for machine translation systems, but it also involves the question, how this service can be developed, so that it delivers a good experience to all participating parties in such a system.

In a university setting, there are multiple lectures at the same time, with sometimes hundreds of students listening to a presentation.

To allow crowdsourcing in such a setting, it is important that the underlying service architecture is developed to be flexible enough to support multiple translation requests at the same time, provides mechanisms for crowdsourced feedback and is also able to serve multiple students in different lecture halls at the same time. In this setting, the optimal case would be, that a student who corrects a translation error in one lecture will cause a better translation for some other student in another lecture later. This other lecture can even be held in a different language compared to the lecture the correction was made in.

But in addition to the live translation system, the more likely case where students invest time with translations and are therefore willing to provide corrected translations, is when

they are able to review past lectures during their studying for exams. In this case it is even more helpful for students to see other students corrections, so that the group of people who review a lecture all benefit from other peoples corrections. Such a framework will result in increased willingness to correct translation errors and the overall quality of the translations. This will benefit the students on the one hand and on the other hand at the same time generate valuable training data for the statistical machine translation system.

1.6. Goal of this Thesis

The previously made observations and discussion of problems for the improvement of statistical machine translation systems established the objective of this thesis.

The goal was to develop algorithms and techniques for the full automatic, fast improvement of statistical machine translation systems based on crowd sourced user feedback in a distributed speech translation environment.

This goal was split into multiple different steps that needed to be solved. The first step was to develop the algorithms and techniques to compute model updates which allowed quick modifications of the underlying translation models resulting in an improvement in translation quality. This included the estimation of better language and translation model statistics and also improved lexical and alignment probabilities for the incorporation of additional crowd sourced data.

The next step was to find methods to identify potential bad translations or vandalism and to support users when they are willing to provide corrected translations.

All these techniques then needed to be integrated into a new framework, which was able to deal with the aforementioned requirements of a lecture translation system used at a university. This involved the development of techniques which were able to address the dynamic requirements within such a system and the question of how to present this to students and how professors can use such a system.

The next step was then to evaluate the proposed methods and algorithms to show that the assumptions made are correct. Because of lack of example data, the goal was to evaluate on the publicly available TED data, which consists of crowd sourced public talks, similar to what can be seen in university lectures.

The final step then was to deploy the developed system as a running lecture translator at the Karlsruhe Institute of Technology.

1.7. Related Work

The research in improving systems directly on user feedback has drawn more and more attention over the last couple of years. Most notable is the work by (CFNV08) and (OMGVC10). They explored the usage of *interactive machine translation*, where human translators are aided by machine translation systems and these systems were improved by the feedback of the human translators. In a very similar line is (CBF+14) and (BSC+14)

who were also looking into the process of updating models for usage by professional translators for post editing translations.

In contrast to that work, this work focuses on the usage of a statistical machine translation system by students on speech data instead of professional translators in a computer aided text translation system situation. In our case we also expected that sometimes there will be feedback even by users who are not quite able to speak the original source language. That means in contrast to (OMGVC10) and (CBF$^+$14) we focused on the question how to improve a statistical machine translation system without experts at hand, namely how to crowd source this information in a distributed environment set up in the lecture translation setting at a university where students might even provide bad translations, while (OMGVC10) looked at the question how professional translators benefit from feedback to machine translation systems in their aided translation tasks.

In addition to the automatic and fast improvement of machine translation systems, another very active topic in current research is also the question, how to crowd source this data at all and if all data is equally important. The approaches taken in (Amb11) and (ES08) focus on the selection of an optimal set of sentences for the creation of a machine translation system with minimal cost. In their work, they show that with this selection of a subset of the original data for creating a bilingual corpus, similar translation performance can be achieved at lower cost for generating the bilingual data used for training. This especially allows the creation of machine translation systems for languages, where only limited or no translated data is available, and therefore additional data needs to be created. Another work going into this direction is (ZCB11), which focused on the question how to crowd source the correct translations from non professionals using technologies like amazon mechanical turk (KCS08) or similar ones, or even using crowd sourced evaluation (CB09). We also experimented with the collection of crowd sourced speech data in (LWER10).

This thesis distinguishes itself from that line of work, by going a step beyond, after the creation of such an initial translation system, investigating the question how such a system can be incrementally improved based on the real world usage of such a system. This results in a machine translation system improving continuously over time, without ”forcing” users to translate sentences they are not interested in, but learning from whatever is corrected by users. The previously mentioned approaches focus on the question how to select specific sentences from background corpora which should be translated by users, without regard to the real world use case of the translation system. The work proposed in this thesis however

provides more targeted optimization by focusing on the improvement of translation errors which happened during the real usage of the system, which is especially for a university lecture environment a favorable thing, because similar lectures happen to occur every year. This would allow users (professors and students) to expect improved translation quality year over year.

Also the work on how distributed machine translation systems can be developed and managed was the focus of attention for some time now. Already in 2001, (FWS+01) showed an architecture of a speech to speech translation system where speech recognition, translation and speech synthesis were combined in a modular fashion. This system was the basis for future extensions, which resulted in the first lecture translation system (FWK07), which however still was a static architecture for fixed language pairs. In (SPF+11) the original architecture from (FWS+01) was extended to build a translation system for multiple different Asian languages. However to our knowledge, still there was no architecture developed that allowed the highly dynamic requirements needed in a lecture translation system.

This thesis extends over those previous systems by developing a framework for general speech technology needs in a distributed environment. We allowed for multiple parallel streams with dynamic language processing demands at runtime. The developed framework is able to cope with failing components by falling back to similar components or replacing them with different paths which lead to the same results. To our knowledge this was the first such framework developed for the purposes of real time speech to speech translation.

The system was developed mainly for the task of speech translation, however the framework is general enough to be applied to other tasks as well. In one installment we showed that it was also able to handle transmission of slides in a lecture environment, allowing even optical character recognition and translation of slides as an additional modality of in- and output.

Another important aspect in machine translation daling with crowd sourced data is the identification of noise and removal of bad translation examples which might hurt translation quality, significant work has been done in the past years. In (MM05) it was shown, that algorithms capable of the identification and removal of noisy data in the training corpora for machine translation lead to improved translation quality. (ZCB11) focused on a preselection of input sentences for better training data, which is among others the main difference to our research on this aspect. We focused on the real time identification of reasonable additional training data to decide whether to learn from a sentence pair or not. In (HMNW11) a comparison of different mechanisms to identify noise in parallel training

data is given and the effects in improved translation quality are shown. We built upon the findings of that work and developed mechanisms to directly apply a noise identification method to the feedback provided by the user.

Finally with improving statistical machine translation systems, it also became possible to support professional translators at the task of translating documents. With (ABC+14) a whole workbench was developed to support professional translators. This workbench built upon findings in (AOMSC10) for the support of translators in interactive machine translation, which also looked at the task of online learning, however for the task of interactive machine translation alone (OMGVC10).

1.8. Outline of the Thesis

The structure of this thesis is as follows. Chapter 1 provides an overview of current difficulties in building and training statistical machine translation system. We especially point out the problems of slow model updates and the difficulty to obtain good training data. In addition to that we also look at the situation of a lecture translation system in a university environment and the resulting implications and requirements for such.

Chapter 2 gives an overview of the underlying theories and their implications of statistical machine translation. We explain the typical models used in todays state of the art machine translation systems and how the search, the so called decoding, for the best translation hypothesis operates. Because the evaluation plays an important role in the development and improvement of statistical machine translation, we also give details on the internals of the most famous automatic evaluation metric used for comparing machine translation systems.

In Chapter 3, we provide the reader with the description of the distributed machine translation environment we developed in this thesis. We explain the developed mechanisms to handle multiple incoming streams of language data at the same time and the dynamic processing of this data based on the demand of the users at any given point in time. We show that our developed framework is able to automatically deal with the dynamic requests that might happen during typical university lectures, and we explain how the optimal sequence of steps with regard to the language processing are found. This chapter also introduces the developed tools for crowdsourcing translation data which helps to improve the underlying models of the machine translation systems. Those tools define the ways, how users can provide feedback, and how this feedback is then handled and distributed in our framework to the corresponding models.

Chapter 4 presents our results on the task of automatic learning of improvements for statistical machine translation based on user provided translation feedback. We describe how the data collected within the framework we described in Chapter 3 can be used to modify the underlying machine translation models, so that the overall translation quality improves. We furthermore describe which changes were necessary to the models described in Chapter 2 to make the models improvable and we explain the developed algorithms that were used for doing this. Furthermore we show a method for discriminating between noise and good translation feedback to reduce the effect of malicious users and bad feedback. We also show, how we can support users willing to provide feedback, by trying to provide an auto completion mechanism for translations being typed in by users. This includes the development of a modified decoding strategy for the decoder used throughout this thesis, which is also described in this chapter.

In Chapter 5 we show extensive evaluations of our developed algorithms and techniques. We compare the differences in computing time and the real time capabilities of our developed algorithms. We furthermore provide details on the translation performance and how the translation performance of our incremental training procedure compares to the standard batch processing training pipeline.

We also consider the effect of out of vocabulary words and evaluate how big the influence of out of vocabulary words is in our developed system for automatic improvement of machine translation quality.

Furthermore we provide results on the question how a hybrid learning approach, mixing standard batch training and incremental training, can help to improve the translation quality even more, while keeping the benefit of fast model updates.

In our evaluation we go beyond the simple question of how translation quality improves on unseen test data, but we also show that our proposed models are able to provide significantly better translations on the user corrected training examples. We also show that even an improvement of future unseen data of the same document is possible and we prove that on different language pairs.

Finally we evaluate the effect of the noise removal to verify that the reduction of malicious training data is possible within the developed framework.

In Chapter 6 we show the complete end to end system we developed throughout this thesis which was deployed as a lecture translation system at the Karlsruhe institute of technology. We describe the components used by professors to provide the lecture data,

and the endpoints we developed for students which allowed them to revisit past lectures and correct translation or speech recognition errors. This was fully integrated with our automatic improvement technologies, so that feedback from students helped to improve the overall performance of the underlying models.

In the end of this thesis, in Chapter 7, we give a conclusion of the presented work and the results we obtained. We also provide the reader with an outlook to future work which can build on the research done in this thesis.

2. Fundamentals of Statistical Machine Translation

In this chapter we present an overview of the underlying principles of statistical machine translation which were used throughout this thesis and which are therefore laying the foundation for our work.

We present the most common models used and the mathematical background behind the translation process used for finding the optimal translation hypothesis, including the algorithms used for finding the best translation in the search space, the so called decoding. In addition to this, we also explain the most commonly used evaluation metric, since its understanding helps with the interpretation of the experiments.

2.1. Mathematical Model of Statistical Machine Translation

The foundation of statistical machine translation was already made back in the 90s when (BPPM93) presented a new way of doing machine translation. They proposed the usage of a model based on Bayes formula, introducing the so called *fundamental equation* of machine translation.

$$\begin{aligned} \hat{e} &= \arg\max_{e} Pr(e|f) && (2.1)\\ &= \arg\max_{e} \frac{Pr(e)Pr(f|e)}{Pr(f)} && (2.2)\\ &= \arg\max_{e} Pr(e)Pr(f|e) && (2.3) \end{aligned}$$

The main contribution of this formula was that the search for the best translation $\hat{e}$ of a given source sentence f was reduced to the task of finding the translation e which maximizes the probability of being generated by f: $Pr(e|f)$.

Following *Bayes rule*, this problem can be reformulated by splitting the problem into two sub problems which are easier to solve than the combination of both into one single problem. The decomposition of the translation task then focuses on the computation of the probabilities $Pr(e)$ and $Pr(f|e)$ with ignoring the denominator in *Bayes rule*. This is possible, because the optimization is only searching for the maximum probability, not the exact value. The denominator is the same for all different $Pr(e|f)$ since the input f is fixed and is therefore not relevant for the search.

The probability $Pr(e)$ is the so called *language model*, which assigns a probability to the sequence of words in the target language, allowing to decide if the generated word sequence is a reasonable sentence in the target language without looking at the original input sentence.

On the other hand $Pr(f|e)$ is the so called *translation model*. It is able to assign a probability to the question how likely the original sentence f is, given the translation into e. This reduction into two sub tasks made the computation of the $argmax_e$ easier because of the better estimates and the ability to directly evaluate a generated translation hypothesis.

With the ongoing research, this model was modified into a noisy channel log-linear framework, which allowed the inclusion of additional models over the baseline language and translation models while keeping a mathematical model which still explained the use of these models.

In the log-linear framework, different feature functions h_m are used (ON02) with weights associated to them according to their contribution to the final probability. This resulted in the reformulation of the probability into:

$$Pr(e|f) = p_{\lambda_1^M}(e|f) \tag{2.4}$$

$$= \frac{exp[\sum_{m=1}^{M} \lambda_m h_m(e,f)]}{\sum_{e'} exp[\sum_{m=1}^{M} \lambda_m h_m(e',f)]} \tag{2.5}$$

The normalization $\sum_{e'} exp[\sum_{m=1}^{M} \lambda_m h_m(e',f)]$ however is not needed for the same reason as above since we are only interested in $\arg\max_e Pr(e|f)$, the sentence e that maximizes the probability of the translation of f. Again removing the normalization made the computation efficient and feasible. It is even possible to go one step further, since the exact value of the probability is not needed, only the order of the probabilities is important. Since $exp(x)$ is a monotonic function, searching for the e which minimizes $-log(Pr(e|f))$ is feasible as well. One of the side effect there is, that the usage of log is also numerically more stable, which is important for very low probabilities.

Therefore we obtain as a new objective function:

$$\hat{e} = \arg\max_{e} \sum_{m=1}^{M} \lambda_m h_m(e, f) \tag{2.6}$$

When the only models h_m that are used, are the language model $Pr(e)$ and the translation model $Pr(f|e)$, this boils down to the original bayes formulation from above(ON02), explaining the noisy channel model as an extension of the original fundamental formula.

$$h_1(e, f) = \log Pr(e), \quad \lambda_1 = 1 \tag{2.7}$$

$$h_2(e, f) = \log Pr(f|e), \quad \lambda_2 = 1 \tag{2.8}$$

In this special case we only use the two models language and translation model. But the main benefit of this approach is the mathematical foundation for integration of further models. This allows the integration of more information into the decoding process for more elaborate decisions when the best translation is searched for, given that the scaling factors λ_m have the right proportions.

2.2. Optimization

Finding the best scaling factors however is another problem that had to be solved for machine translation. It is a multidimensional optimization problem where heuristics for optimization are used.

In addition to the estimated regular model parameters, the scaling factors introduce weights to the different models, resulting in better translations. For the optimization of those λ_i different approaches are used today, an overview of some of these techniques can be seen in (CF12). Our work used the technique described in (Och03) and (VV05), the process of the *Minimum Error Rate Training (MER)*. This is a common approach based on Powell's algorithm (WP67) for the parameter tuning also available in the Moses statistical machine translation toolkit (BHF09).

While there is research going on in evaluating other optimization methods, minimum error rate training is still state of the art among the approaches. In this thesis the part of the optimization only plays a minor role why a further investigation of the different approaches is not necessary at this point.

The idea behind MER is to optimize the scaling factors λ_i in such a way, that a good translation gets a better score assigned compared to bad translations. This is done with a so called *development* set of sentences S, where one (or more) reference translation r_s

for every foreign sentence f_s is available. The next building block of the MER training is a function which assigns an error to every translated sentence $E(r, e)$, where e is the translation and r is the reference. With this definition in place, following the notation in (Och03), the optimization of the λ_i becomes an optimization problem:

$$\hat{\lambda}_1^M = \arg\min_{\lambda_1^M} \{\sum_{s=1}^{S} E(r_s, \hat{e}(f_s; \lambda_1^M))\} \tag{2.9}$$

where

$$\hat{e}(f_s; \lambda_1^M) = \arg\max_{e \in C_s} \{\sum_{m=1}^{M} \lambda_m h_m(e|f_s)\} \tag{2.10}$$

and C_s is the set of candidate translations for a sentence f_s.

The minimum error rate training generates sets of candidate translations (the *n-best* translations) for every sentence in the development set and computes the error function for those translations. The optimization is then based on Powell's Algorithm (WP67), (Pre07) where an initial random point for λ_1^M is selected. The optimization optimizes a single dimension λ_γ of λ_1^M by identifying those intervals where the translation of any of the sentences changes and the resulting change in error rate. These intervals and changes in error are collected over the whole development set and the whole *n-best* translations allowing the identification of the best value for λ_γ by only evaluating the relevant values.

This is done over the different dimensions of λ_1^M which results in a new set of scaling factors. The search for the points, where the translation changes is straight forward, when only one dimension is changed at a time. The selection of the best translation can only change at those points where lines representing the scoring dependent on λ_γ of two different translations intersect. These points are easy to find as they can be computed as a simple line intersection.

Among all these possible intersections, that value for the current λ_γ is estimated, which maximizes the evaluation score of the translation for the development set and this setting is used as the optimal scaling factor for the next optimization round.

The development set then is translated again with the updated scaling factors and another iteration of optimization is started. This process iterates, until no further significant gains are seen or when a maximum of iterations has been reached.

2.3. Language Model

In 2.1 we introduced the concept of splitting the translation process into the computation of a *language model* and *translation model* (and other models).

Since the language model played an important role in our developed algorithms and methods, we are giving a more detailed description of the used language model, which is an *n-gram* based language model, as used in many current systems.

The idea of a language model is to assign a probability to a monolingual sequence. This probability describes how likely this word sequence is in this language. The standard approach for a language model in statistical machine translation is typically an *n-gram* based language model which estimates the probability of a word, based on its preceding $n-1$ words. To compute the probability of a longer sequence, a language model uses the chain rule to decompose the probability of a word sequence into the subsequences of smaller length.

$$\begin{aligned} Pr(e) &= Pr(e_1, \ldots, e_l) && (2.11)\\ &= Pr(e_1)Pr(e_2|e_1)\ldots Pr(e_l|e_1, \ldots e_{l-1}) && (2.12)\\ & && (2.13) \end{aligned}$$

The *n-gram* language model makes an approximation for these smaller word sub-sequences and assumes that the probability for seeing a word given its preceding words can be approximated by looking at the probability of the last n words alone. This results in the modified version of above equation:

$$\begin{aligned} Pr(e) &= Pr(e_1, \ldots, e_l) && (2.14)\\ &= Pr(e_1)Pr(e_2|e_1)\ldots Pr(e_l|e_1, \ldots e_{l-1}) && (2.15)\\ &\approx Pr(e_1)Pr(e_2|e_1)\ldots Pr(e_{l-1}|e_{l-n}, \ldots e_{l-2})Pr(e_l|e_{l-(n-1)}, \ldots e_{l-1}) && (2.16) \end{aligned}$$

By doing this approximation the number of unseen word sequences for which probabilities are needed is highly reduced, allowing a storage of all statistics that can be extracted from the training corpus, while exact statistics would highly suffer from data sparsity and nearly impossible to store. The problem of missing coverage on long range dependencies introduced by this approximation is a research topic on its own and not within the scope of this thesis.

The probabilities used in $Pr(w|history)$ are usually based on observations made on the training corpus:

$$Pr(w|history) = \frac{count(history, w)}{count(history)} \tag{2.17}$$

But even with the *n-gram* based approximation, very often word sequences will be seen in the search space at translation time which have not been seen in the training data. This

would obviously cause the above equation to generate a bad language model probability for this sequence and would prevent the system completely to generate those sequences. For that purpose different smoothing and backing off techniques were introduced. One of the most used techniques today is the *Kneser-Ney* smoothing which had been shown to generate good results for the purpose of machine translation. Because of its still very strong usage we only focus on the description of the *Kneser-Ney* based smoothing which was used in this thesis.

The idea that drives the *Kneser-Ney* smoothing (CG96) is that lower order word sequences are significant, if their extended versions into higher order *n-grams* have low counts as well. This is directly intuitive for example for a phrase like *San Francisco* where it is obvious that the unigram of *Francisco* should not be overestimated only by the fact that it appeared often in the training corpus. Most of its occurrences have the word *San* directly before, so if you see the word *Francisco* without *San* directly before, don't overestimate its probability. On the other hand, if you look at another word like *pressed* (which had exactly the same count as *Francisco* in one of our training corpora), you would expect a much higher probability for an occurrence of this word, even though the word in front of it has not been seen in this sequence before as well.

Details of the Kneser Ney smoothing can be found in (CG96), as well as in this thesis at a later point where we describe how we developed techniques to keep the benefits of Kneser-Ney even in the case of incremental learning, which was necessary, since the incremental learning will often face new word sequences which needed the application of smoothing techniques.

2.4. Translation Model

Besides the language model, the *translation model* (TM) is another big building block used in todays phrase based statistical machine translation systems.

The task of the translation model is to assign a probability to an English and foreign language sentence pair which indicates how likely it is that the English sentence e is a translation of the foreign sentence f (compare the fundamental equation of statistical machine translation).

This is usually done by splitting the sentence into smaller subparts for which the translation probabilities are known. While in the very early days of machine translation, single words were used as building blocks of a translation, in recent years the splitting into multi word

phrases became the standard (ZON02), and still is the foundation for current state of the art statistical machine translation systems. The phrase based approach relies on the computation of a word alignment between the input sentence and the output sentence. This alignment is usually based on what was proposed in (BPPM93) and is computed in an expectation maximization manner. With an existing word alignment it is then possible to compute probability for word to word translations between the languages. With the introduction of phrases, the extraction shifted to those longer word sequences, which in addition to the underlying word translation probabilities also looked at the phrase occurrences and co-occurrences with the target language phrase for better estimation of the translation probabilities.

It is important to note at this point, that the computation of the final translation model probability for a whole sentence usually is based on an assumption that is arguably wrong. The assumption is, that the probabilities of two phrases used for translation in the same sentence are independent of each other. There are multiple reasons for this simplification, one is the fact, that for the estimation of a probability based on other phrases or words, there is not enough training data, which means such an approach runs easily into data sparsity problems. Another problem is the efficiency to compute the translation model scores. With the indepence assumption, the computation becomes very easy:

$$Pr(f|e) = \prod_{phrase} Pr(f_{phrase}|e_{phrase}) \tag{2.18}$$

For that reason, this assumption is still made in most state of the art translation models and usually the introduction of other models is made to reduce the effect of this simplification.

In addition to the smoothing in the language model as mentioned in (2.3), translation models also suffer from data sparsity which results in bad estimates for the translation probabilities. While the situation for the language model is that the sequence has never been seen at all, the problem for the translation model is that a phrase has been seen just once or twice, resulting in very bad estimates for the phrase probabilities, usually overestimating the probability for a phrase pair. For this purpose the usual approach is to also apply smoothing techniques. In this thesis we used Good Turing discounting(CG91) which is a common technique with proven success when applied to the translation model in a statistical machine translation system.

The details of Good Turing discounting are presented in a later chapter. But the main idea used in Good Turing estimation is that the probability to encounter a phrase pair is based on an underlying binomial distribution.

2.5. ITG Decoding

The work presented in this thesis is based on an underlying machine translation decoder using *inverse transduction grammars*. The usage of *ITG* for machine translation was proposed by (Wu97) and has proven to provide similar translation performance as *beam search* decoders (WW97). At the point of writing this thesis, most of the freely available translation engines have at least support for chart based decoding in contrast to the stack based decoding.

We used and extended upon the implementation of (ZV07) which is a very efficient implementation of the data structures and algorithms, that allowed running a full fledged translation system on mobile phones and limited resource platforms. We decided to use this implementation as a basis, because of the requirements we expected for the storage of information and speed of computations.

In general the task of the decoding in a machine translation system is to find the optimal translation based on the given underlying mathematical models. In (Kni99) it was shown that this kind of translation approach is *NP-complete*.

A common approach for this search algorithm is the so called *stack decoding* (WW97).

The most often used approach of implementing this, is to generate the translation of the input sentence in a left to right manner, starting with the first target word until the last word in the hypothesis is generated. This is done by building stacks of hypothesis, based on the source words covered by the hypothesis in those stacks.

Within such a stack, the generated hypothesis cover the same source words and are therefore directly comparable. This allows pruning among the generated translation hypothesis and is basis for speeding up and reduction of the search space. After selection of the best translations within a stack, the hypothesis can then be expanded by translations of words not covered yet to generate new sub translations which cover larger parts of the sentence.

This results in a modified shortest path search, where the best hypothesis for substrings of the original source sentence are expanded until the whole sentence is covered by translations and the best translation is the one with the minimal costs.

In contrast to that approach, (Wu97) proposed a different approach for treating the generation of bilingual data as a result of an *inversion transduction grammar*. A simple transduction grammar (LIS68) generates two streams:

$A \rightarrow Bx_1y_2Cz_1$, meaning that A is mapped into symbols x and z in language 1 and y in language 2 and the two non terminal symbols B and C.

In (Wu97) the proposed *inversion transduction grammar* extends the simple transduction grammar by allowing production rules, where the target language orientation of words is inverted:

$A \rightarrow \langle BC \rangle$. The interpretation of this is that in the final output the production for C is output first in language 2, while for language 1 the final production for B is output first.

For the ease of notification, $A \rightarrow [BC]$ denotes the production rule, where in both languages the final production for B is output first. As a result (Wu97) shows that any inversion transduction grammar can be converted into an equivalent grammar with the following production rules:

$$S \rightarrow \epsilon/\epsilon$$

$$A \rightarrow x/y$$

$$A \rightarrow x/\epsilon$$

$$A \rightarrow \epsilon/y$$

$$A \rightarrow [BC]$$

$$A \rightarrow \langle BC \rangle$$

This is an important observation, because it converts an inversion transduction grammar into a normalized form and allows efficient decoding algorithms for the translation. Usually the Cocke-Yange-Kasami algorithm (You67), (Kas65) is used for finding a chain of productions for the translation of the sentence.

In our application and throughout this thesis, we did not allow those productions containing ϵ. In addition to that the production $A \rightarrow x/y$ are the entries of the phrase table and the only other rules used in our grammar are the two *glue* productions $A \rightarrow [BC]$ and $A \rightarrow \langle BC \rangle$. With these rules and the function

$$\text{trans}(f_l \dots f_m) = \text{set of phrase translations for } f_l \dots f_m$$

the decoding for finding the best translation for a sentence $f = f_1 \dots f_n$ can be solved with the CYK-algorithm:

The idea behind this algorithm is to compute the final translation from bottom to top. Starting with single phrases and then stepwise increasing the span of covered words in the sentence $f_1 \dots f_n$ adjacent translated parts of the sentence are combined. Either in linear or in inverted order. The final translation of the sentence is the best translation in chart entry $C_{1,n}$.

Data: InputSentence $f_1 \dots f_n$
initialize chart C of size $n \times n$;
for $i \in [1, \dots n]$ **do**
  for $j \in [1, \dots n]$ **do**
    $C_{i,j} = \text{trans}(f_i \dots f_j)$;
  end
end
for $length \in [2, \dots n]$ **do**
  for $start \in [1, \dots n - length]$ **do**
    $end = start + length$;
    for $split \in [start + 1, \dots end - 1]$ **do**
      $C_{start,end} = C_{start,end} \cup \text{merge}(C_{start,split}, C_{split,end})$;
      $C_{start,end} = C_{start,end} \cup \text{merge}(C_{split,end}, C_{start,split})$;
    end
  end
end

Algorithm 1: Decoding based on CYK-algorithm

A schematic overview over the first steps in such a decoding run is shown in 2.3

An important aspect of *ITG* based decoding in statistical machine translation is the merging of two adjacent hypothesis in the decoding chart. Since the decoder always keeps an intermediate score for every hypothesis in the chart, the question is how to merge the scores of two adjacent hypotheses into the score of a single one.

Following the equation 2.10, the score of a single hypothesis can be decomposed into the sum of its contributing model scores. This allows to split the translation problem into the combination of single scores of the different models instead of one single score combination. At this point the method of combining the translation model and the language model score are shown as a reference to other models.

The combination of the translation model boils down to the question how the translation model score for the whole (combined) hypothesis is computed. This thesis used the same approach as commonly used by other state of the art systems as well, which is the sum over the constituent phrase table scores for every used phrase is the translation score of the combination of those phrases. In this case the assumption of independence between the used phrases reduces the computational time quite significantly. Without this a simple

addition would not have been sufficient, since the moment two partial hypotheses are combined, the probability of the used phrases would change based on the additional hypotheses in the combination.

In contrast to this, the combination of language model scores for two adjacent hypothesis is slightly more complicated. When the language model score is computed for a hypothesis covering only a sub sequence $e_j \dots e_k$ of the final translation, neither the preceding words $e_1 \dots e_i$ nor the following words $e_m \dots e_n$ are known. Therefor the final score is approximated either by assuming start and end of the sentence or some other assumption, of how to approximate the language model score of this part. This however means that the combination of two adjacent hypothesis requires a subtraction of the approximated parts and then an addition of the correct language model scores for the affected parts.

In our used *n-gram* language model, the scoring of the last $n-1$ words of the left and first $n-1$ words of the right hypothesis is subtracted and then a new score for the n-grams covering the combination of these words is added:

$$lm_{merged} = lm_{left} + lm_{right} - lm_{left_{k-(n-1)}^{k}} - lm_{right_1^{n-1}} + lm_{left_{k-(n-1)}^{k} right_1^{n-1}} \quad (2.19)$$

To speed up the decoding process, the left and right language model context - the relevant words needed for a computation of the updated language model score - can be stored for every hypothesis, which allows faster computation of this merging operation.

Finally an interesting thing to note for ITG based decoding is that there are multiple ways to create the same hypothesis using different combination of hypotheses (compare Figure 2.3). In (ZN03) this was investigated and they showed that using the *ITG-constraints* it is possible to remove duplicates which are generated caused by different recombinations resulting in the same output. The *ITG-constraints* are restricting the usage of reordering for the right constituent of a new combination, to be of the opposite type than the current combination. It is easy to see, that this results in a more linear construction of hypotheses, especially when few reorderings happen within a language pair.

Even with this restriction however the number of hypothesis becomes so large, that pruning mechanisms are needed. Here different pruning methods as described in (RVW06) like keeping the *n-best* translations for every chart entry and keeping all translations within a certain beam can be used.

2.6. Evaluation

A very important part in the improvement of machine translation systems is the question how to measure translation quality. While a human evaluation seems to be a possible solution at first sight, it has many disadvantages though. The most obvious problem with human evaluators is the speed of performing such an evaluation. Manually scoring the quality of a translation is a very time consuming task and therefore not possible in situations, where continuously improvements at a large scale need to be measured. The problem of the time comes in hand with the problem of the cost. People doing manual scoring need to be paid as well, which makes the evaluation an expensive task. And finally a human evaluation is very subjective. One human judge might like a translation better than another one, which makes it important to have multiple judges, which will make the two stated problems even worse.

These factors caused a lot research in the search for automatic metrics. In recent years many different metrics had been proposed (PRWjZ02), (SDS$^+$06), (BL05), (Dod02) and it has been shown that those metrics correlate with human judgement (PRWjZ02), (BL05). This work will use the *BLEU* metric, which is the most commonly used one nowadays.

The basic idea of *BLEU* as described in (PRWjZ02) is to measure the number of n-gram matches between reference translations and the translation hypothesis. Given one or more reference translations, the modified precision n-gram score is defined as:

$$p_n = \frac{\sum_{C \in Candidates} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in Candidates} \sum_{n-gram' \in C'} Count(n-gram')} \tag{2.20}$$

where $Count(n-gram)$ is the number of occurrences of the specific $n-gram$ in the translation, and $Count_{clip}(n-gram) = \min(Count(n-gram), MaxCountInReferences(n-gram))$. This modified precision will already penalize translations which are too long, since the number of unmatched n-grams will be higher than the clipped counts. On the other hand this also means, that shorter translations would be preferred by such a metric, which was prevented by introducing another component, the brevity penalty. The brevity penalty is defined as:

$$BP = \begin{cases} 1 & if c > r \\ e^{1-r/c} & if c \leq r \end{cases} \tag{2.21}$$

where c equals the length of the candidate translation and r is the length of the reference translation.

This results in the final formula for the *BLEU* score defined as follows:

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} \frac{1}{N} \log p_n) \tag{2.22}$$

, with p_n as defined above.

Even though it is a known problem that this definition of BLEU ignores the severity of translation errors[1] and it treats flexions of words the same way a wrong word would be treated, *BLEU* is still the mainly used metric for comparing the output of machine translation systems and it has proven to be helpful for judging among different translations and to drive the development of machine translation systems. For these reasons it was also chosen in this thesis as the metric for the evaluation purposes.

[1] compare a missing "*the*" with a missing "*not*"

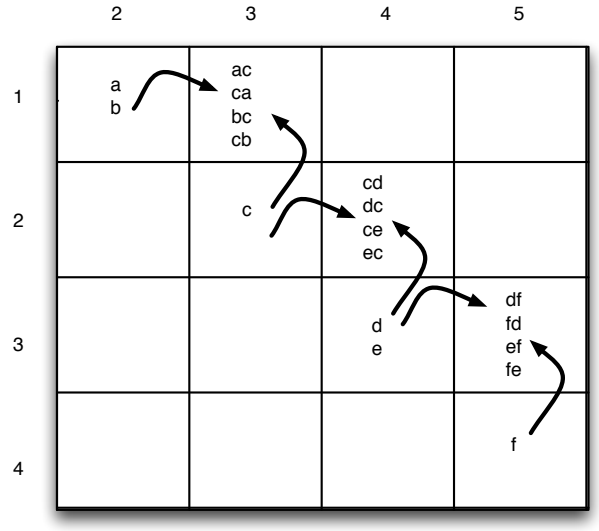

Figure 2.1.: Schematic overview of cyk decoding Step 1

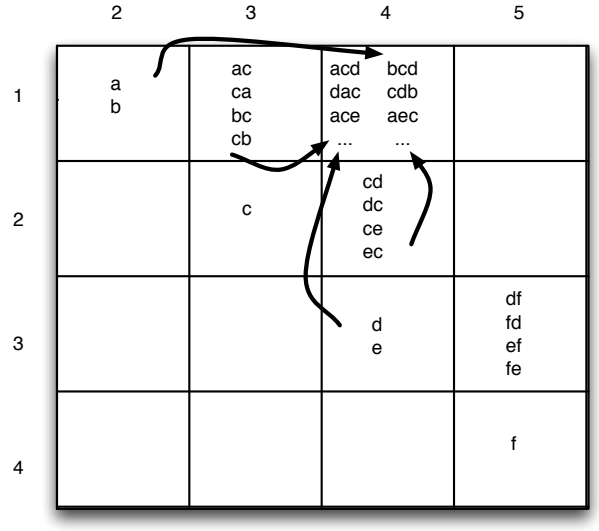

Figure 2.2.: Schematic overview of cyk decoding Step 2

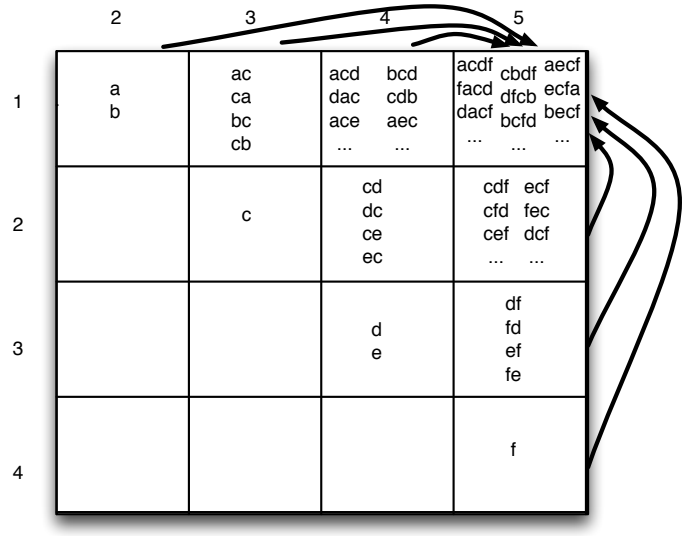

Figure 2.3.: Schematic overview of cyk decoding Step 3

3. Distributed Speech to Speech Translation System

In this chapter the algorithms and the distributed network architecture we developed for handling distributed language processing tasks in parallel are described. The developed architecture allows for dynamic adjustments to the processing of the incoming data without breaking or stopping the active overall processing. The developed framework is able to automatically find the optimal possible routing of data streams over multiple different service providers, depending on the current need for data transformation.

We invented mechanisms which are able to fulfill different needs and tasks of a speech to speech translation system on demand depending on the current audience or presenter situation.

The proposed architecture was put into regular use at the Karlsruhe Institute of Technology after an introduction in June 2012 [1] and has been serving simultaneous speech translation needs since.

While our main focus during development of this architecture was the task of real time speech translation, the proposed architecture is not limited to that and can be easily applied to different tasks, that involve a processing pipeline. In a sample setup we also applied it with success to the task of providing optical character recognition as a parallel channel in a simultaneous speech translation situation.

The automatic improvement of the machine translation (and possibly other) components

[1] http://www.spiegel.de/unispiegel/studium/dolmetscher-fuer-die-vorlesung-kit-entwickelt-uebersetzungsprogramm-a-838409.html

was one of the major requirements we set when we formulated the goals for the architecture. For that reason we designed the architecture in a way, that it allowed us to share features needed for improvements among different services.

To our knowledge the developed architecture is the first such dynamic architecture used for speech to speech translation systems and in addition to that, it is to our knowledge the first architecture with a full integration of mechanisms needed for automatic improvements.

3.1. Requirements for Distributed Framework

In the beginning we defined the requirements for the framework we planned to develop and identified the following objectives which were necessary for the usage of such a system in multi channel distributed environments.

- Since the system was developed with the initial use case of a lecture translation system in mind, where students who might otherwise be unable to understand what was said, the most obvious requirement for such a framework is its *robustness*.

  This means, that in this distributed environment, a single failing component should not break the whole system. Ideally, the system should continue to provide the best possible service with the remaining working components. Even for situations where a non replaceable service failed, the overall system should not fail. As a result, other tasks which are handled in parallel should be unaffected by the failing component and for the task where the service was essential, this should not automatically mean a complete loss of data. In the worst case of failure, the system should still allow a generation of everything in post processing steps, when problems with the components have been resolved.

  If the failing component is in addition replaceable by other components or alternative processing would result in similar outcome, the framework has to be able to recover from the failing state, without users noticing this problem and continue to provide the service.

- In a distributed environment the *scalability* of the system plays another important role.

  With increasing acceptance and quality of machine translation and speech recognition services the usage for such systems will increase. The system therefore has to scale, without major efforts in modifications of the framework itself when the

system grows. The requirement is that the system has to be extendable to provide the services to more users without other users noticing changes. In addition to the requirement of being able to provide service to more users, the same holds for the number of services being provided by such a system. We identified that it is very important to extend the system to additional services, without affecting the existing provided services.

- For the adoption of such a framework, the *usability* of the framework is crucial.

  The system has to be usable by non expert users. This affects both, on the one hand the speakers who want to make use of a speech translation system for their audience and on the other hand the audience, who is willing to use a speech translation system to understand someone else.

  The system has to be developed in a way, that it can provide front-ends to users which abstract from the complex logic and provide an easy to use entrance into the system. In the best case, the usage for a speaker should look like opening an app or a web page starting the recording and then starting the presentation or talk. This optimal setting would also include the identification of the language the speaker is using. Since this work did not focus on the problem of language identification, we made one step back and allowed the user to provide the input language that is used while speaking. On the other hand we still kept the requirement, that our developed system was able to deal with language identification, when such a component is available.

  On the audience side the usability is also very important, because we want the audience of our system to provide feedback on which we later improve the components. Accordingly the usability includes the presentation to the audience. One important aspect of this is that we wanted to decouple the presenter from the audience in a way, that the presenter does not have to know anything about the audience at all. Consequently this meant that the services had to be selected dynamically based on the current requirements of the audience, accommodating for possible changes in the audience during run time. In contrast to a static selection of processing when the presenter starts a presentation.

- For the reason that we wanted a distributed environment with multiple language processing tasks running at the same time, the *parallel processing* of the different data streams in the system was another important part.

The system should be able to provide services to users in the end, without users knowing of each other and therefore the system needs to be able to handle multiple requests for different services and processing at once. Especially at universities, classes of different subjects take place at the same time. Hence a simultaneous speech translation system should be able to provide simultaneous translation results to different classes at the same time. Closely related to the need for scalability of the system, the parallel processing has to be hidden from the users and different service requests should not interfere with each other.

- We also decided that we want *long time storage* of the data handled by the provided services.

  This was for two reasons. On the one hand we wanted to enable users to review past recordings, which is especially helpful in the case of a lecture translation system, where students are able to review past lectures and they are able to use this stored information as an additional tool for learning. On the other hand - and closely related to the review of past recordings is the improvement of the services which was one of the main goals when developing this system. With a storage of past processing, it is possible to enable users to provide corrections to the system in a situation where they are not anymore closely following a presentation, but the user is now able to revisit the information at a slower speed, decoupled from the original real time speed. This creates the possibility of providing better feedback to the system.

  Furthermore the storage of the data also is part of the robustness, allowing a reconstruction of output for processes that failed during the realtime processing.

- As already described in the scalability requirement, we wanted the system to be *dynamic*, meaning easy to extend and still dynamic when it comes to new requests. This involves the automatic identification of components which are part of the processing, but also the automatic freeing of unused components when the needs during a longer processing task change. One can imagine the situation in a speech translation environment, where previously no Chinese translation was needed for an incoming speech signal, but later during the usage a Chinese translation is needed. This maybe either because of a change of audience while the system is running, or because of a change in the input signal, like a previously Chinese speaking presenter switching to english presentation.

  While the obvious and easy solution would be to always generate all outputs possible, this is a very expensive solution and does not scale well. It would occupy components

in the system which might be necessary for a different language processing task (as in a different lecture hall), but are not used in the current one. For these situations a dynamic approach is necessary which allows to dynamically allocate and free the needed components.

- Finally we wanted our system to be *extendable* and to not only be able to handle speech translation tasks, but more general tasks. Even within the environment of a lecture translation system, multiple different use-cases come to mind, like an automatic keyword extraction or the translation of slides which are part of the presentation. For that reason we wanted to minimize assumptions which are based on the fact that we deal with audio and text messages, but keep the system open to as many tasks as possible.

These requirements formed the technical requirements we identified which defined the minimum capabilities the framework must meet to be successful in a distributed language processing environment.

Beside these technical requirements however, we identified additional requirements with regard to the usability. This is especially the case since we wanted users to give feedback on recognition errors. For this reason we defined the additional following requirements to maximize the acceptance and as a result of that the feedback from users.

- *Multiple users* should be able to use the system at the same time and give feedback. Preferably on their own machines and at any time (in the lecture environment we expected that users - students - start to use the system more heavily when preparing for the exams). This also includes, as already described in the technical requirements, the handling of multiple streams within the same session. This might be required for example in the lecture translation environment, when a lecture is held by more than one person, so that it is possible to provide different translation services for the different speakers.

- When users provide feedback to the system, we had to give them some kind of reward - they had to see that *they benefit* from giving feedback. While money obviously comes into mind when speaking of rewards, the reward in a translation environment can be much simpler than that: correct translations in the future and a correct translation for an attended lecture. A user should be able to see that improvements he made have an impact and he does not need to make the same corrections over and over again. With limited resources money wise for such experiments in the

University environment, we identified this as the preferable reward.

- This directly brought us to the next point. When feedback is given to the system, the models have to be *updated immediately*. Waiting for a complete retraining is too slow and would not satisfy the previous requirement. The research for this is discussed throughout the rest of this thesis.
- When collecting feedback from users it is important to keep in mind that the quality of this data varies and not all users will provide good translations. We expected even malicious users who want to manipulate the system and reduce its usability. For this reason we identified the need for *identification of bad translations* and mechanisms to exclude them from the automatic trainings.

The above described requirements built the base of our developed simultaneous speech translation architecture. The details of the algorithms and components we developed are laid out in the rest of this chapter.

3.2. Framework for Distributed Language Processing

With the above made observations in mind we identified, that the framework to develop needed to provide different services to users depending on their current needs. Users of the services are on the one hand consumers, who want to consume the output generated by internal components within the framework, or on the other hand they are producers of data which needs to be processed by the system. There is however no strict distinction between both, and a producer can also consume output of the system at the same time (this can even be the case in a lecture translation environment, when the interactions with the audience should also be handled by translation, where for example the presenter interacts with students asking questions in a language the presenter does not understand).

With this observation in mind, we developed the framework in a way, that it allowed components to provide services in a service oriented architecture (Erl06). The framework we developed was able to meet the requirements stated in the first part of this chapter, while following the standard approach of providing services to the end user.

For solving these issues, we identified one main task for the system to develop. In general we wanted to solve the problem of transforming data (language data) generated by some producer, into the desired format requested by some consumer. This transforming itself also consisted of multiple smaller steps which do some specialized transformation on their own.

Data that is to be transformed needs to be distinguishable by the original producer who fed this data into the system. This lead us to the concept of separate *streams* which we integrated into our developed framework. In the case of speech translation, such an input *stream* could be the audio signal of a single speaker, text input when someone input text directly into the system, or even pictures like screenshots of the slides that were shown in a lecture environment.

The data of the stream is defined by a sequence of one or multiple packages containing the data that needs to be transformed to generate the output requested by the consumers.

As already pointed out above, in many cases where such a system can be used, it might happen, that multiple input streams are generated at the same event. This can be the case in a conversation, when two people use a speech to speech translation system to talk to each other in different languages. This would mean two streams where we have audio data in different languages from the two participants, both streams would need to be transformed into the language of the other person. But also in the situation of a lecture translation system, one lecture can generate multiple different types of data the consumers are interested in. This would be the case for the audio data of the presenter, slides that are projected and translated after optical character recognition or even additional audio channels for a second presenter or a feedback channel from the audience. To deal with this situation in a meaningful way and allow the combination of multiple streams to make it accessible for participants in such a system, we developed the concept of so called *sessions* which bundled multiple streams into one logical unit.

We started the description of this system with pointing out, that the processing of the data was split into multiple smaller transformation steps. These steps were handled by different units which are attached to the system and which we called *workers*. The single task of such a worker is to accept data in one format and then convert this data into some other format. This other format can be for example another modality of the data, another representation (like audio encoding) or different language.

In the processing of a stream it is also possible - and the most common way - to combine multiple workers into a chain, so that other workers continue to transform the output they receive from a previous worker. This chaining allowed a modularization of the services provided by our system.

To handle this mechanism easily and in a transparent way, we developed so called *processing graphs*. The idea behind those graphs was that every session has one single directed acyclic graph which describes the processing steps of the data. The edges in this graph are

the connections between the components which produce, process or consume the data. A connection is made from the output of one node to the input of another node, where the accepted inputs and generated outputs of a node are defined when a processing component is made available to the system.

This is an extension over a principle which was first described in (Wai96). We enhanced this principle by developing additional mechanisms which allowed automatic routing and the dynamic allocation and freeing of components used for the data processing, allowing a distributed multilingual version which was usable for other tasks as well.

Within this architecture it is important to understand what it means if one component failed. That would mean that a node in the processing graph is not working anymore which would cause all following nodes in the graph to stop working at all. For this reason we also developed a fall back mechanism which dealt with failing nodes. In the case of a failing component, we started a search for an alternative route using available components attached to our framework which can provide the same data conversion as the failing component. Once such a path was found, the original path within the processing graph was changed, by removing the failing component, rerouting the data into the new found path and connecting the output of the new path to the node previously attached to the failing component. When a node was removed from the processing graph, we also took care of cleaning up the system. This was done by checking, whether a node's output is still connected to another node's output. If that was not the case, this node produced output that was not consumed by any other component anymore and the node was completely removed, freeing the underlying component for usage by other sessions in other processing graphs.

With the processing graph at hand and modeled as a central point where the connections and data processing was managed, remaining question was how to feed data into the system for the further processing. For this purpose we developed an interface open to the public, where a client was able to attach itself and provide data of a certain modality and language. Since we already mentioned that it makes sense to allow for multiple streams within one session, this client can also provide multiple inputs and feed different sets of data into the system. The moment a client attached to our system was also the moment where the initial session object was created with an initially empty processing graph.

On the other hand of the spectrum, we also needed a possibility to present the data processed in the processing graph. For this purpose we developed the opposite to the client, the so called *display*, which was a sink in the processing graph. A display was

able to connect to a session and create an output node, where data of certain types was collected and then presented or stored. It was also possible for a display to accept multiple streams at once, without the need of using multiple connections.

In total we identified the three different types of modules for the processing in our distributed processing environment, the *client* for providing or generating the data to work on, the *worker* for the processing of the data and finally the *displays* for the consumption of the processing results.

In addition to these external modules which handle the data processing, there was also the need for one central unit which handles the connections and data exchange between those components, meaning the unit which took care of the different sessions and their processing graphs. For this purpose we developed the so called *mediator*.

Another component which one might think of, and we treated as some kind of special case of a display was the *storage* component. The storage handled the task of storing and distributing persistent information. This component was a little bit different to the display components, since we also allowed this component to provide data. This was on the one hand the case, when a past session was revisited, but also for the case of user made corrections. In this case the storage unit provided workers with relevant information they were able to use to improve their models.

The following sections of this chapter will provide the reader with the details of the described components we developed. A high level overview over the components which were part of our developed system can be seen in 3.1. The mediator in the middle of this diagram was developed with multiple processes running in parallel. We used Erlang for implementation because of its excellent support for handling parallel processing and robustness when it comes to distributed architectures.

3.2.1. Data Streams and Services

As mentioned earlier, when we started to develop our system, we identified the need to handle different input signals within the same session. This is immediately clear in the environment of a lecture translation system for universities, where a lecture might be presented by two different speakers, one presenting the first half and another one the second half - possibly even in different languages or even alternating in the form of a dialogue or discussion.

One can think of many more reasons why it makes sense to model different sources of

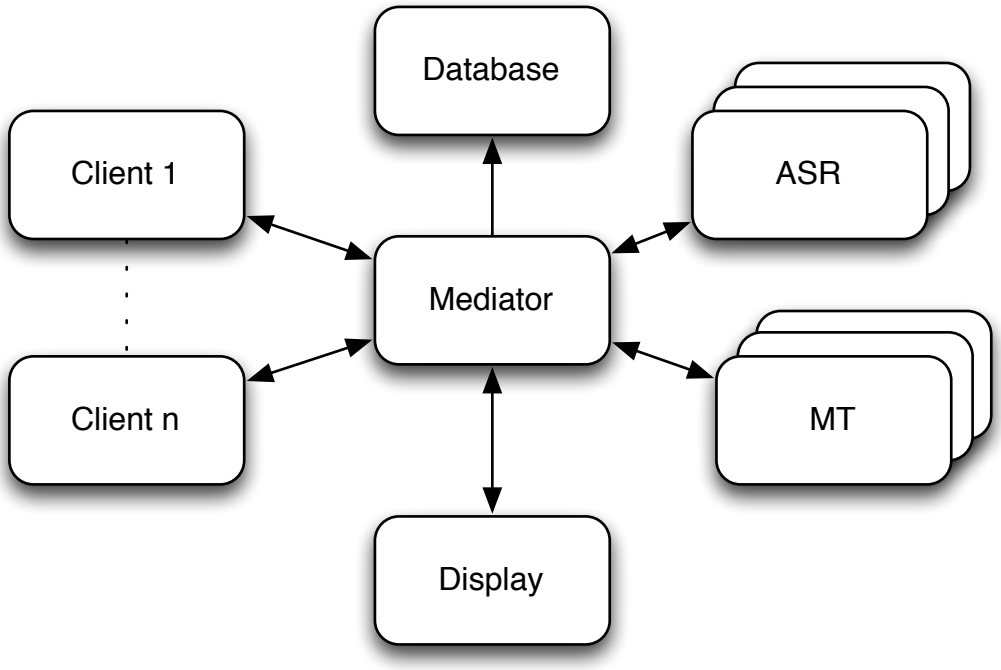

Figure 3.1.: Schematic overview of the interaction between external components and the language processing framework

input in a speech translation system, that is why we invented a mechanism that allowed us to handle this directly by the design of the infrastructure.

The idea behind solving this problem was, as briefly touched above, to assign all data within a session to that input node in the graph, that originally brought this data into the processing pipeline. This was done by adding a stream descriptor to every data package being sent, the so called *streamid*. Typical stream ids were for example `speaker1` and `slides` during the usage of the architecture as a lecture translation system. This naming of data packages allowed us to identify what stream a package is part of and allowed easier retrieval of past session information, even when the whole session information was just dumped to a file without the usage of a database to better organize the data.

In addition to that we also introduced additional mechanisms to annotate the information contained in a data package being sent on a stream. This annotation was intended to give information to the worker which was going to process the data next, and it made the system more robust by preventing wrong interpretation of the data. A worker was able to check if the package received contained data it was able to process. This concept of data encapsulated in the packages was integrated very deep into our developed architecture by making it part of the information used to create the processing graphs.

Since one of the tasks for the mediator process was to find the set of workers and the correct processing order in which they need to work to transform data from the input format into the requested output, it was important to know which workers can process the output of which other workers.

As indicated we were able to solve this problem when we introduced the concept of *data types* encapsulated in a package and the language the content is in. Here the concept of the language remains abstract and is not necessarily bound to a real world language. Only for the usage of the architecture in a lecture translation environment the language also identified a real world language.

The type of data describes the current encoded type in a package and is dependent on the implementation of the workers (or clients) generating this type. Our protocol definition established a few initial types which were sufficient for the usage in the lecture translation system we developed, namely: `audio`, `text` and `image`. We did however not impose any restrictions on the type itself and essentially any data could be encapsulated in a package with a name identifying this type. The only restriction we made was that the input data type for a worker must match the data type of the packages sent to this worker. This definition of the data packages made it possible to determine if a worker is able to process data.

For the data packages on a stream the use of data types implied, that on the path of the processing route a data package takes, the input data type of a worker must match the output data type of the worker that sent the data to this input.

For the definition of the language encapsulated in a package, we introduced another abstraction which was the concept of *fingerprints*. The idea behind the fingerprints was to allow a preference on the usable services for one single processing step.

Especially in the speech recognition it is often the case, that a domain specific speech recognition has been trained for one specific use case and should be preferred over another speech recognition for a more general use case. In contrast to that however it was often the case, that just one generic machine translation model was available without a specific adaptation to any domain. This implies that we allowed for non perfect matches on the language fingerprint for situations, where not the exact match was available. However these non perfect matches still required a perfect match on the data type contained in a package. We split the language definition into three components, describing the significance needed for a match.

The basic building blocks of a fingerprint were the three components: `ll` - language, `CC` - country-code and `domain` - the domain this type of language is used in. For the language part `ll` we used the two letter language identification according to *ISO-639-1*, for the country code we used the two letter code according to *ISO-331661*. The domain part was free form allowing the user to describe the content in a meaningful way. A valid fingerprint that was used in the lecture translation project was for example `en-US-CognitiveScience`, describing the language an english speaking person from the US would use when talking about cognitive science.

For the matching algorithm we developed for the fingerprints, a match was possible as soon as the language codes were the same. Besides that, the match was scored according to the following order:

- exact match of fingerprints: `es-MX-lecture es-MX-lecture`
- match of language code and country code `es-MX-lecture es-MX-medical`
- match of language code and domain `es-MX-lecture es-ES-lecture`
- match of language code `es-MX-lecture es-ES-medical`
- no match `es-MX-lecture de-DE-lecture`

With these mechanisms, the *data type*, the *fingerprint* and the *content stream* in place, the processing of stream data was defined and the concept of compatibility between the output of one worker (or more general output of a node) and the input of another worker (or node) was introduced.

The mediator is the process which every worker announced its availability to, and is therefore responsible for finding the paths which were at a later stage spanning the processing graph.

In the following we will use the term *service* for the processing a worker provides. A service in our system is according to the previously made definitions fully described by an input and output fingerprint and the input and output type. In addition to that we also introduced a third component to further specify a service by a service name. With all this in place, a valid example for a service as it was used in our system is:

```
{ 'input-fingerprint': 'en-US-lecture',
  'input-type': 'audio',
  'output-fingerprint': 'en-US',
```

```
'output-type': 'text',
'service': 'ASR'}
```

This described a speech recognition service, which was optimized for handling US english audio speech data as found in typical lecture settings, and it output US english text.

3.2.2. Mediator

As the central unit where all other modules register at, the mediator keeps track of the available workers in a system, the connected clients and the displays which might be interested in receiving processing results. It is also the component identifying which components can be used for fulfilling requests for a certain output. However the mediator is not responsible for the handling of the data being processed on a stream, this was handled by the separate client processes on the server side which handled the connections themselves reducing the load on the mediator process.

In general, the mediator can be seen as the one central directory, that knows about all attached modules, and which got asked, when an attached module needed information about other modules.

The communication between the processes internally of the mediator was done over message passing between the processes, while the communication with the ”outside world” was done over regular *TCP* connections. Those external messages were using *XML* to allow easier inspection and easier development and usage of an open API to allow third party to connect to our service without the need of implementing a binary protocol. Because of the known benefits in scalability, robustness and speed, we wrote this central unit in *Erlang* (AVW+96), a functional programming language which was developed for the purpose of being used in distributed network architectures like the one we developed.

We always focused on the robustness of the framework and therefore, all external connections to this component were handled by separate processes, which were unrelated to each other. This ensured, that in the event of a failing process, other processes would not fail in a chain reaction and algorithms for error recovery and handling were able to kick in, to ensure a safe recovery from such an event, in the optimal case without the user of the framework noticing any anomaly.

Another important part in the development, apart from the handling of the communication, was the fact, that the mediator needed to be very fast, otherwise it would have become the bottleneck of the system. For that reason we developed the mediator in a way,

that it did only a minimum of additional computation. We achieved this, by having the mediator only handle the control messages, while every other data message (those sent between the components were handled by the processes of the involved modules alone) was not looked at in the mediator.

This made the whole processing in our developed framework bandwidth bound, while the cpu was able to handle all sessions without problems. In the end, even under conservative assumptions and pessimistic usage patterns, we determined, that the proposed framework is able to handle 750 simultaneous sessions (with multiple streams and multiple users, as is the average in university lecture settings) using a standard 1 Gbit connection without compression of the audio data. In this experiment, the cpu usage of the used server never rose above 5%. It is important to remember, that in this case a session does not equal the number of users, but there can be multiple hundreds or even thousands of users following one single session. This number is sufficient even for large universities, where it is usually not expected to have more than 750 simultaneous lectures or seminars at the same time.

But even for that case a simple duplication of the mediator with a load balancer in the front was able to handle this and we were able to proof this in a slightly different setup for a handheld translation device for tourists which was using a pruned down version of this mediator at large scale, where we handled multiple thousands of request with this underlying architecture.

The internal state of the mediator was therefore defined by a list of connected and available workers with the information of the services they provide, a list of the current sessions and the corresponding client processes and a list of the attached displays.

The list of the attached displays was necessary, so that new sessions can be announced to all connected displays. The list of the attached client processes and their sessions was also needed, so that new attached displays can be informed over the already existing sessions.

Furthermore the list of the workers was required when the mediator process searched for the best way of generating output of a certain type and language for a stream. The same holds for the situation when a display process was supposed to get the information of all the output types and languages that can be created (without actually requesting a specific one at that moment).

For this purpose the list of workers combined all workers with the same service description in one bucket and when a worker with that exact service description was needed, one of the servers of this collections was used and removed from the list in the mediator.

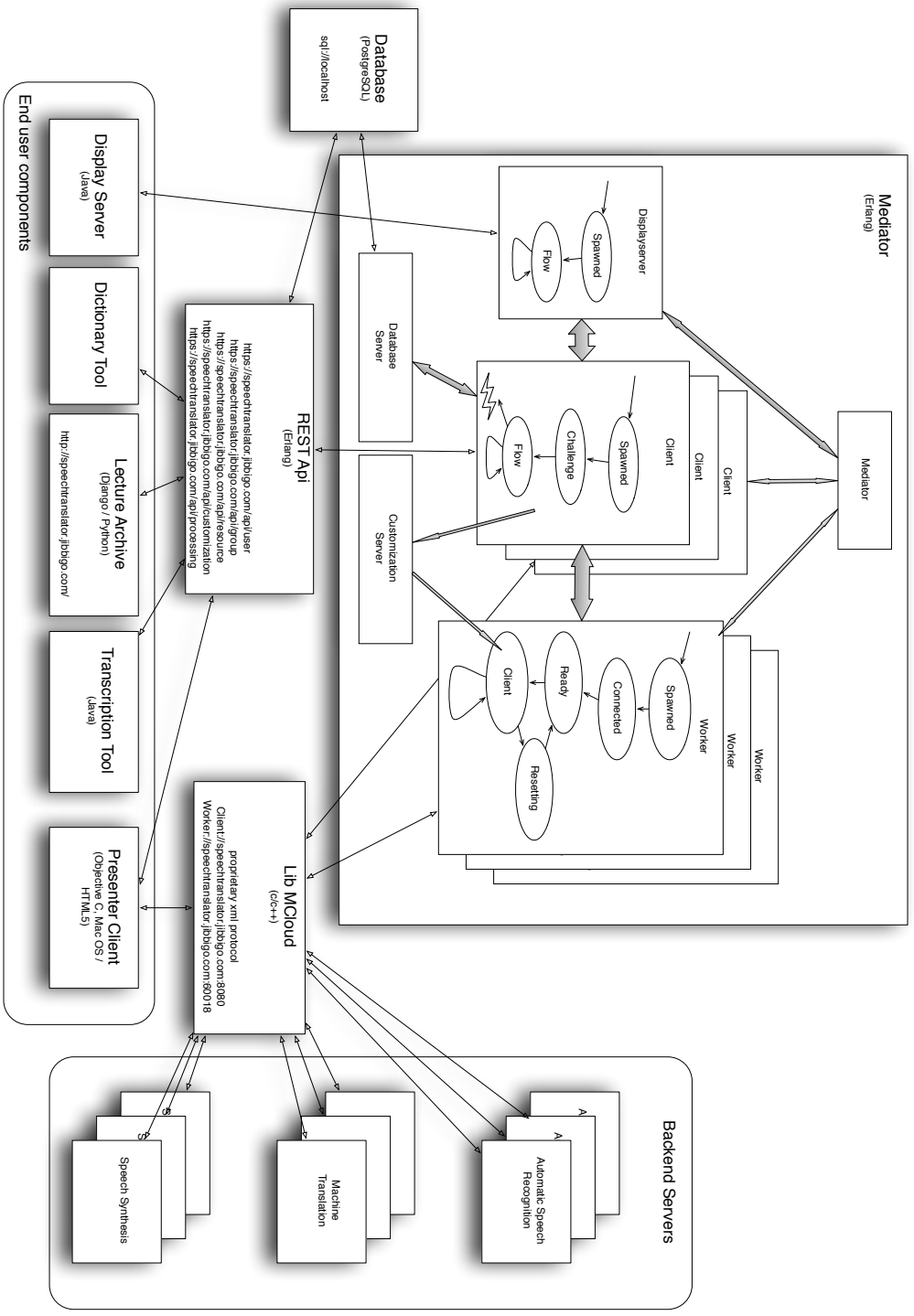

Figure 3.2.: Internal architecture of the developed framework

The selection of the worker can be either arbitrarily from this bucket or based on past operations. The latter one was introduced for efficiency reasons. It was helpful to use the same speech recognition for the same user again. This had the benefit of not always reloading the correct adaptation models and caching, only in situations, where a new user was requesting the service in between.

As described earlier in this chapter the client processes contained a processing graph which described the flow and processing of the data of a certain stream a client creates. This processing graph was stored and evaluated in the client process, while the mediator was responsible for finding the correct and possible routes which can be used for such a graph and sending the resulting paths to the client which was then in charge of creating the processing graph and reserving the modules. A sample diagram for such a graph is shown in 3.3 and shows a client with two input streams and multiple workers providing services on the data of the streams of data.

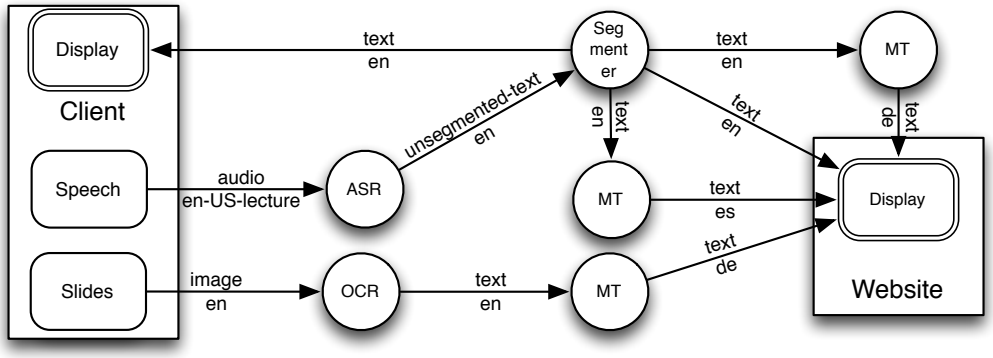

Figure 3.3.: Sample processing graph for a lecture

We supported two different searches in the mediator process. One was a breadth first search, which was able to find all possible outputs that could be generated for a given input, and in addition to that a shortest path search for finding just one specific path to generate output of a certain type and modality for a clients input.

The algorithm for this search is given in Algorithm 2

As seen in the algorithm, we did not require an exact match for the language fingerprints of the connection between two nodes in the graph, but we accepted also non exact matches on a path. The cost of using a path however was dependent on the score of the matches. The better the match between two nodes was, the better is the score of the path and the more likely it was, that this path was chosen for the processing.

```
Data: RStream, RType, RLang
Result: Path of Workers
(Type, Lang, Score) = InputNodes[RStream];
Q = new Priorityqueue();
Path = [];
Q.insert((Type, Lang, Score, Path));
while Q.first().Type != RType
  && Q.first().Lang != RLang
  && extendable(Path) do
    c = Q.first();
    for w in Workers do
        Score = match(w.InLang, c.Lang);
        if w.InType == c.Type && Score>0 then
            NewPath=Path ++ w;
            Score = Score + c.Score + ϵ;
            Q.insert((w.OutType, w.OutLang, Score,
                  NewPath));
        end
    end
end
if Q.first().Type == RType && Q.first().Lang == RLang then
  return Q.first()
end
else
  return failed
end
```

Algorithm 2: Path search for requests

The motivation behind this approach were the following thoughts. First of all, it is a very time consuming task to develop workers for every language pair, and this becomes even more of a problem when additional workers should be developed for every flavor of a language. Most of the times however a specific worker could provide the same service for requests which needed a variation of the language handled by that service. An example for such a use case could be when a speech recognition trained for a University lecture is used for a tourism domain speech recognition task. And even the opposite case is thinkable,

that a general machine translation which was trained without a specific domain in mind is used for a specific task.

In addition to the obvious case where the exact requested service was not available in the system, because there was no worker for handling it, this approach also allowed for more robustness during usage. When a component failed and it's exact replacement was not available, it was possible to replace this worker by another worker, which was not providing the exact same service, but a comparable one.

We also added a node penalty for every worker used on a path to the computation of the path score. This was motivated by the fact, that a shorter path should be preferred over a longer path, if besides the length the cost for both paths would have been the same.

This accounted for two important points in this system. The obvious point is that a longer path made more transformations of the data in the processing chain. Therefore this path tended to introduce more errors since errors tend to be propagated through the following components in speech to speech translation systems.

The other reason for the decision to prefer shorter paths over longer ones was simple efficiency. With a longer path more nodes were used by a session, which on the other hand means that the worker providing the services of these nodes were not available for other sessions and in average more computing power is needed for the processing on this path[1]. This would have made it more difficult for the system to scale - more components would be needed for the same amount of sessions, and it would have been a cost factor, since every node in the processing graph needs computing resources that could be used otherwise.

The search for a path was triggered by a client that requested information how to generate a specific output language fingerprint and type for one of its streams. The mediator returned the result of the above described search to the client which then started to construct the expansion of its current processing graph with this new path.

The path in this case was created by the client in the inverted order, from the last node on the path to the first node on the path. The reason for this direction was, that while the path was being created we didn't want the workers on the path to start computing their services until it is ensured, that their computations will be needed by following nodes. By simply not connecting the new path until it was fully ready, we did not need any flags

[1]This is a generalization, under the assumption that workers have approximately the same cost for computing their results. Of course this is highly dependent on the worker and sometimes processing by multiple workers might be more efficient than using one specific worker. The computing costs can be easily integrated in this architecture, by assigning another descriptor for this value in the node descriptions.

or additional information if a path is already usable. Furthermore this allowed an easy attachment of the new branches in the processing graph to already existing ones, with only little computational overhead (essentially the overhead is to look up if a needed node on the path is already existent in the processing graph).

As we already pointed out above, it is important to note that our architecture did not restrict the types used for the package descriptions. As shown in Figure 3.3 this allowed for example the usage of a segmentation unit after the speech recognition, by ensuring that the output of a segmentation worker won't be interpreted as normal text - in our case, the segmentation unit provided more meta information than simple text, which required additional processing in an additional step. We also had success in the usage of multi pass speech recognition and translation of language pairs where no direct translation was available. In both cases, the graph consisted of chaining multiple engines of the same service, but different types or languages. With our internal search for the shortest path to provide the requested output, this required no additional interaction by any user, following our principle of having an easy to use system.

In addition to the described finding of paths and management of attached workers and clients, the mediator also provided the necessary functions needed to handle cases of failing workers. In those cases the mediator was notified about which worker failed and needs to be replaced. This automatically triggered a search for an alternative path to transform data in the form of the input the worker accepted into the data type and modality that was provided by the output of that worker. With this function in place, it was possible to replace failing workers during the lifetime of a session, usually without users noticing anything at all[1].

3.2.3. Workers

Like already said, the workers represented those modules in the system we developed, which provided data processing services. When an external worker module connected, an internal process was launched, which handled the connection and all interactions with this worker. After the connection was established, the worker immediately announced its services in the format described in 3.2.1.

We allowed for a worker to announce more than one service, however the mediator makes the assumption, that a worker can only serve one of these services at once. If a worker was

[1] Usually such an event caused the loss of some data packages to this worker. Depending on the amount of information in these packages, the problem was more or less visible

able to handle multiple service requests at once, this meant that the worker opened up multiple connections for every service it could provide at the same time. Usually this was done, by having a pool of multiple connections opened by the worker immediately after startup, and then when it hit some low watermark further connections were opened.

A typical example for workers which were able to provide more than one type of service was the machine translation engine used and developed in this thesis. This engine was laid out as being bidirectional, meaning it was able to handle translation request in both directions for the trained language pair. The same thing is possible for a speech recognition, which supports multiple different acoustic or language models and easily switches among them depending on the demand.

On the receiving part, the central point where processes managed the connection to the outside world worker, we used a finite state machine to reflect the status the associated worker was in. In the process for the finite state machine we also managed the socket handling and information to send messages forth and back.

The graph we developed for handling the connections of this finite state machine can be seen in 3.4. As mentioned before, the communication between the worker and the mediator process was reduced to a bare minimum once a client connected, and was restricted to special events only, like disconnects or other errors which needed communication between mediator and worker to allow the mediator to react to such events (making sure, that the disconnected worker was not used in a future processing graph). All other data exchange necessary for the processing on the processing graph was directly handled between the connected client and worker process.

3.2.4. Clients

The client modules were the part of the architecture responsible for initiating the creation of the actual sessions in the system and feeding data into the session's processing graph. Every connected client was handled by its own process, which managed the session and the used processing graph. The external connected client sent the input data of different streams into the graph by sending the data over tcp to the handling process which then in turn emits this data from an input node in its processing graph. In the case of a lecture translation system, that meant a client module was recording audio and fed that data into the systems client process for the further transformations.

The actual processing graph, used to provide the needed output to displays was stored in the client process, allowing the process to handle the sending of all the messages on its

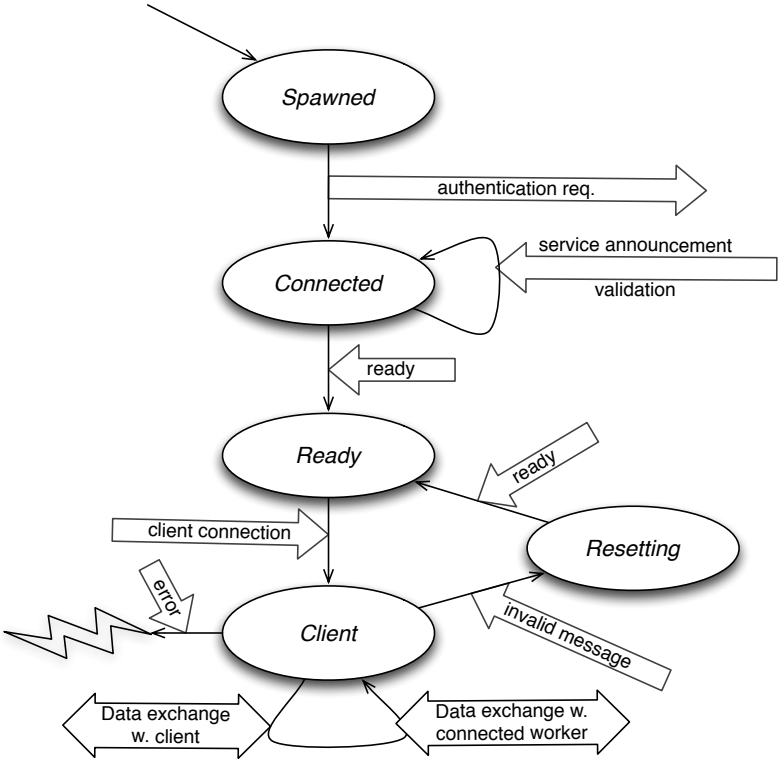

Figure 3.4.: State graph of worker processes

own. This meant, that this process had all information that was needed to determine to which node in the processing graph the output of another node had to be sent for further processing.

We developed two different mechanisms for the creation of the processing graph in a client process. The first method is the one already described above, the search for the shortest path to provide output of a stream. When this method was used, the client sent a message to the mediator to obtain the needed path and then started constructing this path backwards.

In addition to that we also allowed a second method, which was the manual creation of a graph. In this second method, the connected client directly defined the nodes and edges that should be created in the graph and built the graph node by node without any implicit search for the shortest path. While this method was only for experts who knew the details about the connected nodes, this allowed to define exactly which nodes were used when systems needed to be combined. We used this for example when multiple translation engines were used and then a system combination step was following.

The client is also responsible for making a session public. Unless the client does not send a message to the mediator, the session of the client can not be observed by a display process.

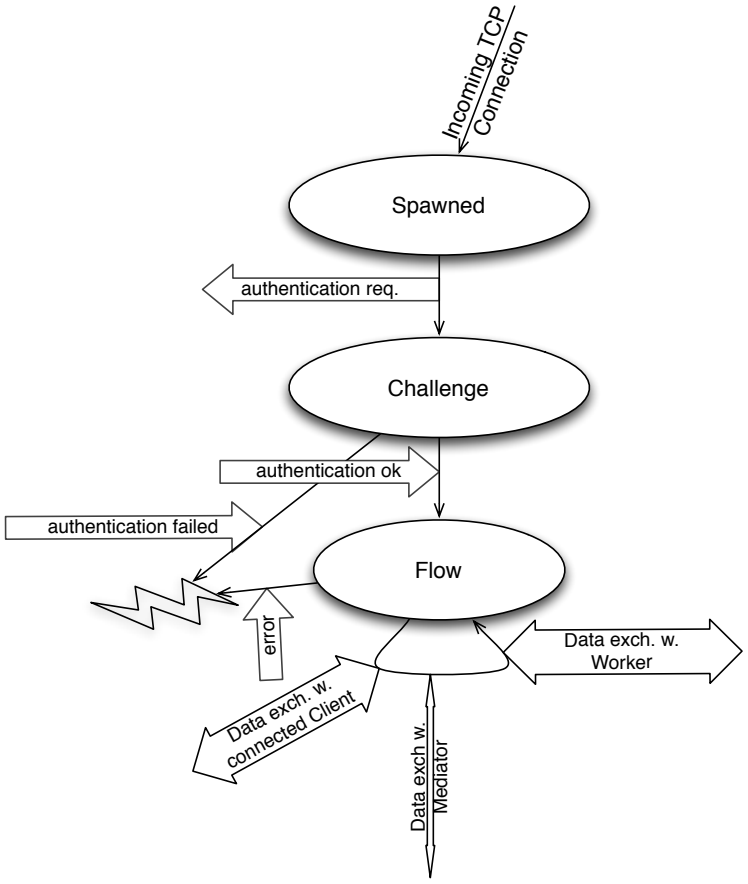

Figure 3.5.: State graph of client processes

As with the workers, the client's state was represented as a finite state machine, shown in Figure 3.5. This graph also shows, that the data exchange between the client and the mediator process was reduced to a bare minimum, to keep the load away from the mediator, preventing it from becoming a bottleneck. All messages including the data are directly exchanged between the worker and client processes.

3.2.5. Display-Server

The *display-server* or short *display* is another part of the developed framework which is directly benefitting the scalability. This component collects all the information of a session and presents it to the users. In our deployment we used a standard user facing web server for the presentation to the users. This allowed the presentation of a single session to hundreds of students, using only a single connection to the central *mediator* unit. This decoupling of the end consumers from the internal network was key to the scalability and allowed the distribution of the generated data to multiple different devices and operating systems with reduced development cost. There was no need to develop multiple tools for different operating systems or platforms, as long as the platform had a

browser. Furthermore this allowed us to build a cascading distribution of the information, which allowed to distribute the generated processed data even further without increased load on the underlying processing framework.

When a client made a session public by a call to the mediator, the displays that were attached to the mediator were made aware of this new session and they were able to attach to this session. When a session was announced to the displays, the mediator also provided the information about all possible outputs for which a path in the processing graph could be built. However no creation of paths was triggered at the moment of the session announcement.

After a display attached to a session, the corresponding client process was informed about the connected display. This generated a new output node in the processing graph representing the display. This was a regular node without generating any output. Since the client was aware of the display process and had a corresponding node in its processing graph, it was possible, that all data exchange between client process and display server was happening between these processes, without involvement of the mediator itself.

Whenever the display needed output of a specific type of a session, it issued a message to the client requesting this service. In the following steps, the client issued a request for a path generating this kind of output to the mediator and then generated this path, connecting the display node to the end of that path. In the end the display was also informed about the successful generation of the path and all future data packages of the requested type were then delivered to the display after the creation of this path.

In addition to this, a display on the other hand was also able to let a client know when it did not need a specific output anymore. This was also done by sending a message to the client. The client then removed all unneeded workers and edges in the processing graph. It is important to note that only workers were removed, that did not have any outgoing connection in the processing graph, because otherwise it meant they were still part of another path still in use.

With these mechanisms in place we were able to ensure that the processing graph was dynamic enough to adjust to the current needs to handle future requests in the same session without wasting resources by preallocating them or holding them even though they are not used anymore.

3.3. Archive

In the previous section we described the underlying system we developed and which could be used for the online speech to speech translation task. This architecture built from the aforementioned modules allowed us to build a dynamic and robust system, which was scalable and easy to extend. With the usage of the displays it was also easy for people to follow a current session, without needing to download special tools or attaching somehow into the system.

The final step in the development of the speech translation architecture for live lecture translation was the development of an archive which allowed the long term storage of the data and easy retrieval of past sessions. This was especially helpful in a lecture translation environment, where students then could use such a system for their learning needs and review past lectures, but it is also important for other situations where such a system might be used, like the usage in a video transcription service or even as a personal translation tool.

Since this architecture was supposed to build the fundament for automatic learning from crowd sourced improvements (not only restricted to machine translation), we tightly integrated the functionality of storage into the architecture itself.

3.3.1. Storage

For achieving these goals, we introduced another module in addition to the modules needed for the live translation, which was directly attached to the mediator. The so called *storage*. The storage module was the component which was able to consume all messages being sent during a session, so that in the future an exact reproduction of this lecture and the processing results created by other components, like the machine translation or speech recognition was possible.

When a message was sent to the storage component, this message was stored with the accompanying information like session id and session name. In addition to that, every message sent got a unique message id, which allowed a reconstruction of the order for the messages that were sent.

For support of online learning of the components, improvement information that was needed for components to update their models was also encoded in normal data messages. This allowed for example clients or workers to insert information for future improvements into the system.

The background database which stored the information only needed a wrapper for implementing the storage module. We made no assumption as to what kind of database was used. All access to stored data was handled by the storage component which then abstracted from the underlying data storage. This allowed easy migration to different database systems, even NoSQL (Lea10) databases can be used as long as the storage component was able to implement the protocol. This made it easy to integrate the framework into existing environments.

With the introduction of a storage component, which allowed the retrieval of past data we also developed mechanisms to protect access to the data.

As described earlier a client is able to announce a session to the public, but can also remain private. This required however that a session that was private during creation is not accessible by the public once it is stored. For this reason we developed a user accounting mechanism and made the access to the data bound to users and groups. Still this is not user friendly for the use case of a lecture translation environment, where hundreds of students might want to access the data without the need of creating a user account beforehand.

We solved this problem by the introduction of a hash for sessions which provide direct access to the session. These hashes could be managed by the owner of a session and they provided access to the session as long as the owner of the session decided to keep this open, similar to url shortener services.

The introduction of the concept of users also helped us to identify the person who provided a correction. This is very helpful for the crowd sourcing approach to collect improvements to the system. In the situation of crowd sourcing mixed quality feedback was expected and sometimes users try to sabotage systems. For that reason it was important for us to identify all corrections or messages made by one user, so that those messages can be removed in case of a saboteur. In addition we hoped that the possibility of identifying a user who provides translations made users feel accountable and increased the overall feedback quality[1].

3.3.2. Customization Manager

Closely related to the general storage for storing all data, we added another abstraction layer on top of the storage component, the so called *customization manager*. This was a very simple task which had the sole task of handling and returning the data that was necessary for improvements. Whenever a correction was made by a user this correction

[1] this was something that was not possible to evaluate in this thesis and is left open to future work

was sent to the customization manager and the customization manager took care of storing it indexed in the background database. The indexing was done over the language pairs and the type of feedback, so that it was possible to retrieve the important information in a later step.

The customization manager was also the component which took care of serving the already made corrections back to other worker components, which then were able to improve their models. Since the data was indexed by the type of feedback and the used languages, a worker was able to request the feedback specific for the needed improvements.

This however could easily end up in sending multiple thousands of corrections every time a worker needed to update its models. To prevent this, we also introduced the concept of versions, which was a simple unique number counting the up the number of corrections stored. This allowed the worker to only ask for those corrections that were not seen so far.

3.3.3. User Provided Corrections

In our main goal of developing a self improving machine translation system, we wanted to learn from users providing feedback of correct translations. In our situation that meant the user saw a wrong translation generated by our system and based on this the user decided to correct this wrong translation and input a better one. This in turn allowed our system to automatically update its models, so that its quality improved based on this feedback.

A problem with user generated translations however is the problem that users are able to generate free translations which are correct, but not very literal and provide more a translation of an interpretation. These types of translations impose a lot of problems to machine translation, because they are very hard to learn. Therefore more literal translations are preferred for training a translation system.

This means that in the case of a correction our system could learn better statistics for its model when the translator tries to fix a generated translation compared to a translator generating a new translation from scratch. However the speed of translators is significantly higher when they generate a translation from scratch compared to fixing an existing translation. The reason for this is the higher cognitive load a translator faces when correcting a translation. In addition to the task of just translating a sentence, when fixing an existing translation the translator also has to understand the wrong translation and identify the error made in this translation. Then the translator is forced to fit his original translation into the corset of the wrong translation. This introduces a huge task to the translator who normally would have noted down the first translation that comes to mind. Significant

work has been done in (AOMSC10) and is also the basis of what we did. We integrated mechanisms into our machine translation system which enabled us to extend a translation based on a given prefix. With this mechanism we were able to reduce the load of the translator to only checking if the proposed completion of the sentence based on the given start of translation is correct. If that was the case, the translator was able to accept this proposed translation without the need for further input. The evaluation was done on the crowd sourced data from the TED lectures.

3.3.4. Selection of Translations and Transcripts

With the introduction of user corrections another problem had to be solved as well. That is the task of providing the best transcript or translation for a segment based on user feedbacks. For this task we assumed that there was a set $T = t_{i_1,j_1}, t_{i_2,j_2}, \ldots, t_{i_n,j_n}$ of n user provided corrections for a segment $S = s^1 \ldots s^m$ covering the words i_r to i_t. These corrections were a list of m words which are used instead of the original words

$s^i \ldots s^j$: $t_{i,j} = t^1_{i,j} \cdot \ldots \cdot t^m_{i,j}$.

The task we had to solve then, was to build a new hypothesis, based on the feedback all users had provided for this segment. One important factor for this is, that we can do better, the more exact we know the boundaries of the words for a user correction. In our notation from above, this means, that we have to find an optimal set of user corrections covering the sentence, resulting in a final transcript for this segment: $\hat{S} = \hat{t_{1,i_1}} t_{i_1} \hat{t_{1,i_2}} t_{i_2} \hat{t_{1,i_3}} \ldots t_{i_u} \hat{t_{1,m}}$

To not face the problem of non existing corrections for some of the words, for every subsequence of words $s^i \ldots s^j$ we also introduced the correction $t_{i,j} = s^i \ldots s^j$ as an alternative in the set of corrections.

In the search for the final segment based on the user corrections, we required that used segments do not overlap. This might sound like a strong restriction leaving out a lot of helpful information. We decided for this however because of the problems when allowing this for the correction of translations. While in the case of speech recognition, a small correction within a larger correction should preserve the meaning (assuming it is a valid correction), this is not necessarily the case for the correction of translations. In the case of corrections for a translation, the problem is that usually in translation also a reordering happens, so that a different translation for the same subsequence in a sentence might cover words in a different order. This makes the words covered by a translation of a sub part unpredictable without an additional component providing translation information.

However with this restriction in place the search for the best final transcript or translation based on the obtained user feedback can be solved as a search for a path in a graph, where the edges are the corrections and the nodes are the word boundaries.

The next step in this solution was to integrate the information of agreement between the corrections. When the same correction was seen multiple times, it should have a higher probability to be used than when it was generated just once.

On the other hand it is also important to respect the fact, that there might be saboteurs trying to manipulate the system. Not every user is equally fluent in the task of translation or sometimes simply errors happen. For this reason we also developed a mechanism for the credibility of a correction. All users start with the same credibility, and once a correction made by a user is verified by another user, the credibility of the creator rises, while in the same situation other users correcting in different ways, the credibility of the creator sinks. In addition to this, as part of the automatic improvements for machine translation we also integrated a mechanism which provides a score for the provided correction. Based on these scores it was possible to select a combination of corrections for a segment and present this to the user.

3.4. APIs for Third Party Components

As one of our requirements we identified that we wanted the developed architecture to be easy to extend. For that reason we developed programming interfaces for developers to connect to the architecture and provide additional services or use the system in new ways. These APIs were also the only way to connect as a *client*, *worker* or *display* to the architecture. This decision resulted in well tested APIs which were put into use not only in the lecture translation project of the University of Karlsruhe, but also in the EU-Bridge project. We developed two different possibilities to connect to our service. One was using the raw XML based communication scheme over regular TCP connections, for which we also provided a reference implementation in *c*. The other one we developed was a *REST* interface (FT02) which was provided as a higher level API providing a more abstract interface to the underlying services. For the REST Api we decided against support for worker connections, because of the initial connection overhead and the more difficult debugging of network problems for long living HTTP connections. The existence of a REST based client however proved the possibility of this and the addition might be a future task when a higher demand for this functionality exists.

The REST API encapsulated multiple XML messages on the server side, to achieve typical

tasks we identified in the usage of the lecture translation architecture. In our final deployment of the lecture archive (see Chapter 6.), we exclusively used the developed REST API in the web-server for all tasks interacting with the lecture translation architecture.

For the real time translation service we used exclusively the *c* library for performance reasons because of the slightly smaller communication protocol compared to the overhead induced in a RESTful interface.

As a proof of concept however we also developed a complete standalone client module that used the REST interface and interacted with the presenter in a HTML5 compatible browser.

3.5. Conclusion

This chapter presented our developed distributed natural language processing system and architecture which was also used for the final lecture translation system that was developed as part of this thesis. We started with the identification of the requirements for such a system and used this as a basis for the development of the architecture. In the second part of this chapter we then described the design and components needed for the simultaneous translation in this architectures with a description of the internal communication and the external interfaces.

The third part of this chapter focused on the persistent storage of the data and the backend components needed for crowd sourcing efforts within this setting.

We concluded this chapter with a description of the available APIs for developers to connect components to this architecture adding more services to the platform.

4. Online Improvement of Statistical Machine Translation Systems

This chapter describes the mechanisms we developed to improve a baseline system in an incremental manner without the need of a time consuming full training. We built a framework which was able to improve over a pre-trained system online, allowing updates to the models at runtime. To our knowledge with the deployment of the lecture translation system at the Karlsruhe Institute of Technology, this is the first such statistical machine translation system put into use in a production environment. Our system extended the work presented in (OMGVC10), showing that automatic improvement for statistical machine translation can be done in an efficient manner allowing it to be used in realtime speech translation situations. We developed mechanisms to flag potentially harmful translations which should not be included in the automatic improvement. And finally we developed a way to support users who are willing to improve the translation quality with their feedback. We were able to show that our developed system was able to improve over time when crowd sourced translations are used.

4.1. Batch Training for Improving Statistical Machine Translation Systems

The standard approach for training statistical machine translation systems is a batch training of the whole system given all the available training data at training time. If new data is added to the pool of training data, a new iteration of training has to be done. For statistical phrase based machine translation this training usually boils down to a training iteration containing at least the following steps.

1. Cleaning and preprocessing the training data.
2. Building a word alignment and lexicon based on the parallel training corpus.
3. Extracting phrase pairs using the word alignment and lexical information.
4. Training language model(s) using the monolingual data.
5. Extracting additional models depending on the actually used machine translation system
6. Optimizing scaling factors for model weights in the log-linear model.

For this process many tools are publicly available and to some extent also used in this thesis. For the task of doing such a batch training we used the commonly used GIZA++ toolkit (ONJN03) to create a word alignment with the extensions introduced by (GV08) which allowed us to run the word alignment in parallel. In the following steps these alignments form the foundation of the training scripts. To be comparable to other groups, we used the training scripts provided by the University of Edinburgh (KHB+07) in the Moses statistical machine translation toolkit, to extract the phrases, assign scores to them and optimize the scaling factors. For reproducibility the language model training was done using the SRI language modeling toolkit (S+02) which is freely available for non commercial research purposes.

In a last step these models were converted and optimized for our own machine translation system (ZV07). This included a merging into single scored translation models and language models. Since this step involves the combination of multiple translation model scores into one single score, the previously optimization using the multi score phrase table was necessary, to keep the correct relations when the scores were merged. In this step also the scaling factors for the model weights are re-estimated to optimize their scaling with regard to the changed decoding process (Wu97) used in our machine translation system compared to the beam search decoding (Och02) used in (KHB+07). This re-estimation is necessary, because of the huge difference in the reordering between those two decoding algorithms. While in the stack based decoding the reordering is an additional step when expanding a translation hypothesis, the translation with an ITG based decoder directly takes reordering into account by the grammar rules which define the decoding process.

The drawback with this batch training approach is the time used for one training iteration, which can easily exceed multiple hours until a full training is finished[1]. This makes it

[1]One might argue, that the conversion into our own machine translation system is an unnecessary step that introduced additional overhead. The conversion step however only takes a small fraction of time

unfeasible to perform such a training for every new sentence pair available to learn from. Even for a larger number of additional training sentences up to 10.000 a retraining does not make sense, because of the little influence on the translation quality given the long training time. Only for building systems towards the usage in a very limited domain, it might make sense to use such a training for an additional 100 sentences.

This results in a typical setup where additional data is collected until a certain threshold is reached and then a full retraining is started in the background, while the original system continues to operate until the retraining is finished. This process is easy to automate but the unpredictable jumps in quality only at certain times are hard to understand for users not familiar with the topic of statistical machine translation. And it leads to frustration for the users, since these quality gains are not happening the moment a user gives an improved translation as a correction. This is leaving the user with the impression, that the system does not improve after giving feedback.

4.2. Incremental Training for Continuous Improvements

The batch training method and its inherent problem of long training time lead us to think about how the underlying models of a machine translation system could be improved without this huge time overhead. It is obvious, that even with the future development in hardware, a full training from scratch is not possible within such a short time that it could happen between two translation requests. Therefore we looked at the possibility of doing an incremental approach, using the statistics of a full trained system as a foundation to build a base system. Then this base system could be adapted with additional data without a full retraining, but with minimal model changes instead.

With this idea we were able to provide model updates in a fast and efficient manner, allowing updated and improved translation systems even between two translation requests.

In this chapter we describe how such an incremental update of the models based on single sentence pairs was achieved. This section is closely related to the work of (OMGVC10) where they described similar techniques for interactive machine translation (IMT) which is the use case of computer aided translation. We also extend over past research of (SLZ12), which looked at binary feedback to improve the translation quality as another way of improving the quality of a machine translation system.

compared to the full training run before, so that the following arguments still hold and this step can be seen as any additional work needed to put a system into production

In our setting however we looked not only on the improvement within a single text to translate, but we also focused on the ability to improve the overall performance of the system for future unrelated and unseen texts. In addition to that our implementation allowed simultaneous updates of both translation directions, source to target and target to source at the same time. This was very helpful in our use case where we dealt with multiple translation directions in the lecture translation environment (an increasing number of lectures at the KIT is presented in English, while German still remains the major language). Furthermore our previously described distributed environment made it easy to distribute feedback to multiple components, allowing for example every component which translated into one of the two languages part of the correction to at least improve their language models.

4.2.1. Single Sentence Corrections

We start our description of how we learn from user feedback with the description of a single sentence improvement (in this case it is even more a single *utterance* which is corrected). The correction of a single sentence then builds the foundation of all future improvements, which in our framework are nothing more than multiple single utterances.

Our base assumption was, that in a crowd sourced environment, we have a user who provided our incremental learning machine translation system with an input sentence f and the correct translation of this into the target language e.

Following the above made description of the batch training, we identified the steps that were needed for an improvement based on a single sentence. In a first step the input data had to be cleaned and preprocessed. This preprocessing was necessary, because the user who provided feedback does not know anything about the underlying mechanisms and therefore was neither able to provide the sentence pair for learning in the correct format nor should it be the task of the helpful user because that would only increase the overhead of providing feedback and therefore probably reduce the overall number of users willing to improve the system.

For the preprocessing we integrated our preprocessing and cleaning steps into the online system and made it callable for every single sentence pair.

For the improvement of the translation output, we then identified the components where we needed to modify the statistics to improve the overall translation quality. The components where this was needed were:

1. *Lexicon*

2. *Phrase Table*

3. *Language Model*

These components are crucial for a (statistical) machine translation system and need different strategies for updating. When we implemented our update mechanisms we tried to stay as close as possible to the original training results. The ultimate goal was that there is no or only little difference between trainings performed incrementally and the original batch training, but without the huge amount of training time.

The following sections describe the mechanisms we developed to achieve this goal.

4.2.2. Lexicon and Phrase Table Updates Based on Single Sentence Word Alignment

The word alignment is typically one of the first steps in the training of a machine translation system. The word alignment describes a mapping between the source words of a sentence to the corresponding words in a translation of this sentence.

This is typically an n to m mapping and not restricted to the first intuitive assumption of a 1 to 1 mapping. This is caused by the fact that phrases are typical building blocks of languages and they translate into another, like in the example

```
I would like to take a shower -> Ich will duschen
```

where only one single point is a 1:1 mapping (”I <-> Ich”). Computing the word alignment is crucial for both, the *lexicon* and the *phrase table*, since both components rely on these mappings of words to determine probabilities and building blocks for the generation of a translation. To determine a word alignment for an input sentence pair, one main idea is that the probability for a translation of a foreign sentence f_1^J into an english sentence e_1^I can be rewritten as:

$$Pr(f_1^J|e_1^I) = \sum_{a_1^J} Pr(f_1^J, a_1^J|e_1^I) \tag{4.1}$$

Here the probability of the overall translation for a sentence is the sum over all possible word to word alignments and their translation probabilities.

When computing the alignment the main interest is not to find the exact probability, but just the $\hat{a_1^J}$ which maximizes the probability among all other alignments, the so called *Viterbi alignment*(ONJN03).

$$\widehat{a_1^J} = \arg\max_{a_1^J} Pr(f_1^J, a_1^J|e_1^I) \tag{4.2}$$

For the computation of the word alignment based probabilities used in this formula, many different approaches were introduced and are still used. The most popular ones are as the moment of writing the so called IBM Models 1 through 5 introduced in (BPPM93) and the HMM alignment described in (VNT96).

- Model 1, which uses a uniform distribution $p(i|j,I,J) = 1/(I+1)$:

$$Pr(f_1^J, a_1^J|e_1^I) = \frac{p(J|I)}{(I+1)^J} \cdot \prod_{j=1}^{J} p(f_j|e_{a_j}) \tag{4.3}$$

  Model 1 has the property that it is easy to compute and the decomposition of the probabilities allows a greedy determination of the optimal alignment for this model when the probabilities are known.

- Model 2 included additional information, by looking at the absolute length of sentences and modeling the alignments based on this.

$$Pr(f_1^J, a_1^J|e_1^I) = p(J|I) \cdot \prod_{j=1}^{J} [p(a_j|j,I,J) \cdot p(f_j|e_{a_j})] \tag{4.4}$$

- With the IBM Model 3 the concept of fertility was introduced. This concept models the fact, that usually a different amount of words has to be used in translation of a sentence, compared to the original sentence.

$$Pr(f_1^J, a_1^J|e_1^I) = p(B_0|B_1,\ldots,B_l) \cdot \prod_{i=1}^{l} p(B_i|B_{i-1},e_i) \cdot \prod_{i=0}^{l} \prod_{j\in B_i} p(f_j|e_i) \tag{4.5}$$

  where B_i is the set of words aligned to e_i, $\phi_i = |B_i|$ and

$$p(B_i|B_{i-1},e_i) = p(\phi_i|e_i)\phi_i! \prod_{j\in B_i} p(j|i,m) \tag{4.6}$$

  In this model the dependency of B_i on previous aligned words is neglected. The probability for the fertility is modeled as

$$p(B_0|B_1,\ldots,B_l) = \binom{m-\phi_0}{\phi_0} p(1-p_0)^{m-2\phi_0} p_1^{\phi_0} \frac{1}{\phi_0!} \tag{4.7}$$

- Model 4 also attributes for the dependency on previously aligned sets by two HMMs in the computation of the probability $p(B_i|B_{i-1},e_i)$

- Model 5 corrects the problem, which arises in Model 3 and 4 of assigning probability mass to non source language sequences, resulting in a sum over all probabilities being not equal 1. The problem is that the models allow the assignment of multiple output words to the same position in the translation with a positive probability, thus wasting probability mass and making the sum of all possible translation probabilities smaller than 1.

- The so called HMM model introduced in (VNT96) is another alignment model which is often used as an additional step. The underlying observation motivating the use of the HMM alignment is the fact, that the positions of aligned words in the target sentence is not totally independent of the previous word in the target sentence. This can be expressed in a probability $p(a_j|a_{j-1}, I)$ where the alignment for position j is dependent on the alignment of position $j-1$ and the length I of the English sentence. In combination with a simplifying first order dependency:

$$Pr(f_j, a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = p(f_j, a_j|a_{j-1}, e_1^I) \quad (4.8)$$

$$= p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j}) \quad (4.9)$$

  This results in the following word alignment probability for the HMM alignment:

$$Pr(f_1^J|e_1^I) = \sum_{a_1^J} \prod_{j=1}^{J} (p(a_j|a_{j-1}, I) \cdot p(f_j|e_{a_j})) \quad (4.10)$$

As can be seen from the formula of Model 1 it is only dependent on the lexical probability ignoring the position of the words in the sentences, while model 2 also incorporates the length of the sentences. The problem with the higher order Models (above Model 2) is however that there is no efficient algorithm for finding the Viterbi alignments because of the sum over combinations.

For better understanding of the alignment we printed the differences between a model 1 and an HMM model alignments in the Figure 4.1 and 4.2. As it is obvious from the figure, the inclusion of additional word order information in the HMM alignment is able to generate better alignments compared to the IBM 1 alignment. In addition to this better alignment quality, the HMM alignment also has the advantage, that efficient algorithms for the computation of the alignment are known.

Because of this and our goal of having a fast, but also reliable word alignment available for the new training examples, we chose the HMM alignment as basis for the updates of the lexical and phrase models.

With only the jump probabilities (the probabilities of a new alignment position given the preceding position) in addition to the lexical probabilities used, this alignment also has only little memory overhead compared to the IBM 1 alignments lexical information. This small memory footprint was another important factor why we chose to use this model because it also meant that we were able to store the required data efficiently. A shortcoming with every of the models described, is that these models allow source words to be aligned with

at most one target word. This is especially bad given the already stated observation, that typically the word alignment maps multiple source words to multiple target words or vice versa. This happens automatically when a language has a specialized word which has no counterpart in the other language, where then typically a multiword expression is used to describe the same phenomenon (e.g. the German word *Futterneid* which means being jealous on someone else's food).

To solve this deficiency, (ONJN03) introduced the idea to symmetrize alignments after computing them independently for both directions, source to target and target to source, and (AMCb$^+$05) extended over this with additional rules. The naming we use in this thesis follows the naming of the *refined methods* used in (AMCb$^+$05) and are as follows:

- *intersection* means that after computing the alignments in both directions for a sentence, the intersection between the alignments is built.
- *grow-diag* alignment starts with the intersection and adds points to the refined alignment which are in either the source to target or the target to source alignment and neighbor alignment points in the intersection. In this case the term neighboring means that they are horizontal, vertical or diagonal neighbors to existing alignment points
- *grow-diag-final* alignment extends the *grow-diag* alignment by adding those points from the union of the original alignments which are not neighbors to existing alignments and either the corresponding source word or the corresponding target word is not aligned.
- *grow-diag-final-and* is similar to the refinement *grow-diag-final* but with the difference that both words, source and target are not in the grow-diag alignment.

When we looked into the problem of reestimating alignments for new sentences which were not part of a big background corpus but real world usage of an existing system, it was obvious that we had to deal with words which were not seen before, so called *out of vocabulary* words.

The problem with these unknown words is, that the lexical probability for such words - $p(e|f)$ - is unknown. This results in the fact, that all words in the other language essentially share the same probability of being generated by such an unknown word.

In the usual training procedure of statistical machine translation systems, this is no problem. The typical training approach starts with an initial lexical probability model which is

continuously improved over time in *expectation maximization* manner. With initial lexical probabilities, an initial word alignment over the whole training corpus is computed.

In the next step, based on this new word alignment a new lexicon is computed with updated lexical probabilities and another word alignment is computed. Usually this is done over multiple steps using different models of word alignment resulting in reliable lexical probabilities and word alignments (ONJN03).

For our implementation of the incremental training however this was not an option, since we could not do multiple iterations over the whole corpus to estimate the probabilities of an unknown word.

However one observation when we look at incremental training is, that the already existing models are based on the assumption, that we were able to estimate reliable statistics for this data. As a result we did not reestimate these statistics when a single new sentence was added and we were able to approximate the updated lexical probabilities in a different way. Normally the lexical probability is computed as:

$$p(e|f) = \frac{c(e,f)}{c(f)} \tag{4.11}$$

$$p(f|e) = \frac{c(e,f)}{c(e)} \tag{4.12}$$

where the $c(e,f)$ and $c(f)$ represent the number of alignments between a word f and e and the occurrences of word f alone. With a new sentence and an unknown word, we did not have a reliable estimate for $c(e,f)$, therefore we used the assumption, that $c(e,f)$ is equal to one for every pair in this new sentence, and $c(f)$ is equal to the number of words on the other side $|e|$ plus a discounting of ϵ. This is equal to the assumption that the probability for f being translated into one of the words e is evenly distributed among all target words e while we leave a discounting for probably unseen translations of f. At this point it comes into account, that we decided to use the HMM based word alignment instead of the much simpler and easier to compute IBM 1 word alignment. With the IBM 1 alignment, using only the maximum probability, the unknown word could be aligned to all other words, which do not have a better candidate word for translation.

This becomes a problem, since the position among other alignments is not regarded at all, and other word pairs in the sentence for which we had other positive training examples in the rest of the training corpus might have low (but still correct) probabilities and therefore translation, that the new unknown word might be preferred in the alignment, just because rare events are overestimated.

However the HMM based word alignment we used is able to deal better with this problem by the additional information of the jump probabilities used, keeping the alignment in a shape consistent with previously seen alignments.

This effect is illustrated in Figure 4.1 and 4.2. While the latter alignment provides a good estimate for the alignment of the unknown word ”rearview” and ”erkenntnis”, the IBM 1 based alignment distributes these words among all of the possible alignments. Based on the final alignment computed by the hmm alignment it is then possible to give a better estimate of the translation probabilities for this word, while we account for the fact, that the alignment might have been wrong by still assigning a small probability portion to those words which have not been aligned to the out of vocabulary word. In contrast to the work presented in (OMGVC10) we did not implement the EM-training (ONJN03) for the re-estimation of the HMM parameters like the jump probability and the lexical probabilities.

We did this for several reasons. First of all our online system had a much higher availability requirement: one translation engine might serve multiple users. This requires a very high availability of the components and implies very short busy times, where every millisecond saved is important.

But more important is the second reason. In the usual word alignment phase several iterations of different models are used to incrementally improve the alignment and modify the lexical probabilities. This is done over the whole corpus. In our incremental framework this iteration is not done on the whole corpus and we compute only one single additional HMM based alignment. Therefor this new alignment is not as trustworthy as the previous alignment computed in the original baseline training phase and the statistics are not as reliable. For this reason these statistics should not be used for updating the model parameters of the word alignment model.

This however does not hold for the word occurrences and the word co-occurrence counts. After we computed the final word alignment based on the previous description, we update the occurrence counts and the co-occurrence counts based on the final *grow-diag-final* alignment as mentioned before.

This also differentiates us from the work described in (GLQH11). In contrast to their work, our work used the incremental word alignments and continued from here as a starting point for updating the other models used in our translation framework. Not only resulting in faster alignment creation, but in an overall much faster turn around time for building improved models, while also dealing with the problems of user feedback.

4.2.3. Lexical Probabilities

Since we use the lexical probabilities not only for the alignment itself, but also in two scores for the phrase pairs, we kept in contrast to (OMGVC10) an additional incremental model for the lexical probabilities.

Based on the word alignment that was computed before it was then possible to update the lexical counts of the words $c(e)$ and the co-occurrences $c(e, f)$ to re-compute the lexical probability of the words. The modification of the counts and probabilities is straight forward when the absolute word counts are available at runtime. For this purpose, we developed our incremental model that way, that it stores the words and all their corresponding co-occurrence counts. It was crucial for this model to allow a fast lookup and change of values, while leaving a relatively small memory footprint. This is why we decided for a prefix tree data structure for fast retrieval and efficient storage. In this tree we stored for each source and target word how often they occurred in the training data. In addition to that we also stored how often source and target words were aligned together. With these two numbers it was possible to compute an estimate of the word translation probability $p(f|e)$ which was needed for the future computations. The model we developed also included discounting to deal with rare events, like words only seen once or twice, which would have been overestimated otherwise. This was especially in our situation of learning from lecture style data to expect and for this reason we had to develop mechanisms to deal with this problem when estimating the translation probabilities. For the discounting of the lexical probabilities we used the Good Turing discounting(CG96).

4.2.4. Updating Translation Model

After the refined word alignment for a newly provided bilingual sentence pair was computed and the updated lexical probabilities had been estimated, the next step was to identify meaningful phrases into which the translation of the sentence could be decomposed. Following (AMCb$^+$05) we extracted all phrases that were consistent with the word alignment computed in the first step. Consistent means that no word within a phrase pair

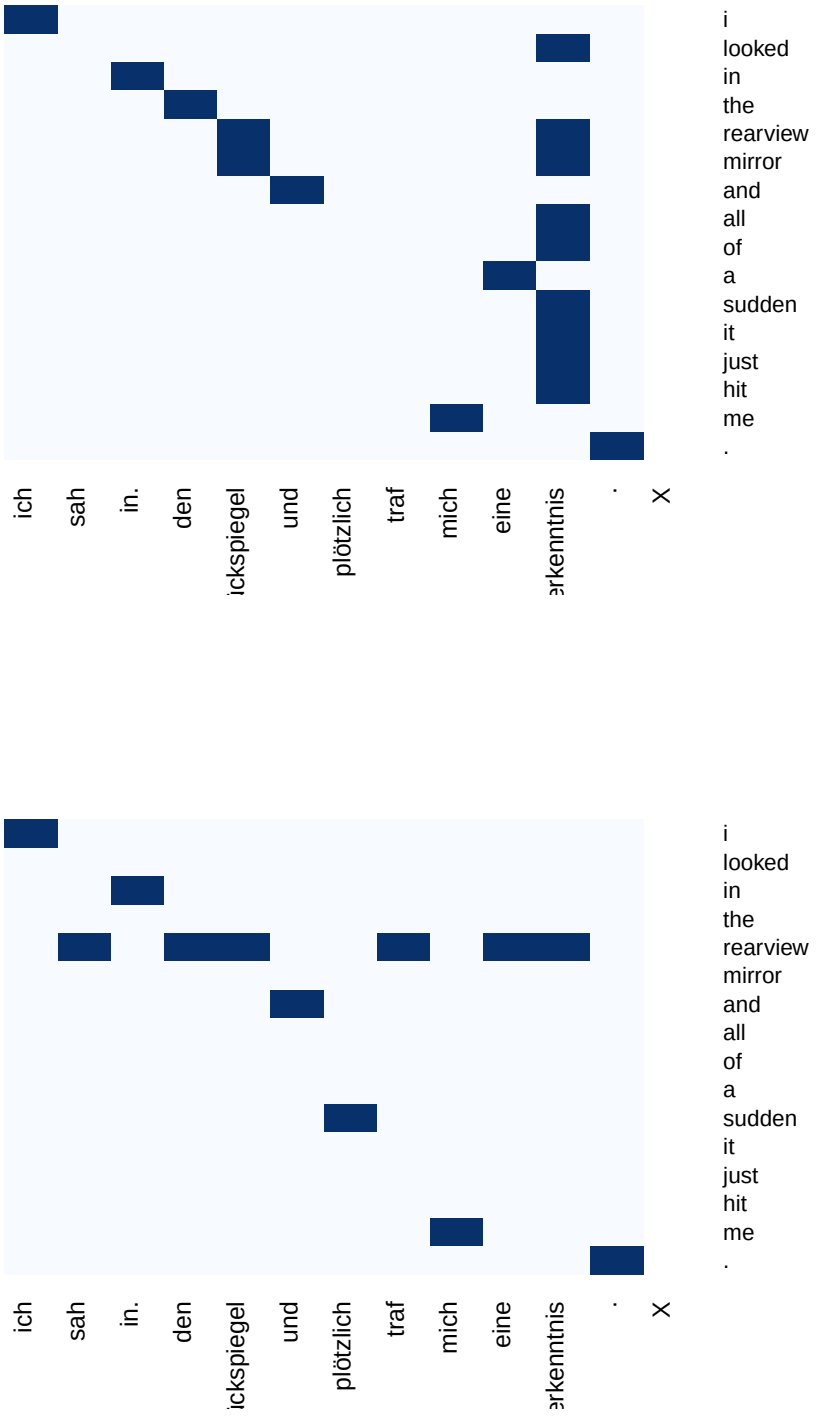

Figure 4.1.: IBM alignments computed from English to German and from German to English

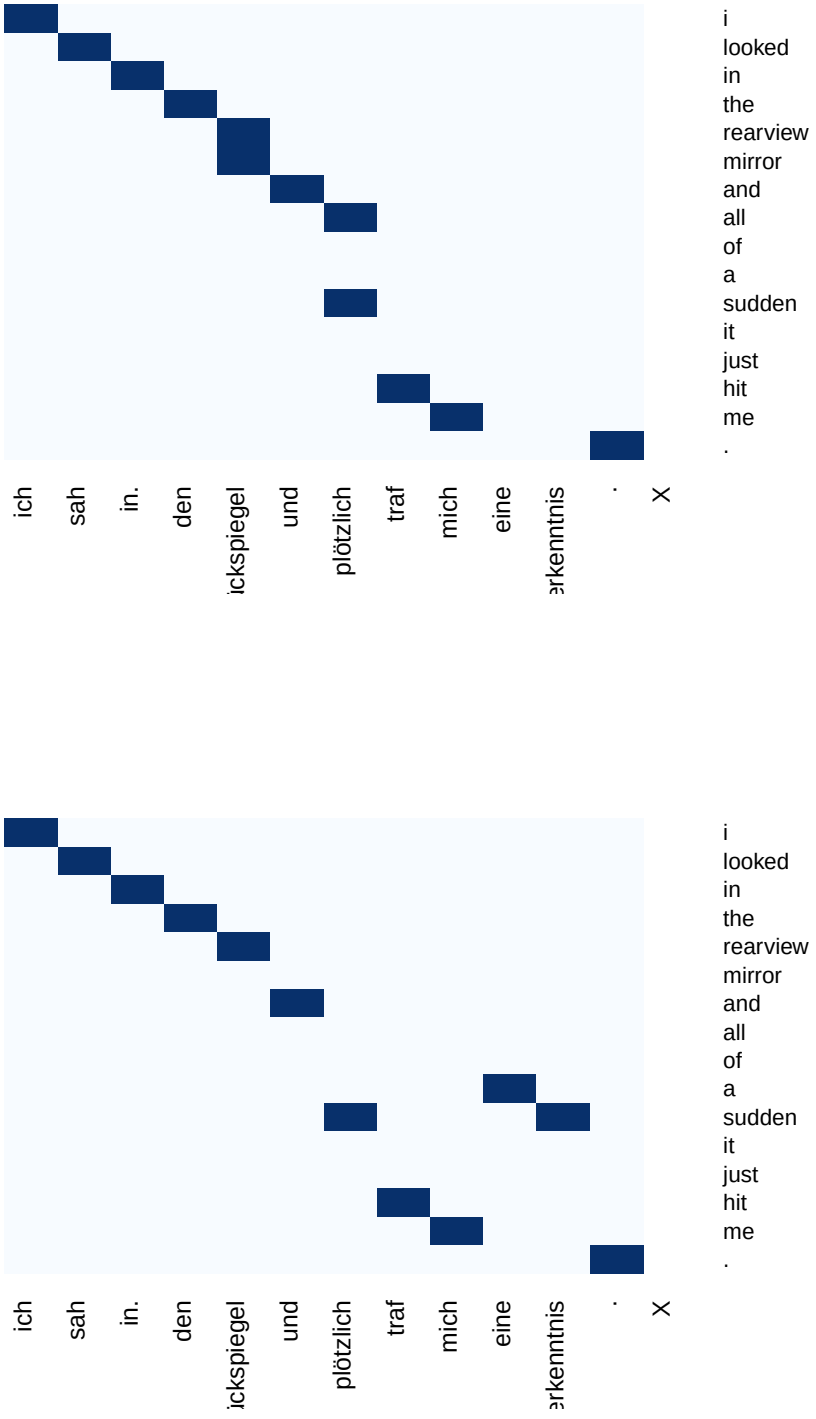

Figure 4.2.: HMM alignments computed from English to German and from German to English

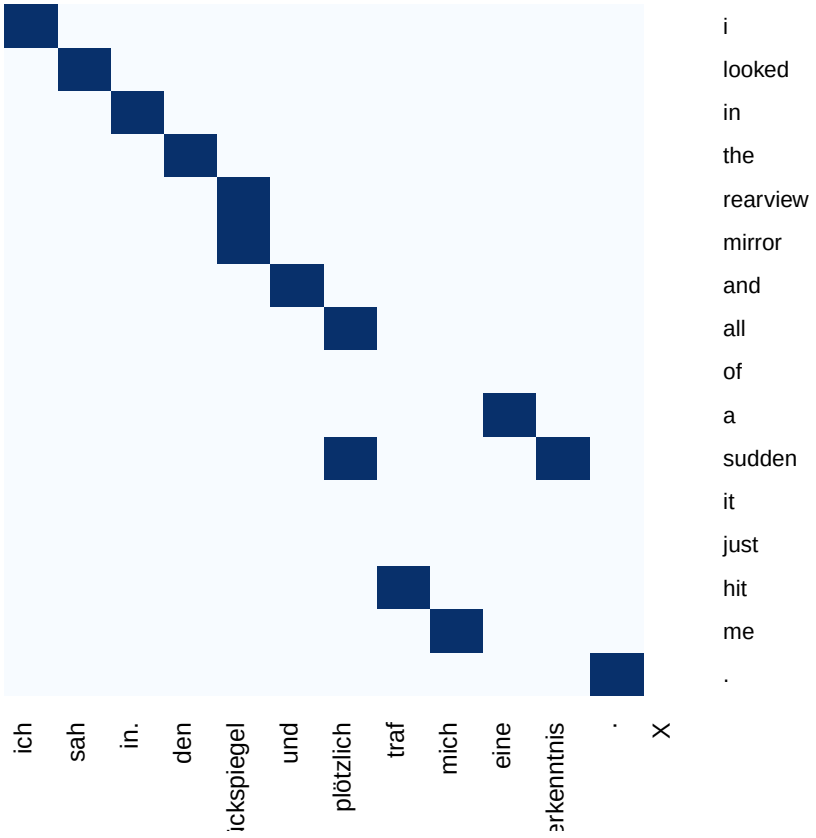

Figure 4.3.: Grow-Diag-Final alignment

was allowed to be aligned to a word outside the phrase pair and phrase pairs must cover at least one alignment point.

This generates a list of bilingual phrase pairs for the new sentence into which the translation process can be split according to the underlying alignment model. In contrast to the batch training, our new incremental method did not collect the phrase pairs over the whole training corpus, but only over the single new sentence. When these phrase pairs were collected, we needed to estimate their phrase translation probabilities. As described in the introductory chapters, these translation probabilities are estimated based on the phrase pairs seen in the rest of the training corpus. In the incremental training approach however, we only look at a single sentence. This meant, that we needed the absolute counts for the occurrences of every phrase pair to estimate good approximations for the translation probabilities.

For this purpose we stored this information in a prefix tree, to allow fast lookup (in $O(log(n))$) and a compact storage. In this data structure we stored the co-occurrence statistics of every phrase pair, and the occurrence counts of the monolingual phrases, which allowed us to compute the exact phrase table scores as described in (AMCb$^+$05), with the difference, that our developed system did this on the fly without the need of a time consuming training phase.

With the collected counts and the updated lexical probabilities it was possible to determine a score for the new extracted phrase which was from a mathematical point of view the same as if the phrase in the corrected sentence pair would have been extracted from the original training data. The only difference to the original batch training and our developed approach was introduced by the computation of the word alignment. For the new incrementally learned sentences, this word alignment was not part of the original EM steps, thus allowing for differences between a full training and this incremental approach. In addition to this error however, this computation introduced another small error, which was the relation to other related phrases, meaning phrases which used either the same source side, or the same target side.

The features used for the translation model as phrase table scores which determine how likely it is to use a phrase are based on the description in (KOM03). In the following, $\bar{f} = f_1 \dots f_J$ and $\bar{e} = e_1 \dots e_I$ denote the source component and the target component of the phrase pair, f_j and e_i are single words in the phrase. $c(\cdot)$ denotes the count operator and A is the word alignment.

- the most obvious feature that is used is the phrase translation probability, the prob-

ability of the source phrase $\bar{f}$ given the target phrase $\bar{e}$.

$$p(\bar{f}|\bar{e}) = \frac{c(\bar{f},\bar{e})}{\sum_{\bar{f}} c(\bar{f},\bar{e})} \tag{4.13}$$

- As shown in (KOM03) adding the inverse phrase translation probability helped to improve the translation performance of the system, which is why this is added as a second score into the system.

$$p(\bar{e}|\bar{f}) = \frac{c(\bar{f},\bar{e})}{\sum_{\bar{e}} c(\bar{f},\bar{e})} \tag{4.14}$$

- With the above described two features alone, the lexical probabilities which were used for the extraction of the phrase pair are only used indirectly only by the fact this phrase pair was extracted at all. However the information the lexical probabilities provides is lost. This can be very harmful however, because alone from the description of the phrase extraction algorithm, it can be seen, that this algorithm tends to extract more phrases than might be actually helpful, and most of these phrases are seen only once or twice in the whole training. The problem with the phrase probabilities in this case is, that for such rare events no reliable statistics exist, while the lexical probabilities could help to identify good or bad phrases.

  Therefore the next feature which is part of the score for a phrase is the lexical probability of a phrase. The better the translation of the words within the phrase matches, the more likely is the phrase a good building block for translations. The lexical probability is the probability to produce this phrase pair given $\bar{e}$ and the word alignment A.

$$p_{lex}(\bar{f}|\bar{e},A) = \prod_{j=1}^{m} \frac{1}{|\{i|(i,j)\in A\}|} \sum_{\forall(i,j)\in A} p(f_j|e_i) \tag{4.15}$$

- For the lexical probability the same holds as for the phrase translation probability, the system can be improved when the inverse lexical translation probability is also used:

$$p_{lex}(\bar{e}|\bar{f},A) = \prod_{i=1}^{n} \frac{1}{|\{j|(i,j)\in A\}|} \sum_{\forall(i,j)\in A} p(e_i|f_j) \tag{4.16}$$

While the computation of these probabilities is still correct in the sense, that if this phrase would have been extracted during training, it would have had the same probabilities, it is important to keep in mind that we are still introducing an error when you look at the model as a whole. The problem with the described method is, that the updates needed to be very fast, making the updates unnoticeable to the actual users of the system. This

however made it impossible to update every score that was affected by the change in phrase occurrences. This meant, that the total in probability mass for the co-occurrence statistics of a phrase affected in the incremental model update does sum up to $\tilde{p} > 1$.

For this reason, we took care in our experiments to keep track of if at some point the improvement speed of our developed system slowed down considerably compared to the traditional batch learning approach. Our hypothesis at this point was, that it is actually a desirable effect, to see the initial phrases accumulating slightly more weight, than the incrementally learned ones, because of the more reliable word alignment and the slightly worse quality for the new phrase pairs caused by the missing EM steps. Alternatively it is possible to keep track of phrase scores that should be modified with a *dirty bit*, which would trigger an update of the phrases on the fly during first usage after modification. However this would slow down the translation speed of our system, which is why we did not investigate this further.

Another thing that set us apart from (OMGVC10) is the fact that our use case was different in that sense that we do not have interactive machine translation where the goal is to generate a one hundred percent correct translation, but that we expected our users to only correct completely wrong sentences, while understandable sentences might not be touched and corrected at all. That is why we had to prepare our system directly to the situation that often previously unseen phrases are added to the system. Phrases only seen once or twice in the training corpus tend to have bad estimates for their translation probabilities. To deal with this problem we also used smoothing directly in the dynamic updating process of the phrase models. We relied on the well known Good-Turing (CG96) smoothing which we also used during training of the baseline models and which is known to provide good results for the smoothing of phrase tables (FKJ06).

In Good-Turing smoothing the observed counts $c(e|f)$ are replaced by a new estimate according to the following formula:

$$\tilde{c}(e|f) = (c(e|f) + 1)\frac{n_{c(e|f)+1}}{n_{c(e|f)}}$$

where n_c denotes the number of different events having an observed count of c. By this re-estimation we reduce the risk of distributing too much probability mass to events only seen once.

For the computation of the shown probabilities it is necessary to keep the counts available during the incremental learning phase. We achieved this by the usage of a highly compact and efficient prefix tree which allowed a fast lookup of the needed absolute counts. The

same data structure is also used for the storage of the phrase scores. For efficiency reasons, the phrase scores are combined into a single score which is used during the decoding phase. This single score is the weighted combination of the phrase scores described above, according to the log linear model described in Chapter 2. The combination into one score had the advantage, that we did not need to sum up the individual scores during decoding time (ZV07) making the translation process faster.

4.2.5. Updating Language Model

Besides the described translation model, another important building block of most statistical machine translation systems - and in the one used throughout this thesis - is the language model. The task of the language model is to predict a word given its preceding words. For this task n-gram based language models are state of the art which assign a probability to the n'th word given the last $n-1$ words. The estimation of this probability is based on the occurrences of the $n-1$ preceding words with and without the n'th word.

With every new sentence, corrected by a user, we collected all n-grams occurring in this sentence and used these to update the language model.

For the task of updating the model, we had to keep the absolute counts of the occurrences available at update time and we needed the ability to modify these counts. Again, we used a prefix tree as data structure for this task, because of it's compact representation and fast access.

Updating the language model meant a modification of the probabilities, which are computed the following way:

$$p_{LM} = \frac{c(f_i, f_{i+1}, \ldots, f_{i+n-1})}{c(f_i, f_{i+1}, \ldots, f_{i+n-2})} \tag{4.17}$$

While this equation describes the general language model probability in our implementation we used a smoothed estimate for the probability, reducing the risk of overestimating events only seen a few times. In our work we developed a model, allowing the well known Kneser-Ney smoothing for this task. The main idea behind this smoothing technique is to subtract a discounting constant from the counts of n-grams and a modification of the probabilities for the n-grams of the order n-1 and lower which puts the probabilities used in backup in relation to the variability of the preceding words. This resulted in the following equations presented in (CG96):

$$p(f_{i-k} \ldots f_i) = \begin{cases} d(f_{i-(k-1)} \ldots f_i), & \text{if } c(f_{i-(k-1)} \ldots f_i) > 0 \\ bo(f_{i-(k-1)} \ldots f_{i-1}) \cdot p(f_{i-(k-2)} \ldots f_i), & \text{otherwise} \end{cases} \tag{4.18}$$

where:

$$d(f_{i-(k-1)} \dots f_i) = \begin{cases} \frac{c(f_{i-(k-1)} \dots f_i) - D}{c(f_{i-(k-1)} \dots f_{i-1})}, & \text{if } k = n \\ \frac{n(* \cdot f_{i-(k-2)} \dots f_i) - D}{n(* \cdot f_{i-(k-2)} \dots f_{i-1} \cdot *)} & \text{otherwise} \end{cases} \tag{4.19}$$

and:

$$bo(f_{i-k-1} \dots f_{i-1}) = \frac{1 - \sum d(f_{i-(k-1)} \dots f_i)}{1 - \sum d(f_{i-(k-2)} \dots f_i)} \tag{4.20}$$

Following the notation in (S^+02) with $n(* \cdot f_{i-(k-2)} \dots f_i)$ the number of unique k-grams ending with $f_{i-(k-2)} \dots f_i$ is meant and equally with $n(* \cdot f_{i-(k-2)} \dots f_{i-1} \cdot *)$ the number of unique k=grams with the middle words $f_{i-(k-2)} \dots f_{i-1}$ is meant. This results in the fact that the score of an n-gram which has not been seen before gets higher, if the corresponding n-1-gram was seen with more different preceding words, and the score gets lower if the number of unique $n-2$ grams that cover the middle $n-2$ words of the original n-gram becomes higher. All in all our model needed to store these additional values, $n(* \cdot f_{i-(k-2)} \dots f_i)$, $n(* \cdot f_{i-(k-2)} \dots f_{i-1} \cdot *)$ and $c(f_i, f_{i+1}, \dots, f_{i+n-1})$ compared to a not incremental improvable system which does not need those values at runtime.

As we already mentioned in the description of the translation model updates, one problematic was introduced with this approach. The update of the language model scores introduced again additional probability mass which resulted in the fact that

$$\sum_{w \in V} P(w | w_{k-1} \dots w_{k-(n-1)}) > 1 \tag{4.21}$$

where V is the vocabulary. While for the translation model, there was also another difference in the extraction of the phrases, this is not true for the language model and the only difference in scores which would have been extracted during batch training and in our new approach was this additional probability mass. To minimize the effect of this, we decided to develop a hybrid training, where we continued to do the incremental improvements for a certain amount of data, and then we made a complete retraining based on the additional training data extracted from the user feedback to establish a new baseline for our incremental training. An additional way of dealing with this problem would be the introduction of dirty bits for every language model entry, updating the wrong language model scores on demand. When we tried this in our training routine we saw a significant increase in decoding time, which is why we did not use this approach in the final developed system and leave this for future research.

4.2.6. Optimization of Feature Weights

In the traditional offline training of a statistical machine translation system, one step that plays a big role is the optimization of the feature weights $\lambda_1 \dots \lambda_n$ giving weight to the feature functions $h_1 \dots h_n$ in the log linear model. This step involves a re-estimation of the optimal values depending on translation and a reference translation of a development set, usually iterated over several steps. However this is a time consuming task. The translation of the development set alone already costs time which makes it impossible to do this step on todays hardware while keeping the constrain that model updates can be made during real time translation usage.

On the other hand the optimization of the feature weights is necessary to estimate a good ratio between the different features. According to our previously described methods, the models are all computed the same way as done during the traditional training, only introducing small errors which can be neglected at the beginning but might accumulate over time. This observation however lead us already to the incremental training setup, which required a full traditional retraining after a certain amount of added sentences. And at that moment the scaling factors will be re-estimated anyway, so that we decided to work with the old scaling factors up to the moment where a new full training is done. One effect we expected from this behavior was that we expected to be less prone to fluctuations in the optimization step, which is still a field of current research (CDLS11), (Eid12).

4.3. Feedback Quality

In a crowd sourced environment one of the biggest problems is the mixed quality of the feedback from the users. There might be users with excellent translation skills providing perfect translations, but also users who provide poor translations or even wrong translations.

This becomes relevant in two different situations. The first one is the presentation of user generated translations to other users. We wanted to present the best translations possible, therefore we needed to prevent bad translations from being shown.

In addition to that however our main goal was to learn from user provided translations and improve our system based on that. For this task the filtering of bad translations was even more important, because a bad translation would lead the system to extract bad phrases and therefore harm the overall quality of the machine translation system. So the identification of bad translations was one integral component of our developed system for incremental improvements.

Already in (ZV02) an approach for doing something similar was presented. They presented a method to perform a cleaning of the training data, where small movements, deletions or insertions introduced offsets in the bilingual alignment which were then resolved with a dynamic programming approach. In their work they relied on the IBM 1 alignment to compute the matching of the sentence pairs.

Work which goes more into the direction of our work has been done in (HMNW11), where a method was used to remove noisy data from the training corpus which was assumed to contain noise and means a removal of the sentences at all. This is comparable to the situation that users provide translations which might be correct or not and which then are either added to the incremental training or not. They trained a support vector machine (HDO$^+$98) which decided if a sentence pair is a correct translation or not.

We extended their work by integrating the HMM alignment information instead of the IBM lexical alignment score and introducing additional features. The information for computing the HMM alignment score is available to our system because of the other components we need for the automatic learning of improvements. So we are able to use this information as well and generate a score, which also incorporates the word order to some extent in contrast to what was presented in (HMNW11). Altogether we implemented the following features for deciding whether a sentence pair is a translation or not:

- normalized HMM-alignment score for source to target direction
- normalized HMM-alignment score for target to source direction
- unaligned source words
- unaligned target words
- length of source sentence
- length of target sentence
- maximum fertility of a source word
- maximum fertility of a target word
- ratio of words which are the same in source and target sentence compared to the length of the source sentence
- ratio of words which are the same in source and target sentence compared to the length of the target sentence

These features are computed on the fly for every new sentence pair obtained via user feedback. Only when the classifier indicates, that a sentence pair is a valid translation, this sentence pair is added to the system for improvement.

However our system still kept all the feedback, no matter if it was used for improvements or not. The reason is that this feedback might still be valuable information in the future. On the one hand it allows to identify users who tried to sabotage the system by providing bad translations. On the other hand there might also be false positives among these examples, which are actually good translations, but were classified the wrong way. Since our classification method relies on the underlying information from the translation engine, it was to expect, that these misclassified examples were very hard to translate for our translation engine. For that reason we kept these examples, so that in a future step they could be identified and used in future trainings to improve the overall quality of the system.

4.4. Assistive Translation

One big issue with crowd sourcing translation data is always the fact, that the generation of such data is a very expensive task and can only be done by language experts. In addition to that it is also very hard to support people who are willing to provide translations with automated methods. The cognitive load a translator is facing when asked to correct translations is very huge and usually it is faster to rewrite the correct translations from scratch instead of correcting them. For some time work has been done in the field of assisting users who are willing to provide translations. In (Koe09) such a tool *caitra* has been proposed, assisting users with their translations. This culminated in *CASMACAT* (ABC$^+$14) which enhanced the post editing process of machine translation tasks, so that in the end the editors needed less time for correcting translations. They achieved this mainly by providing auto completion of user generated input, which is what we did in this thesis as well.

The main difference in our work and the work described in (ABC$^+$14) is that our decoding process for the machine translation is very different to their stack based decoder. In a stack based decoding approach the final translation is constructed word by word from the sentence start (of the translation) to the end. This allows very effective pruning of the search space, when a user input is taken as reference to expand upon.

In an ITG based decoder relying on the Cocke Younger Kasami algorithm for decoding as described in Chapter 2 however this is not the case. Only in the last step it is clear, which translation parts will be used for the start of the final translation hypothesis. For

every sub translation created during decoding it is not known until the full translation is constructed, where the position of this sub translation is in the final translation.

As a result we changed the pruning during the decoding process to create translations given a specific beginning.

Let t be the user defined beginning of the translation, for every entry in the decoding chart, when computing the hypothesis $h_{i,k}$ for the words $w_i \dots w_k$ covered by this entry in the decoding chart, do not prune away those hypothesis matching one of the following requirements:

1. $h_{i,k}$ is sub-string of t
2. $h_{i,k} = t$
3. $h_{i,k} = t \cdot \tilde{h_{i,k}}$
4. there is a suffix $\hat{t}$ in t that is a prefix $\hat{h_{i,k}}$ in $h_{i,k}$

For all hypothesis being part of a translation matching the user provided start t one of the above mentioned requirements must be met. Therefore those hypotheses must be kept for further decoding and may not be pruned.

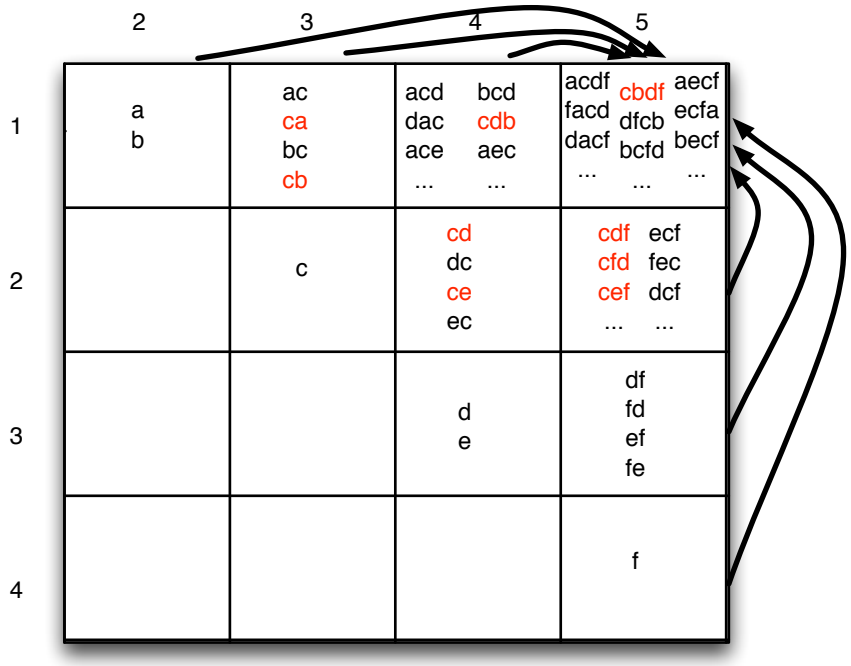

Figure 4.4.: Schematic overview of cyk with marked partial hypothesis immune from pruning

This different decoding however introduced more complexity for the task of auto com-

pletion. While for a stack decoder the task of expanding a given prefix of a translation to a full sentence covering translation essentially allows the pruning of all hypothesises which do not start with this prefix, the above described algorithm prunes less than before, because the *ITG* based search needs to keep all alternatives which contain parts of the prefix.

This results in an increased memory footprint during decoding and in a slower decoding speed. However we were able to keep the speed of the decoding process so high, that we were still faster than the typing of a human, which allowed us to use this technique for our purposes.

4.5. Conclusion

We presented in this chapter our developed algorithms and techniques for making online improvements of statistical machine translation possible. We provided an overview over the traditional batch training approach and used this as a starting point for the development of the techniques needed for incremental model learning.

We started with the description of how we were able to compute word alignments for a single bilingual sentence pair and based on that how phrases were extracted and scored from this alignment without a full retraining. In a next step we described the final model updates and the techniques that allowed us to achieve all these steps within the given time and memory constraints.

In the following part we described how the steps for the language model update were performed with its accompanying data structures needed for the fast turn around times.

The next two sections of this chapter concentrated on the questions, which algorithms were developed to bring the proposed techniques in use for real world usage. For this reason we showed how noise - in this case bad translation examples - can be automatically identified and removed from the incremental training, making the system more robust for the case of sabotage and bad feedback.

In addition to that we showed the developed mechanisms for autocompletion of translation prefixes which we integrated in the ITG based decoding process to assist users who are willing to provide corrected translations.

5. Evaluation

This chapter presents the evaluation of the methods described in the preceding chapters. Since it was very difficult to get meaningful amounts of crowd sourced data, the experiments were conducted on the publicly available TED talks (TED13). These talks are quite similar to lecture situations, having one presenter talking about a topic in a popular-academic manner. This was used as an approximation to the style that is used in university lectures.

Similarly to the work presented in this thesis, the transcripts and translations for the TED talks are generated by volunteers in a crowd sourced manner. This is exactly the situation we built this system for, so that this environment built a very good approximation of what we expected to experience in a lecture translation environment at universities.

For our evaluations we downloaded the first 1000 talks which are available on the website for the TED project, together with the corresponding crowd sourced translations and transcripts. The data was gathered on February 11th of 2013. To allow for comparison across different language pairs, we selected the set for the minimum error rate training and the unseen testing data out of a subset of talks which were available in all languages we planned to look at. The used corpora had the following statistics:

Corpus	Sentences	Words
optimization german	1444	15036
optimization english	1444	15515
crossvalidation german	1057	11510
crossvalidation english	1057	11420
out of domain german	103k	859k
out of domain english	103k	857k
in domain german	283k	2488k
in domain english	283k	2597k

Alone from the size of the corpora, it is obvious, that these are comparable small corpora. Usually the training data size is much larger in the range of millions of sentences.
The reason why we chose to work on comparable small corpora was motivated by the fact, that we planned to make very detailed experiments evaluating the development of translation quality. We planned to make thousands of training and evaluation runs with different training data sets to evaluate how well our developed algorithms adapt to new training data and then comparing the baseline algorithm and our incremental learning. This was necessary, to prove our assumption that incremental training can compete with the traditional batch training was right and provides benefits over that method.
The problem with these trainings however is that the traditional batch training, as described in the previous chapters, needs a lot of time, making it impossible to run that many experiments on larger corpora. And in favor of having more datapoints on smaller corpora compared to only a few datapoints on large training corpora, we opted for the more detailed experiment.
Still, even with the smaller corpora, a parallel implementation of the word alignment (GV08) and parallel trainings on multiple machines with multiple cores, we still experienced a runtime of more than two months for the batch trainings alone.

5.1. Baseline System

To judge the quality of automatic improvements made by our system, we had to create a baseline system first, which then was used to compare our developed techniques against.
The baseline system was built by using the traditional training steps of a machine translation system as described in (KHB+07). The goal of the baseline system was to see, how the traditional batch training improved in translation quality over time when adding more

data. We evaluated this by starting with a system which had no talks from the TED conferences in the training data at all, and then we iteratively added a new talk to the training to evaluate the translation quality before and after the addition of this additional data. Every training cycle involved also a minimum error rate training (Och03).
The algorithm used for this training procedure is shown below.

Data: trainCorpus, talks
select $dev \subset talks$;
$talks \leftarrow talks \setminus dev$;
select $test \subset talks$;
$talks \leftarrow talks \setminus test$;
train MT using $train$;
foreach $t \in talks$ **do**
  $startofiteration = now()$; $train \leftarrow train \cup t$;
  retrain MT using $train$;
  optimize towards dev;
  $s_t^{test} = score_{train}(test)$;
  $trainingtime_t = now() - startofiteration$;
end

Algorithm 3: Baseline training procedure TED talks

5.2. Global Improvement of Translation Quality

In a first experiment, we aimed for comparing the improvements in translation quality on unseen texts between our proposed incremental learning method and the traditional training approach when new data is added to the training.
For this experiment we used a similar setup as described in (Amb11). While in their work they focus on the task of active learning, we did not have the situation of active learning, but we looked at the improvements based on generic in domain data of other TED talks, with regard to the influence it has on the resulting system performance. This means that compared to (Amb11) the improvement we were expecting to gain was depending on the whole talk which was added, instead of a preselection of sentences which might have the highest impact in translation quality.
With this setup, the experiment simulated the effect we expected to see in the real world scenario of a lecture translation system, where corrections in the context of one lecture have influence on the performance of future lecture translation tasks.

5.2.1. Overall Improvement of the System Depending on Training Corpus Size

The initial system we started with, did not include any data related to the task of lecture speeches at all, but was built on data extracted from the Tatoeba[1] corpus on February 11th 2013.
For minimum error rate training we used a set of 5 talks, and for the unseen testing data we also removed a set of 5 talks from the set of the collected TED talks used for training. The main goal of this first experiment was to compare how well both systems, the traditional training and our incremental method improved over time, So the final experiment setup followed the description in Algorithm 4.

Data: trainCorpus, talks
$train \leftarrow trainCorpus$;
select test talk *test* for testing (crossvalidation);
foreach $t \in talks$ **do**
  $startofiteration = now()$; incremental training of t;
  $train \leftarrow train \cup t$;
  $s_t^{test} = score_{train}(test)$;
  $trainingtime = trainingtime + now() - startofiteration$;
end

Algorithm 4: Incremental training procedure for TED talks

As outlined in the algorithm, the evaluation setup we used is the same between the baseline system and the setup used for the incremental improvement of the system.
In comparison to the training procedure for the batch training, the difference between the systems lies in the fact that our incremental system does not perform a full training, but only an incremental training step based on every new sentence.

As can be inferred from the training algorithms, during the training procedures we evaluated the systems basically after the full addition of a talk to the training data. Either incrementally or for the batch training. The development of this score makes the improvement of the system by additional training data visible.
If this score does not show an improvements over time, it would mean that there is no

[1] http://tatoeba.org/

improvement in translation quality when adding additional data to the training, which would essentially contradict the common sense that more training data improves translation systems[1].

In addition to the general question of how the translation quality changes with additional data, we took a very close look on the question of how fast this addition of new data is possible to result in improved models. For that reason we also tracked the time used for the training in the batch training, and the cumulative training time in the incremental training. We chose to track the cumulative training time for the incremental training, because the time used for the addition of one talk's data was expected to be in the range of seconds and summing up the time for every added sentence was then comparable to the total training time of a batch system.

5.2.2. Speed of Model Improvements

We started our evaluation with looking at the time used for training a system. The results of the measurement can be seen in Figure 5.1.

Alone from the development of the update mechanisms described in the earlier chapters, we expected a drastic reduction in training time for the incremental training compared to the baseline batch training. As you can see in the figure, this was exactly what we found. To make the training time visible for the incremental training, we used the cumulative time for the incremental training, meaning the total time spent in the incremental training for the addition of every single added talk up to the most recent one, and compared this to the time used for the single not cumulative baseline training covering the same corpus size. These times included the whole task of translating the unseen test corpora afterwards, which allowed us to include possible changes in the decoding speed caused by the changes in our model representation.

As described above, the baseline training was parallelized running on a 12 core machine, while our new incremental training approach used only a single core without any parallelization. Even with this parallel training approach the Figure 5.1 shows that the training time is too high, to do such a training whenever new data comes in. With a minimum of 76 minutes and an average of 130 minutes for the first few talks added for a complete training, it is not usable in an online setting requiring fast model updates to give users the feeling of immediate model improvements. This long training time became even longer with the larger training corpus, reaching an average training time of 220 minutes in the end. It is important to keep in mind, that even in the end of this experiment, the average

[1] as long as the data is related to the evaluation condition, which is the case in our setup

corpus size used for training is much smaller then what is used for production systems. In those cases the training time would be even much longer.

Our incremental training however managed the whole model update and new evaluation in less than 14 minutes in the beginning and 20 minutes in the end. Since we were always evaluating both translation directions on the unseen data this means that even after doubling the training data in size with incremental training, we managed to translate at a speed of $\frac{1200\text{seconds}}{2114\text{sentences}} \approx 0.6 \frac{\text{second}}{\text{sentence}}$ assuming, that after every sentence to be translated we performed a follow up step of incremental learning. It is clear, that the approach can be highly parallelized, by splitting the different model updates in separate threads, but also on a micro level the different n-grams in the language model and phrases in the translation model can be updated in parallel with minimal book keeping overhead to prevent overwriting of results. Furthermore it is important to note at this point, that the incremental training does not need any time consuming initial steps, but the time used for improving on a single sentence is more or less the same, no matter when a sentence is used for incremental model updates. Therefor, the estimate of 0.6 seconds per sentence is behavior we really see for the task of translating a sentence and then learning the correct translation for this. It is not part of a bigger batch which would make the system unusable for some time.

As mentioned earlier, Figure 5.1 compares the cumulative time used for the incremental training and the single iteration training time for the baseline batch training. This comparison allows us to identify the point where the total time spent for training incremental updates breaks even with the time spent for building new models from scratch for the resulting training corpus. This point is reached after $\approx 100k$ additional sentences. But this point only means that the total time spent for training of both methods breaks even at that moment. It still neglects the fact, that the system that used incremental training was fully usable over the whole time and constantly improving, while the baseline training would have stayed on the same performance level for the whole time and no retraining happened.

These observations in combination with the fact that we compared a very parallel training procedure using parallel training of the word alignment (GV08), parallel minimum error rate training (KHB$^+$07) and language model training, with a single threaded single core incremental training showed that our developed techniques highly outperform the baseline training approach with regard to the time needed for model updates.

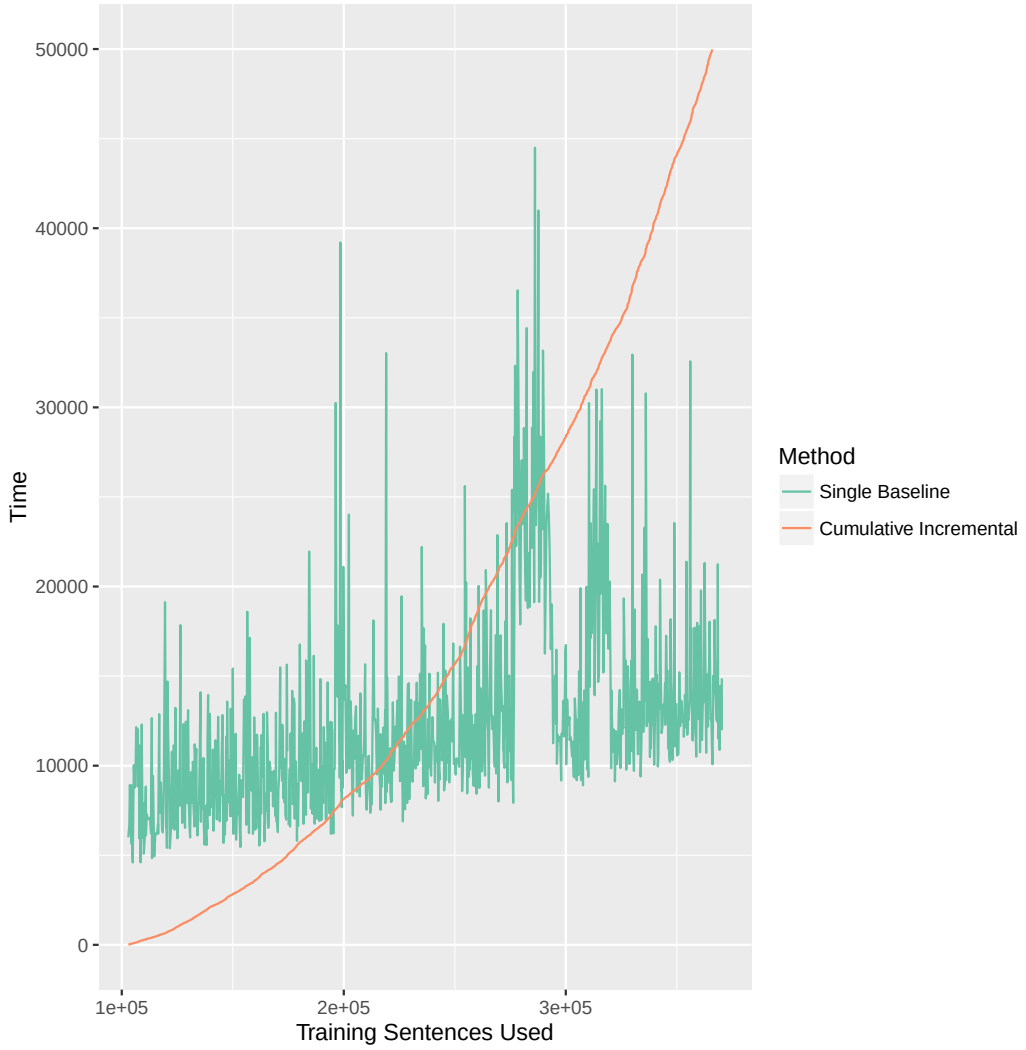

Figure 5.1.: Time used for number of sentences trained

5.2.3. Comparison of Translation Quality

Besides the actual time needed for the model updates, the most important question was however, how did the incrementally updated models compare to the baseline batch training based approach. Fast model updates only make sense when the resulting models also improve the translation quality over the original models, otherwise a model update would be harmful and should not be made at all.

For this purpose, we looked at the translation scores for the unseen test data generated in the above described training procedures for both the baseline approach and for the incrementally learned models (compare: Algorithm 3 and 4). The results for this experiment are shown in Figure 5.2 and 5.3.
It is visible at first sight, that the translation quality as measured in BLEU improved for both systems with the addition of new data. In both, the baseline training approach and our incremental training, we saw strong gains in BLEU in the beginning of this experiment, while at a later point these gains became smaller, but still both systems continued to improve.
It is also visible, that even though the incremental training shows improvements, the overall score falls behind the baseline approach. This is what we expected, based on the deficient math in the probability estimation for the models and the missing minimum error rate training. However these differences were in a range, that the overall improvement and the speed of the system still exceeded our expectations. When modeling this system we actually expected the translation scores to decrease again after some time because of the increasing number of incorrect probability estimates.
Besides the question of the improvements alone, it was also from interest to understand, how this method of model updates compared to the baseline batch training. In the graphics depicting the actual system performances, we saw that the incremental learning can actually keep up to a certain degree, with diverging from the traditional training the longer the incremental procedure is used. While the traditional system managed improvements of around 5 to 6 BLEU over the course of our experiments, the incremental approach still managed to achieve astonishing improvements of approximately 4 BLEU. Even the graph depicting the development of BLEU scores looked very similar for both approaches, with a slightly lower gradient in the incremental improvement, due to the above mentioned differences in resulting models and missing optimizations.
One quite interesting observation that can be made from the graphs showing the score

development is the high variability of scores for the traditional baseline training compared to the low variation for the incremental training.
The reason for this difference is the MER training used for optimization of the scaling factors in the traditional approach. In the MER training, the optimization can end up in local optima, which is why two optimization runs, even when only limited data was added to the models, can still produce very different outcomes. For the incrementally growing models, there is no re-estimation of the scaling factors, resulting in slow changes to the models having only small effects. On the other hand this might also mean that a bad setting for the scaling factors was not updated and the system can degrade over time because of bad estimates for the scaling factors. These bad estimates for the scaling factors are kept for all future incrementally learned systems based on this training run and the missing re-estimation of the scaling factors does not prevent this. For this reason the development of the scores should be tracked on unseen data to prevent the system from continuously worsen.

5.2.4. Effect of Unknown Words

One might argue, that the seen improvement of the data was caused by the fact, that the model mainly learned the translation of new, previously unknown words. It is clear, that lesser unknown words in the translation model cause significant gain in the improvement of the Bleu scores. This thought comes immediately to mind, when you compare the previous graphs showing the improvement of scores, with a graph showing the number of out of vocabulary words over time in the same experiment.

This evaluation as shown in Figure 5.4 shows a curve which is similar to the inverted graph of the development of scores in the incremental training. One might argue for that reason, that all the benefits resulting in improvements of the Bleu score during the incremental training came from the learning of unknown words, because alone their addition to the phrase table will have a positive effect on the translation quality.
That is why we performed another experiment, where we restricted the computation of the BLEU score to only those sentences, which either did not contain an unknown word throughout the whole experiment - so already the baseline system had translations for the words, or if a word in a sentence was unknown, it had to remain unknown even at the end of the experiment with all additional data in the training available. By restricting ourselves to these sentences, we were able to exclude the effect of reducing the out of vocabulary words and the increase in BLEU score was caused alone from having better phrases in the translation model or better n-gram probabilities in the language model.

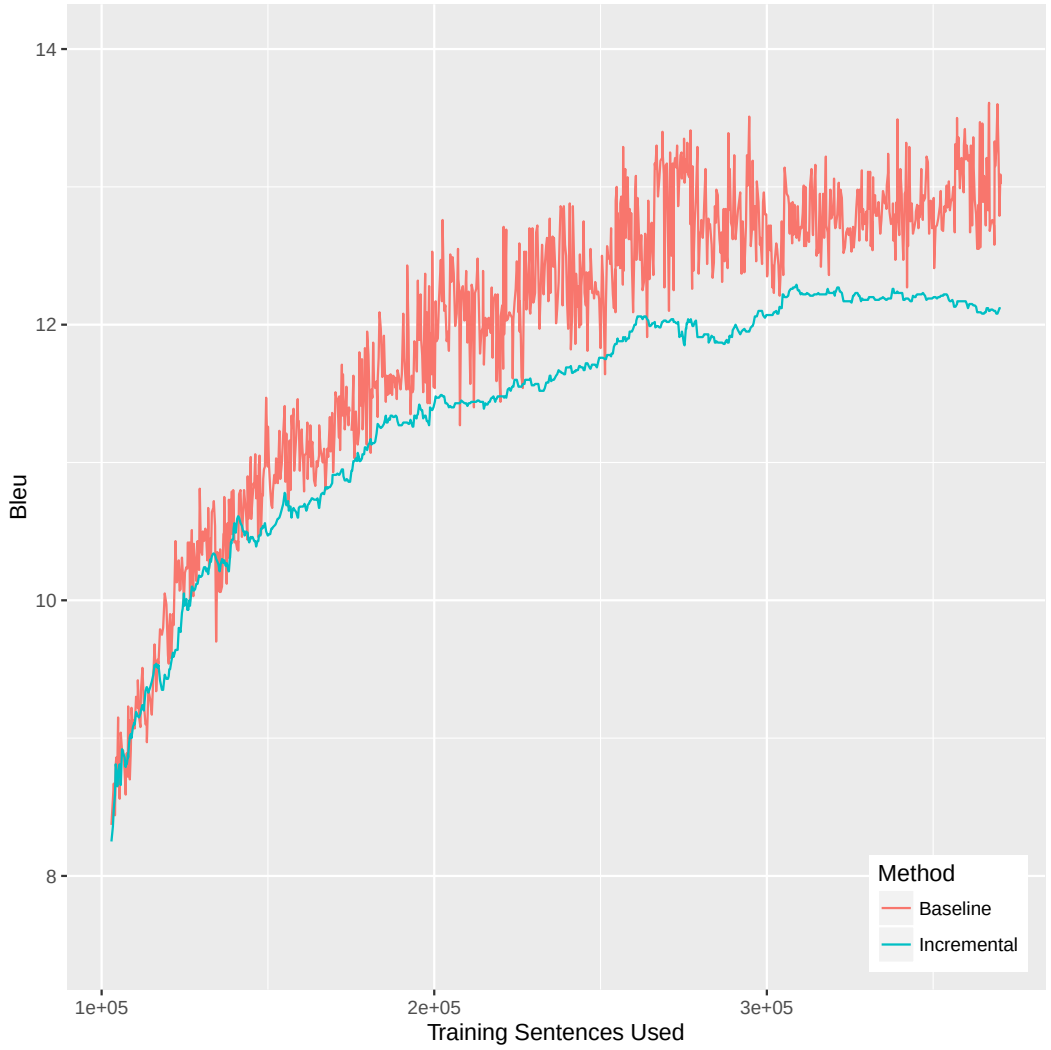

Figure 5.2.: Score by training corpus size English -> German

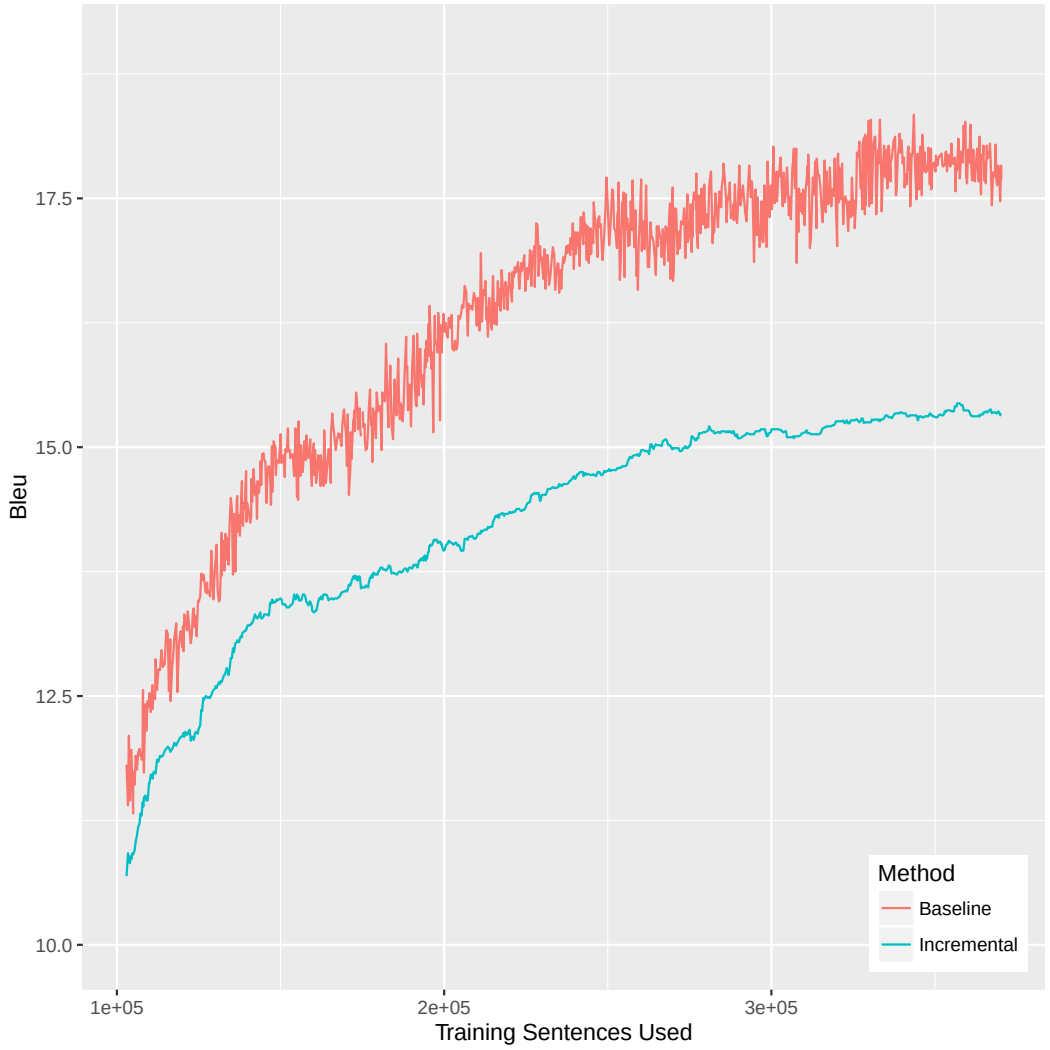

Figure 5.3.: Score by training corpus size German -> English

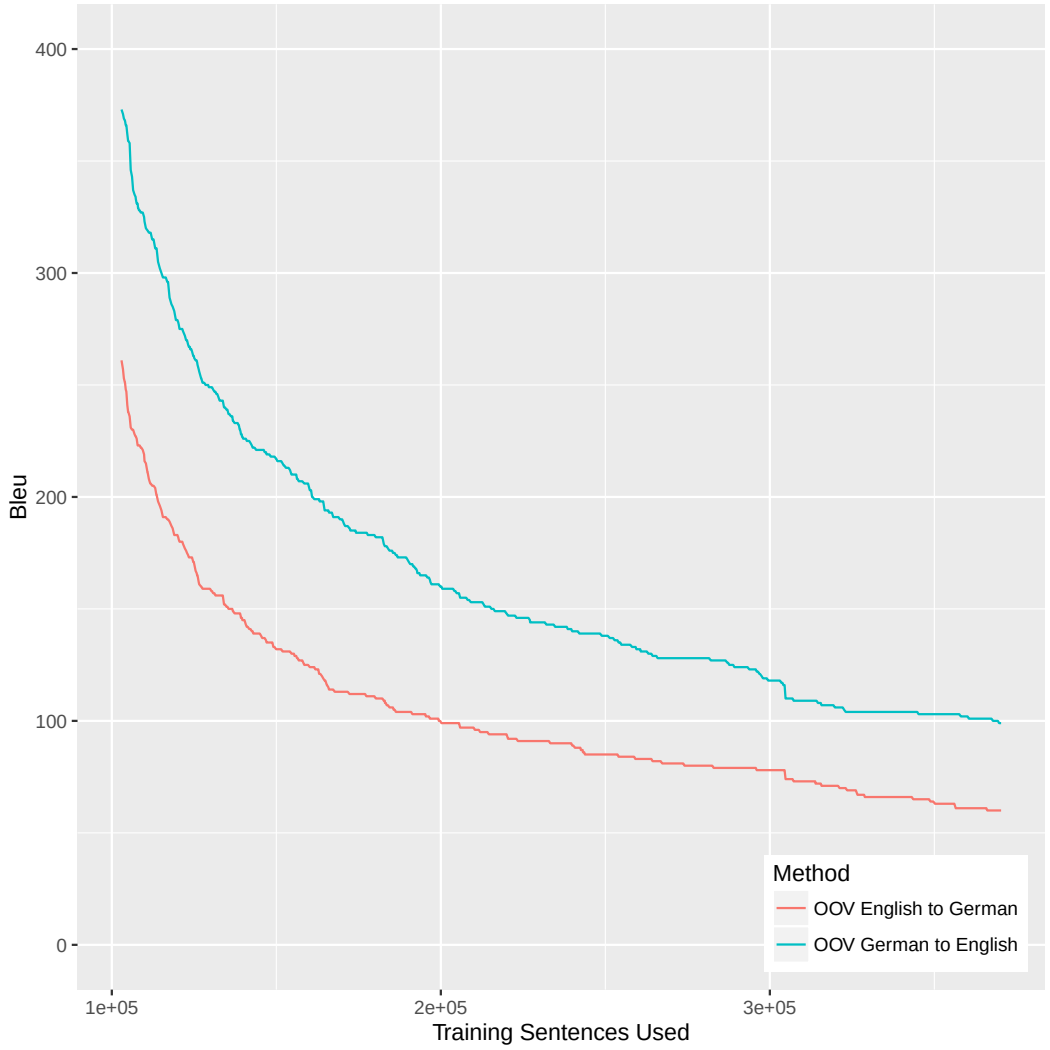

Figure 5.4.: Number of out of vocabulary words over the course of the experiment

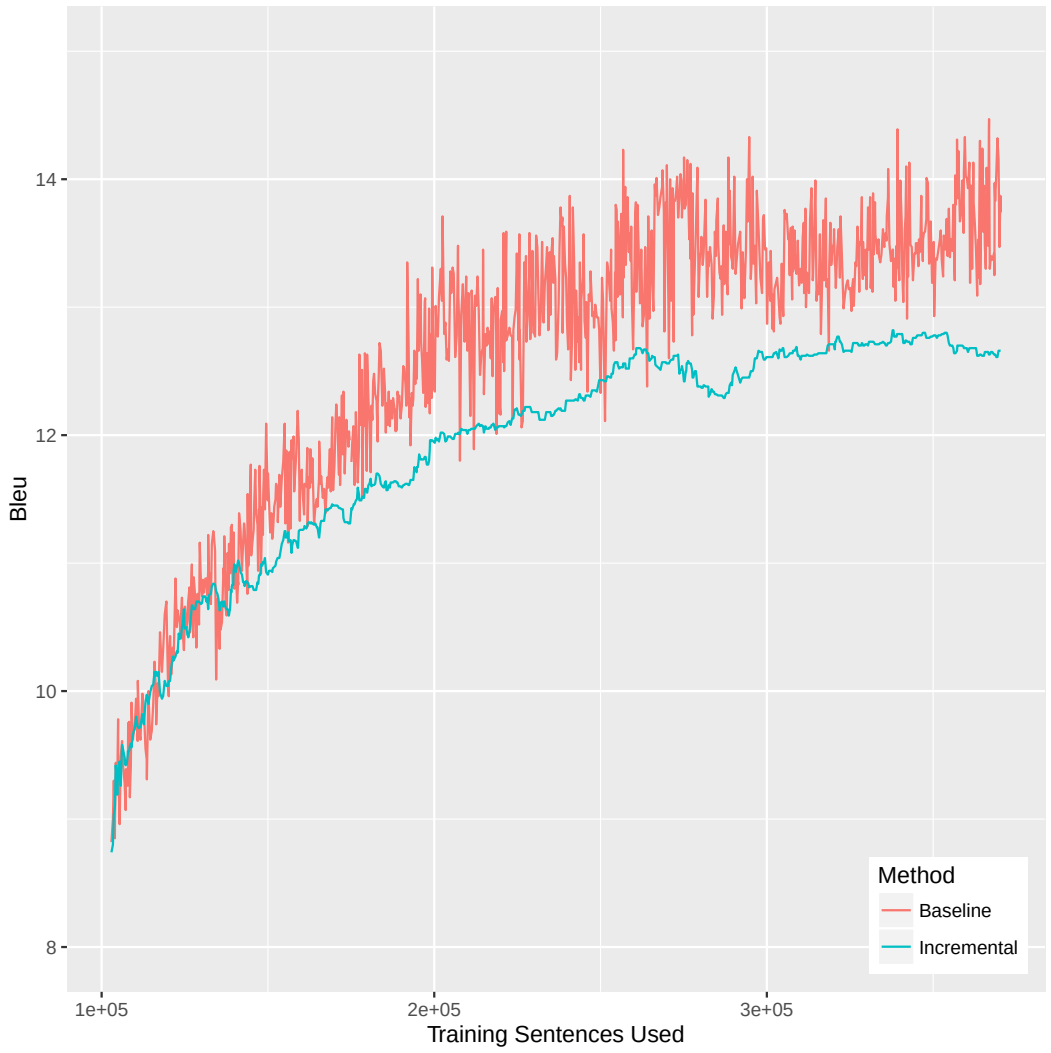

Figure 5.5.: Score without OOV influence by training corpus size English -> German

The results of this experiment are depicted in Figure 5.5 and Figure 5.6. Since a change in the set of sentences used for computing the BLEU score makes evaluations incomparable, we also recomputed the baseline scores of the full training to show how the traditional training improves when the effect of unknown words is eliminated.

One of the first things recognized in these graphs is the fact, that even without the influence of out of vocabulary words, both systems still improved the overall translation quality the more data was used for the training. Alone from the shape of the graphs, they look pretty much the same as before with the influence of the unknown words. When looking at the raw numbers, the BLEU score in this evaluation turned out to be higher than in the first evaluation, but this is due to the fact, that this new evaluation without the sentences which had the unknown words before, is much easier than the original evaluation. This is easy to understand when looking at the computation of the BLEU score itself. The score is dependent on the matching 1- to 4-grams in a translation. With sentences that contain

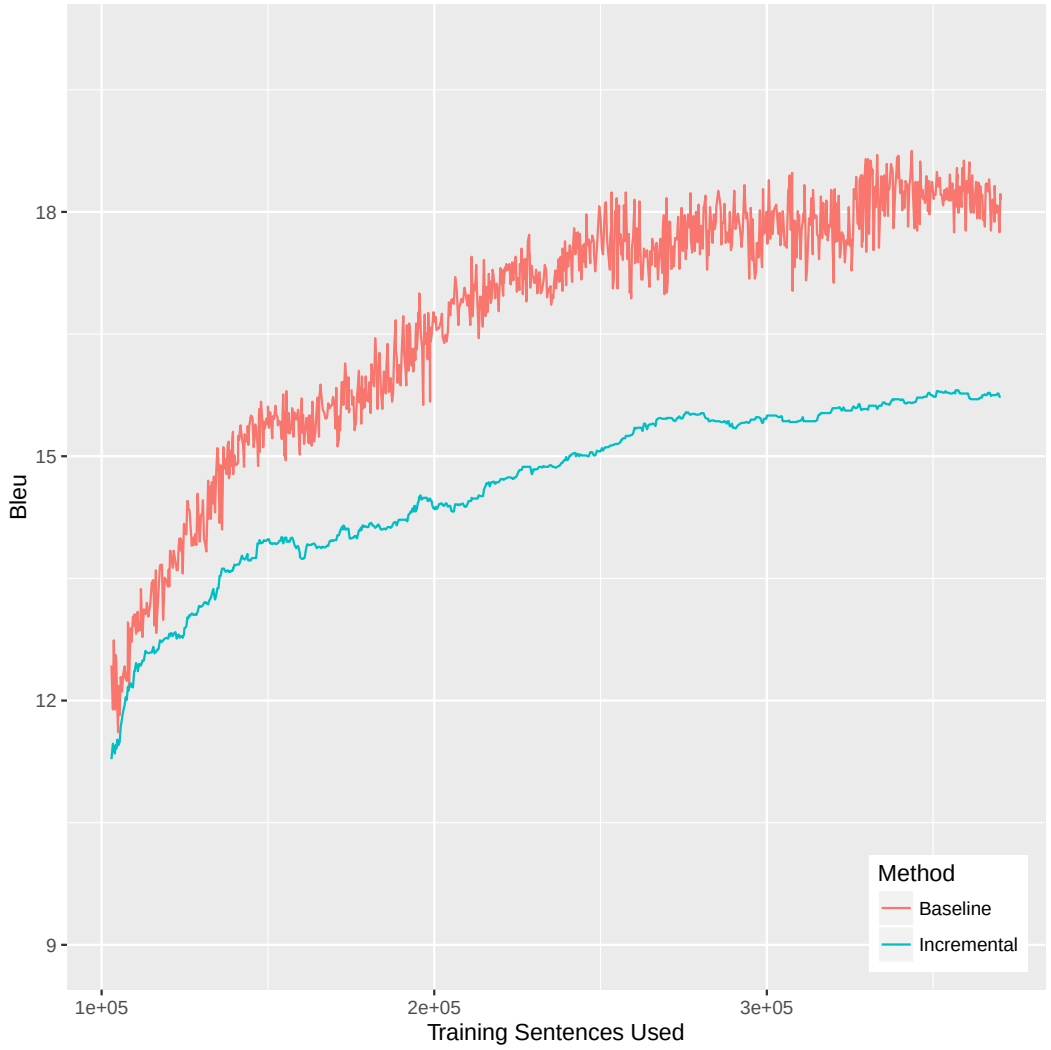

Figure 5.6.: Score without OOV influence by training corpus size German -> English

words which are not covered by our translation models, this means, we have at least a unigram, 2 bigram, three 3-gram and four 4-gram misses, which we don't have anymore, as soon as we remove these sentences.
With the still increasing BLEU scores even for sentences, where we did not profit from new learned single words, this evaluation result was a very strong one, showing that our proposed method for incremental learning was able to improve in translation quality by learning better phrases and improved language model probabilities in a fast and efficient way.

5.2.5. Iterated Incremental Models

In the description of our incremental learning algorithms, we pointed out, that our models have a deficiency in the mathematical models, which results in reduced accuracy of the probabilities the more data is added in the incremental training. Furthermore we pointed out, that the non performing of minimum error rate training might also results in a reduced translation performance, especially when the scaling factors are stuck in a local optimum which might prove to be bad when additional training examples are added.
To overcome this deficiency, we experimented with a hybrid training procedure which is a mixture of the standard batch training and the iterative process developed in this thesis. The idea behind this approach is to keep the fast model updates while in the background a new foundation for future model updates is built. This reduces the risk of running into situations where bad scaling factors result in a degradation of the system over a longer time and the retraining is also a moment which corrects the overestimation of past probabilities in the language and translation model.
For the setup of this experiment, we used the same domain as before, the TED talks, but this time we used a slightly different training procedure. We started a new incremental training after every 100 additional training talks, we rebuilt the whole training models from scratch using the batch training and starting an additional incremental training procedure with the resulting model as foundation.
The results for this experiment are shown in Figure 5.8 and 5.7.
From the results of the experiment it is obvious, that this proposed training approach is able to outperform the pure incremental training by a quite significant amount. Especially in the German to English translation example, the new proposed hybrid training is able to even compete with the batch training, while maintaining the flexibility of the incremental training. However we can also see from those experiments, that the hypothesis of bad scaling factors which might result in decreasing performance over time was also right. Es-

Figure 5.7.: Score by training corpus size German -> English

pecially for the English to German translation direction this is visible in the incremental training that was started after 700 additional talks.

In contrast to the batch based approach, our incremental learning allows a much easier detection of such events. Because of the much smaller variance in scores between the different iterations, it is much easier to keep track over the score development - already a moving average of the BLEU scores on held out data provides enough insight to be able to stop and restart the incremental training as soon as the gradient is negative for a certain period.

If one is not willing to rely on the gradient for deciding on retraining, this experiment and empirical results showed that a new iteration of batch training makes sense after approximately 50.000 additional sentences to prevent BLEU score plateaus or even declining scores.

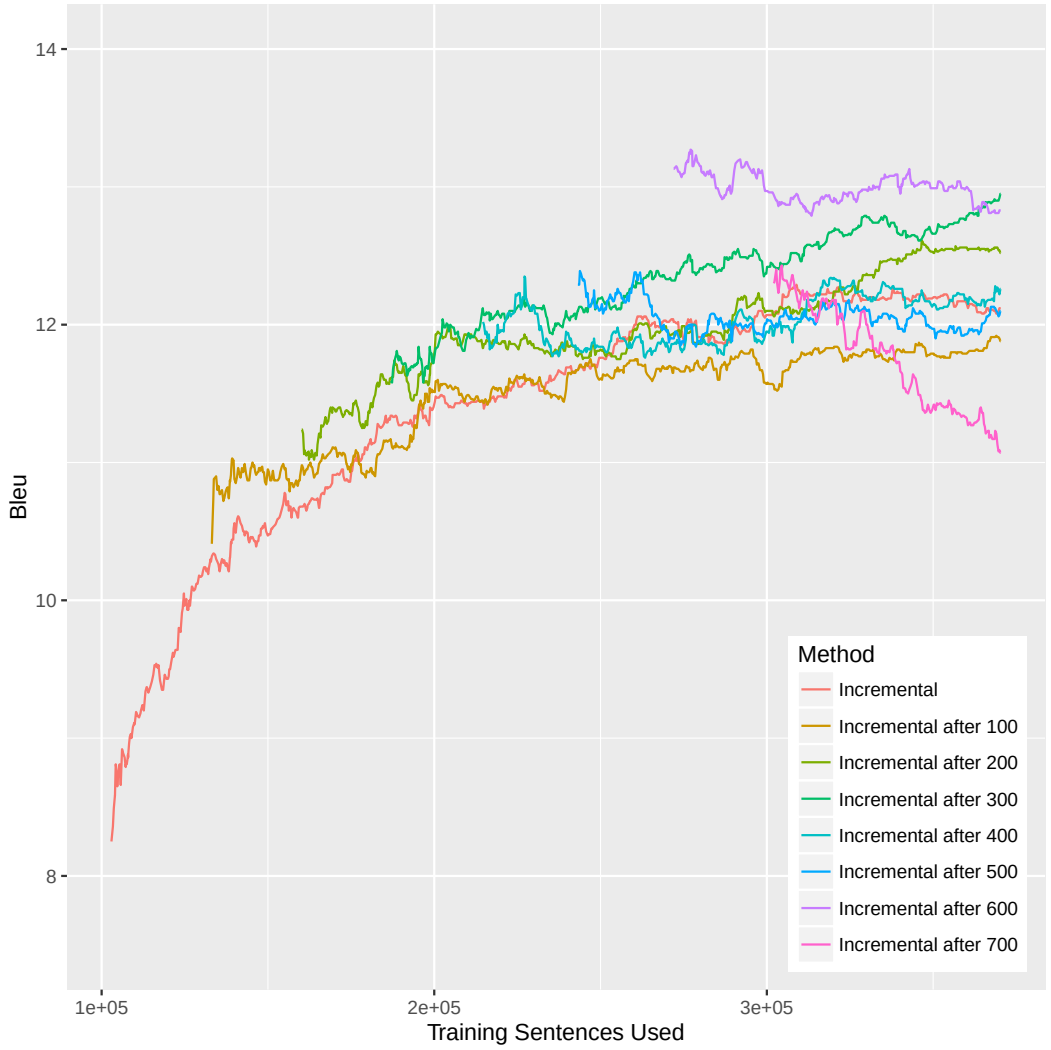

Figure 5.8.: Score by training corpus size English -> German

5.3. Intratalk Improvements of Translation Quality

5.3.1. Setup of Experiment

When we started to develop our system for incremental improvements to machine translation systems, we wanted to enable users of such a system to provide feedback, and then immediately see the effect their feedback had on the overall quality of the system. When a user provides a correction of some translation error, the user does not expect the same error to happen again when he tries the same input sentence again. While this problem is the normal case for systems trained in the baseline batch training approach, our method was developed to overcome this problem and provide the user with the corrected translation at the very next moment.

In contrast to the previous experiment this meant that we were not interested in the overall performance on unseen data, which is the normal scientific evaluation. But we explicitly looked at the translation quality of data that had been corrected and therefore was already part of the trained models.

In the context of this work this is even more interesting, since a correct translation after the user provided feedback means a much higher motivation for the user to continue giving feedback of correct translations, compared to the situation as it was before the incremental feedback, where users had to wait for a complete retraining of a system. This decoupled the moment the user provides feedback from the experience that the system really improved by what the user did.

For verifying our hypothesis, that the proposed incremental learning really provided significant improvement to the corrected sentences, we used a similar setup to the first experiment. Only this time we made two additional evaluations for every training iteration: whenever a new talk is going to be added to the system, the translation quality of this talk was evaluated before and after it was added to the incremental model updates. The resulting experiment used Algorithm 5.

The results of this evaluation are shown in Figure 5.9. In the graph we display the translation result that we obtained before this talk was added to the incremental training and we compare this score with the score we obtained after the addition.

As one would expect, the translation score on the talk was highly improved in the second evaluation. In both translation directions, German to English and English to German we saw on average improvements of more than 50 BLEU. This is similar to what is usually seen when evaluating on data that was included in the traditional training setup and it underlines the already in the first experiment shown competitiveness of the incremental

Data: trainCorpus, talks

$train \leftarrow trainCorpus$;

select test talk *test* for testing (crossvalidation);

foreach $t \in talks$ **do**

  $s_t^{talk} = score_{train}(t)$;

  incremental training of t;

  $train \leftarrow train \cup t$;

  $\widetilde{s_t^{talk}} = score_{\widetilde{train}}(t)$;

end

Algorithm 5: Incremental training procedure for TED talks

training. This very high BLEU score on a single reference essentially means, that most of the training talk was translated exactly as the reference after the manual correction.
On a closer look at the evaluation results of this experiment, one can identify some talks where the second score is not in the same range as for other talks. After an investigation of this issue, it turned out, that these talks were outliers because of either misalignments, or they were extremely short with a very specific translation which actually contradicts a correct translation (like *music* which was translated into *Hintergrundmusik*, a very specific translation which is not desirable when only translating the single word).

As a side note, it is interesting that the ranking of the systems switched for the tested English - German language pair. Even though it needs further investigation to make a solid statement on this, our assumption was that this is caused by the fact that the TED corpus is generated by translating from English into other languages. This might be a reason for less variation when translating into German, explaining the higher BLEU score. In addition to that, a higher BLEU score in the beginning also means that already in the beginning the models were stronger. That however means that new data had to compete during translation with the good models to be used during translation time. As a result this means, the better the base models are, the lower will the influence of the additional data be in the end.
For German as the target language this is especially the case. German as a high inflective language has many variations of word endings, which probably did not all show up in the baseline training. The addition of a German word for which this form has not been seen in the training before, does not have to compete in the language model with existing statistics extracted in the previous training. Therefore it is expected that the final translation of this

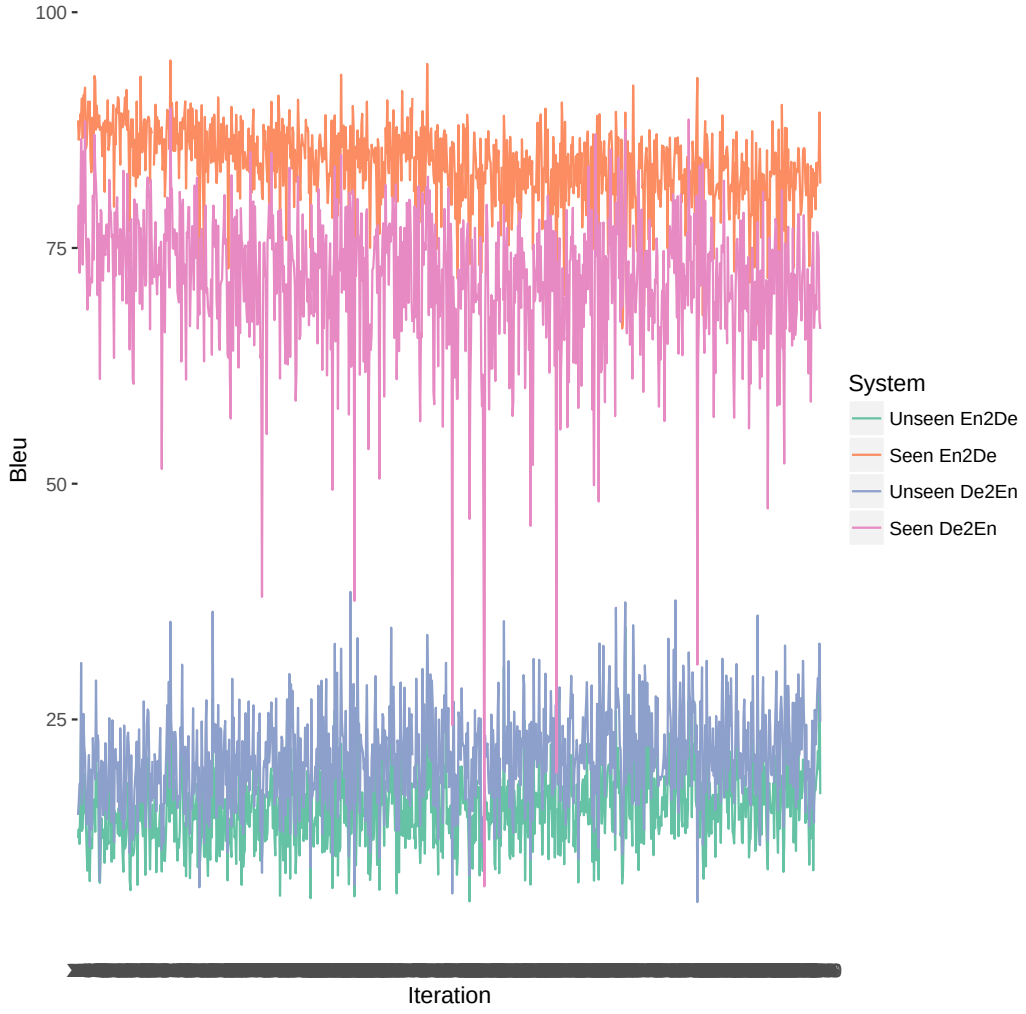

Figure 5.9.: Improvement of scores on seen data

word after adding it to the system is more likely to be correct. In contrast to this, when translating to English, the probability to see a new word form for a word is much lower, therefore the new added phrase has to compete with other existing phrases in the language model. This results in more variation in the translations, compared to the reference. This increased variation results in a higher perplexity, because of the many alternative usages of the word in the same surface form. The result is a slightly worse translation quality when it comes to perfect reproduction, while unseen data tends to be easier to translate compared to translating into a highly inflective language.

5.3.2. Effect on Future Unseen Intra-Talk Data

With the previously described experiments, we showed that our developed models were able to make fast updates which improved the overall quality of machine translation. After we proved that this is possible, we focused on the question if this fast update can be used for short term translation improvements. We were not only interested in improving the overall quality of our translation system, but we also wanted to evaluate, if we can see improvements in the immediate future for the same setting of the just added data. This goes beyond just the same domain, but is looking at the situation where future data of the same text, talk or presentation is influenced by corrections of the initial automatic translations of this event.

The motivation of this is a situation like a lecture which is being simultaneously translated and corrected by someone in the audience. We wanted to answer the question whether it is possible to improve the translation quality for the rest of a lecture given corrections of past data for the same presentation.

To our knowledge this is the first time this had been looked at. The reason for this is simply, that these intra-talk updates were simply not possible with the standard batch training approach and this would have made such an evaluation very time consuming with little effect on real world usage of a translation system.

To test the behavior of our incremental learning approach against this situation, we setup a new experiment, where we essentially first generated the translation of a sentence in a talk, and after this sentence was translated ”unseen”, we added the sentence with its human generated correct translation to our incremental learning. Then we continued with the next sentence in the same talk.

One might argue, that the main effect playing a role here was the correct translation of out of vocabulary words, which occur multiple times in the same talk, but we were also interested if there are additional changes observable, which showed that this learning setup also affected other situations, where all words were known in advance, but with the corrected past translations now better phrases were chosen.

Our assumption was that already in the context of adding only limited data future similar usage of language will result in a slight change and improvement in translation quality.

5.3.3. Intra-Talk Translation Quality Changes

For this experiment we used the following setup. In a first step we translated the talk with the baseline system which had no additional data from this talk included at all. In the following steps we iteratively translated a sentence of this talk, stored this as part of the

final incremental translation hypothesis for this talk and then used our developed incremental learning approach to update our translation models with this additional sentence and its reference translation.

This was repeated until every sentence of the talk was translated and we thus had a second translation hypothesis for the whole talk, this time with incrementally updated models used.

These two talks were then scored using the BLEU metric, allowing a comparison of the translation quality between both approaches. The algorithm used for this evaluation can be seen in Algorithm 6.

Data: trainCorpus, talks
$train \leftarrow trainCorpus$;
select test talk *test* for testing (crossvalidation);
$resultingTranslation \leftarrow \{\}$;
foreach $sentence \in test$ **do**
    $translation \leftarrow translate(sentence)$;
    $resultingTranslation \leftarrow resultingTranslation \cup translation$;
    incremental training of *sentence*;
end
$s_{test}^{onTheFly} = score(resultingTranslation)$;

Algorithm 6: Incremental training procedure for TED talks

For this experiment we used the same unseen test data as before, but in addition to that, we also ran this experiment on English - Spanish unseen test data to validate the results on another language pair. For the English Spanish training we again used a training set compiled of the first 1000 TED talks published, and the testing data consisted of the same talks used for the German English task.

The results of this experiment are shown in Figure 5.10, where the light blue column depicts the baseline system and the darker blue depicts the translation score obtained with the intra talk improvements.

We expected to improve our translation quality a bit, but we were quite surprised when we saw the actual results. The results showed that our intra-talk training clearly outperformed the baseline system, and that the past data of the same talk was incredibly helpful. For the English Spanish language pair we achieved improvements of up to 2 BLEU points. Normally this requires a lot of additional training data, usually in the hundred thousands of additional sentence pairs, while it was achieved in this case with a few hundred sen-

tences, which were extremely close to the domain of the future translation requests.
For the English - German language pair, the improvements were lower, but with 0.5 BLEU still very promising, given that this was the first time to our knowledge, that improvements had been possible in realtime during a lecture or the translation of a document.
As in our previous experiments the translation of unknown words plays an important role in the incremental learning, and one often heard argument is, that just adding the correct translations of previously unknown words would achieve the same improvements in translation quality.
We already showed in the previous evaluations that there is a significant effect beyond the learned translations of out of vocabulary words.
To understand the improvements in this intra-talk improvements, we looked at the sentences which changed in the resulting translations. In table 5.1 we have some examples of typical changes we saw in the results. These were selected randomly and reflect the fact that quite some changes in the translations were not related to the learning of out of vocabulary words.

5.1.

Baseline	as soon as it is a geschlossenes system , with participation of the united states ,
On the fly	as soon as it is a closed system , with participation of the united states ,
Baseline	well , one of the reasons it as a few trees are , is this :
On the fly	well , one of the reasons why it is so few trees , is this :
Baseline	and this is the most common chef fuel cow dung : in india .
On the fly	and this is the most common cooking fuel cow dung : in india .

Table 5.1.: Comparison of on the fly improvements

The first example in this table shows a typical example of a previously untranslated word. In the traditional translation system we are unable to translate *geschlossenes* into English. This is one of the problems with the German language, where the inflections cause problems with unknown words. When we used our incremental learning approach we were able to translate this sentence, because in a previous sentence we saw an example translation for *solange die USA aussen vor bleibt, ist das welt system kein geschlossenes system.*[1] The correct translation provided the incremental system with the needed information to make

[1]This is after preprocessing of the data which includes compound splitting. For that reason the output might look weird because of the single words instead of compounds.

above shown correct translation.

The second example however already shows that our incremental training was able to learn beyond previously unknown words and was able to improve its language model and translation model. All words used in the translation are pretty common, still the incremental system showed a slight improvement by selecting a better phrase, because the translation "*so few trees*" was seen before, so the language model was able to guide the translation into the right direction.

The third example is an example where the incremental training was able to learn a better phrase for the translation. Again, all words were known to the system, but in this case the domain of this talk is about energy and *cooking fuel*, where the German side uses a compound with the ambiguous word *koch* which describes both, a chef, but also cooking. In a previous sentence the words *koch brennstoff* are used with a reference translation *with an alternative cooking fuel*. This provided the information to correct the problem for the translation and disambiguation in this case, resulting in a correct translation in this situation.

Overall this experiment showed that a quite significant improvement is already possible with just the corrections used from the past sentences used in the same document to translate.

This is a very encouraging result, because it also means, that within a lecture environment students would immediately see improved translations, when they provided corrections to past translations of the same talk. As stated before, this is very important for the acceptance of such techniques among users.

5.3.4. Sentence Level Improvements

The promising results in the last experiment lead us to the idea of investigating this effect even further. We were interested in finding an answer to the question how much data is needed to achieve improvements and if it is possible to further reduce the work needed for improving the system.

The situation we had in our mind was again the setting of a lecture where continuously corrections for the machine translation system are provided. However we wanted to know if it would be possible to gain most of the improvements with only a subset being corrected and not every single sentence in the lecture.

To conduct this experiment we modified the previous experiment slightly and evaluated the translation performance, stopping the iterative training after every possible initial sub-

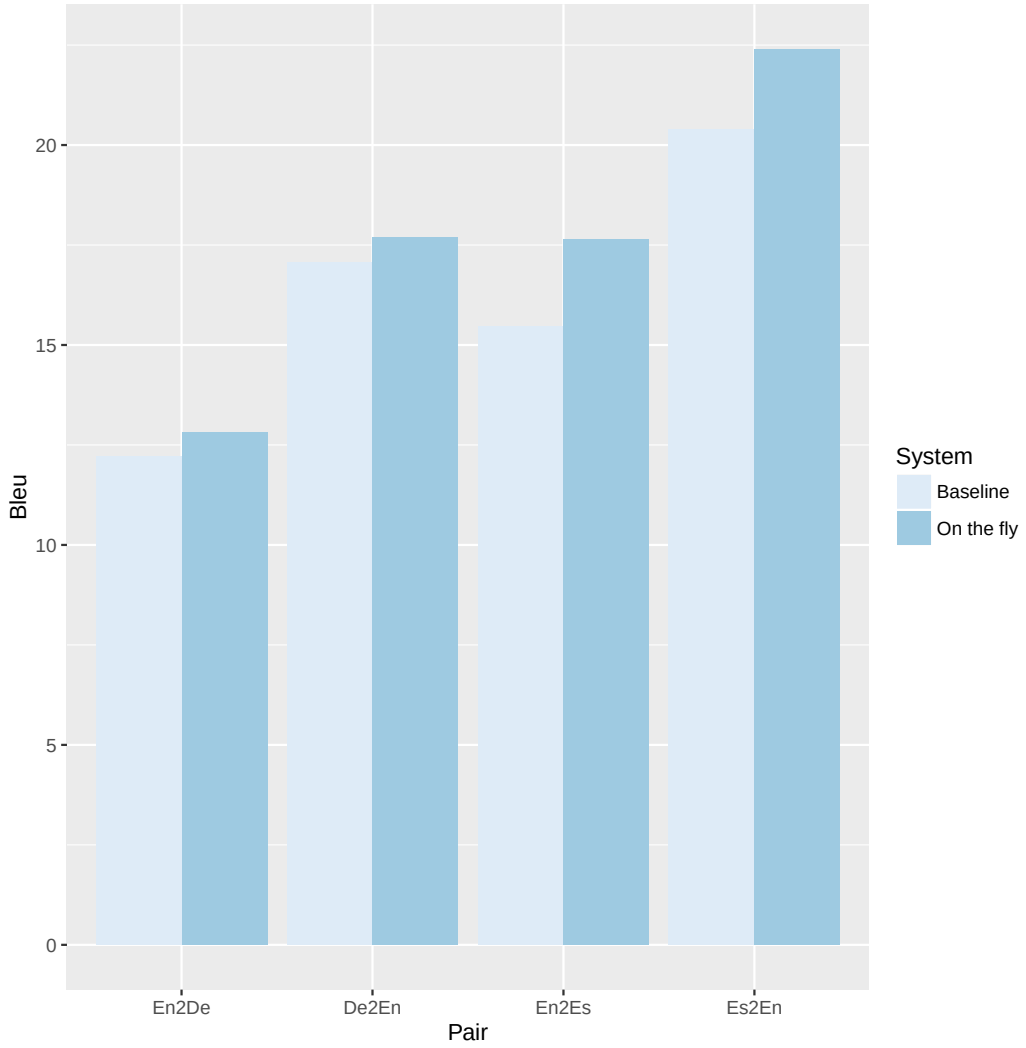

Figure 5.10.: Score of on the fly translation compared to baseline

set of k sentences from the beginning onwards.

The results of this experiment are shown in Figure 7. For this experiment we used the same unseen data as we used for our evaluations before. As mentioned in the beginning of this chapter, the unseen data consisted of the concatenation of four different talks. Therefore we added marker in the figure, to make it visible where a change of talks happened.
This experiment provided us with different interesting observations. Overall, we were now able to see the change of translation quality depending on every added sentence. Interesting is that the behavior of the score change is quite different over the different talks.
Starting with the first talk in the unseen data, we observed an increasing BLEU score in the beginning for the first third of the talk, then a drop in score again, followed by another more continuous improvement up to the end of this talk. For understanding this behavior we looked into this talk which gave a quite interesting surprise. This talk was a talk held by Al Gore, about the climate crisis. But the interesting part of this talk was that in the first third of this talk Al Gore told a more private story unrelated to his main talk. The moment where he switched into the main talk correlates perfectly with the slight drop of BLEU scores probably caused by still the same usage of words, but suddenly applied to a different domain. And even though it was a drop, the system did not degrade compared to the baseline system, only compared to the incrementally improved system which had the previous third of the talk learned. And this degradation was to expect, since the new text was more like a totally new domain.
the second talk showed multiple increases in BLEU followed by some plateaus. Again, a look into the data reflects the structure of the talk, especially the concluding section of this talk can be identified from the BLEU scores alone.
The third talk brought a little surprise to us. Looking at the development of the scores there was no change over the additionally added sentences. This was quite surprising, given the behavior for the other talks. The solution was found when we took a closer look at the talk and its human generated translation data. The problem we found was, that after a few sentence the human translations were not correct and it seemed as if there was a problem with the data for the whole rest of the talk. This however meant, that it was impossible to learn useful new translations from this talk and we were not able to improve on this data, since it was no useful training data at all. This provided the explanation that we did not see an increase in BLEU over this talk.
In addition to that however, this observation proved another point of our work. It proved, that our system was also robust enough to deal with a limited number of false corrections

without losing overall translation quality.

The last talk in this evaluation showed a more steady increase of the BLEU score, which again correlated very well with the structure of this talk. This talk was continuously about aviation and space flights, without changes in the domain and without the major topic changes seen in the first or second talk. This resulted in a continuous improvement of the translation and phrase model for the remaining part, which explains the development of the BLEU scores as we see it in this talk.

Overall the improvements observed in this evaluation tend to slow down in speed once the domain is used for a little longer. This can be attributed to two different interpretations. On the one hand there is the situation, that at later points in a talk, the vocabulary of the speaker for this domain is much better known and in addition to that most of the words which were unknown at the beginning of this presentation should be known by then. This results in less new information in the end compared to the initial additional data.

On the other hand the reason for slower improvements in the BLEU after some time within one domain must also be attributed to the more technical fact of how the evaluation is made, that the remaining set of sentences becomes smaller over time, so the newly learned model updates have the biggest impact on in the beginning, because still a bigger number of sentences is going to be translated. The closer to the end of a talk, the smaller is the set of remaining sentences of this domain. This also results in a slower improvement in the BLEU scores because the number of n-grams affected by a model change is much lower.

Both explanations lead to the conclusion, that the most impact on translation quality will be achieved at the beginning of a talk, or domain switch, while at the end of this talk the improvements are harder to see[1] .

From the behavior of the scores around topic changes the other thing to look out for reducing the cost for doing intra-talk corrections is to look out for these domain changes and keep correcting at the beginning of them, to cover the new language used.

5.4. Dealing with User Feedback

While the above experiments described the handling of good feedback, another question that we had to look at was actually, how to deal with real users data. The TED lectures are already very curated, because they are watched, transcribed and translated by thousands

[1] We would like to mention at this point that the second explanation alone does not justify an early stop of providing corrected sentences, because it is impossible to know what the system is supposed to translate in the future. That means the argument of a smaller set of sentences within the same domain cannot be made.

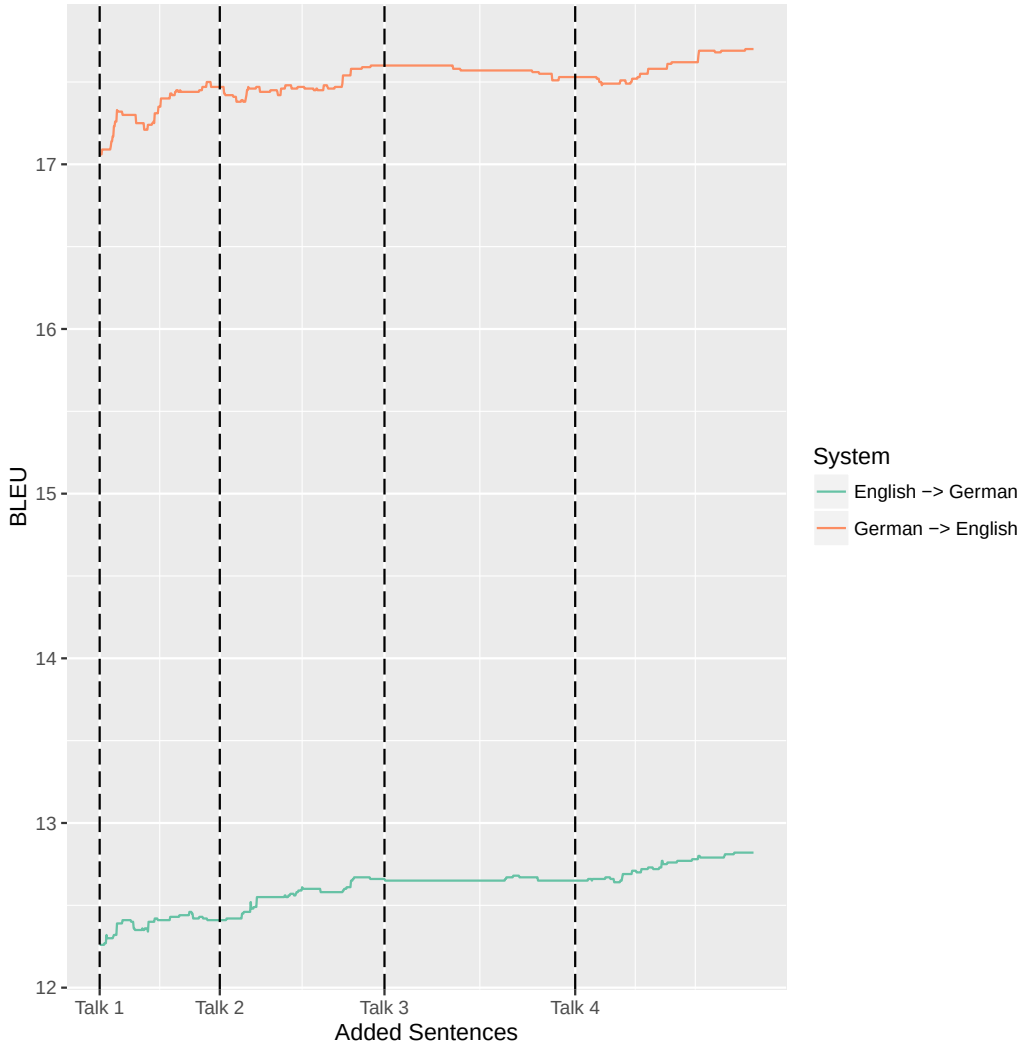

Figure 5.11.: Score after addition of each sentence for in-Domain training

Data: trainCorpus, talks
$train \leftarrow trainCorpus$;
select test talk *test* for testing (crossvalidation);
$resultingTranslation \leftarrow \{\}$;
foreach $i \in length(test)$ **do**
  reset system;
  foreach $sentence \in test$ **do**
    $translation \leftarrow translate(sentence)$;
    $resultingTranslation \leftarrow resultingTranslation \cup translation$;
    if $position(sentence) < i$ **then**
      incremental training of *sentence*;
    end
  end
  $s^{onTheFly}_{test^i} = score(resultingTranslation)$;
end

Algorithm 7: In domain stepwise evaluation

of people. Even if a bad translation exists, it will probably soon be corrected.
In our case however the goal was to use the translation of a user generated translation immediately for the training of the machine translation system. As we described in the leading chapters, we included a classification in our developed system, which decided for every sentence pair, if it was a good one, or if it did not make sense and therefore should not be added to the system. Furthermore we looked at how to support users willing to provide translation corrections.

5.4.1. Identification of Bad Translations

For this experiment we used a much bigger training corpus, including "EPPS", "Subtitles", TED and other resources. The reason was, that we were looking for a corpus that had major problems with noise in the data, which is why we applied our support vector machine based sentence pair selection strategy described above.
This resulted in a reduction of the training data from 5502868 to 4811051 sentence pairs, which meant a reduction of more than 12%. But for this reduction the initial training which was based on TED only was used, which means that even here the additional data was out of domain, reflecting the use we expected in the end. The SVM tagger we used for the decision which sentence pair to use or not was trained on a selection of 3 TED talks,

where we changed the translations of 30% of the sentence pairs, so that we knew which sentences were good translations and which were not.
Then we made a comparison of the original full corpus training, which contains noise and with the smaller, pruned corpus where we hoped to remove some of the noise which was found in the data. Shown below are the scores for the development set and the test set.

Evaluation	German to English	English to German
Training full data	23.66	16.68
Training pruned	23.74	17.0
Unseen full data	16.44	12.39
Unseen pruned	16.17	13.31

Interesting in this experiment is the fact, that we saw a small decline in German to English translation quality measured in BLEU, while there was a huge gain of nearly 1 BLEU when translating from English to German. The answer we found to this question is the fact that from the TED data more than 15.000 sentence pairs were also judged as invalid translations. This is now a problem for the translation from German to English, because of the variable German vocabulary. Missing data from training which is close to the final test case is always a bad thing. On the other hand this problem is not so big for English to German translation simply because of the more restricted English vocabulary.
Still this experiment showed, that the work by (HMNW11) and others can be applied to the online learning task, which showed that a significant reduction of training data from noise is possible with overall improving or at least keeping a steady level of translation quality.

5.4.2. Supporting User Feedback

The last experiment we conducted was looking at the question how to support users who are willing to give feedback. As described, our method for this was to try to finish the beginning of a user provided translation in an autocompletion like setting. Whenever the user typed something, the decoder searched for a translation based on this input. And we measured, how many keystrokes we can save, when the user can

- accept the following proposed word with the tab key
- accept the complete rest of the current proposed autocompletion with the return key

To measure this, we again took the unseen test talks from the TED system and then simulated the typing of the reference translation, assuming the best case scenario that a user immediately sees when he can use one of the two shortcuts in the proposed methods. In the end we then compared the number of keystrokes used in this setting, compared to the total number of keystrokes needed to type the reference translation.
On the small training data set described above, we saw only a comparable small improvement of 1283 keystrokes, while we saw a much better gain of 3463 keystrokes when we used the big background corpus described in the previous section. Even though this is still less than 4 keystrokes per sentence, one should keep in mind, that this was an artificial setup, in which we were aiming to hit one exact translation. In the case of real human translators it is to expect, that even more time can be saved by the fact, that multiple translations are acceptable.
This shows the potential of this method and also indicates that better background models help a lot in this case, making the assistive translation a valuable help for translators.

5.5. Conclusion of Experiments

In our experiments we showed the effectiveness and usefulness of our proposed incremental learning approach. The algorithms we developed were able to modify the underlying models fast enough, that the system was still usable in real time applications, with the benefit of having improvements available immediately. We showed that the model changes achieved improvements in translation quality, even though no full optimization of the scaling factors was made.
To justify the usefulness in a realtime situation, where userfeedback is given during usage, we conducted an experiment where we looked at the translation quality of sentences that had been corrected in the past. Our experiments showed that these corrected sentences were translated nearly perfect. This was an important observation, because it meant instant gratification to a user, when the same sentence worked after correcting it.
For further translation quality improvements, we proposed a combination of incremental and batch based training procedures to further improve the overall performance of the system and minimize the effect of bad scaling factors in the log linear model. The results showed, that such a combination was able to keep up with the traditional batch based training, while keeping the benefits of instant user feedback.
The next set of experiments looked at the question, how such improvements influence intra document translation quality. In a simulated online lecture translation use case we were able to show that our developed methods are able to directly improve the translation

performance for the rest of a talk in which continuous translation feedback was available. We showed that in this situation the translation quality for the rest of the talk increases, not only by being able to translate previously unknown words, but also by a better phrase selection.

In an extension of this evaluation, we also showed the development of the improved translation quality in relation to the number of corrected sentences. This lead us to the observation, that the correction of sentences in the beginning of topics covered in a talk is more important than corrections in the rest of the talk. This was caused by adaptation to the language of the domain and also by the learning of domain specific previously unknown words. We were also able to show that a limited amount of incorrect training data did not degrade the translation performance.

The last series of experiments looked at user behavior for translation feedback. The first of these experiments showed that our proposed method of noise reduction in the training data also was able to reduce the number of bad training examples, thus reduced the overall training corpus size significantly, while keeping comparable and in some situations even improved translation quality.

And the second experiment in this series showed that using machine translation as an aid to users willing to correct sentences can be of actual help to reduce the work needed.

6. Lecture Translation System

In this chapter we describe how our developed architecture can be used as a framework for a lecture translation environment. We tested our own translation engine with the previously described mechanisms for online incremental improvements in this setup, but also proved in the context of the EU-Bridge project the easy usage and integration of other components into our proposed architecture. We did this by showing a full end to end lecture translation environment where speech recognition and machine translation components developed at the Karlsruhe Institute of Technology and other partners were used and attached to our proposed framework. We describe how the components interact with each other and the interfaces we developed to interact with the outside world.

In the communication, the mediator we developed and described in Chapter 4 played the central role. It was the component, which managed the routing and the finding of the correct components to fulfil service requests.
In addition to that, the mediator took care of sending the messages that needed to be stored to a long term storage unit, which then took care of distributing this information to components which needed it for updating their models, or for presentation purposes, when students wanted to revisit past lectures at a later time.
For corrections, we included our proposed mechanism for noise removal, which automatically identified bad training examples and prevented that our incremental model update in the machine translation modules learned from these bad examples. For every corrected sentence pair, we also logged, which user provided this example, which allowed us to distinguish among the users in the future and on the one hand identify malicious users, who don't provide good examples, but on the other hand it also allowed us to identify those

users, who provided good training examples, and therefore one can think of asking these users to provide translations for situation, where no other user provided one.
The following sections describe the additional components we developed, to make this a complete end to end system usable in a lecture translation environment.

6.1. Presenter

A very important task for the lecture translation system was that end users should not be bothered with details about the background architecture and the whole experience of using it should be as simple as possible. The idea was that a presenter did only spent very short time in setting up the environment and the rest is done full automatically.
For this purpose we developed a standalone interface for the presenter which provided an abstraction from the underlying problem to a simple recording solution. A picture of the interface can be seen in Figure 6.1 and it shows the complete interface for the presenter that was used in the lecture translation system. The presenter had the possibility to enter a name for the talk that was presented, and a short description. In addition to that it was also possible to provide the system with a domain of the presentation. This allowed a better selection of backend engines suitable for the task, because the underlying fingerprint mechanism used the domain information in the matching process. The language the presenter gave the presentation in, was selected beforehand and like all additional information given on startup it was persistent over multiple sessions, so that the presenter only had to change input settings when he needed, but in the optimal situation the presentation could start the moment the presenter started the session.
For the connection to the background archive, we allowed the presenter to insert a username, which allowed organizing, maintaining and sharing of the lecture after the presentation had finished.
Since not all what is said during a lecture should be recorded and stored, the presenter was also able to start and stop the recording - and therefore the recognition and translation at any time. This was especially handy for setting up the whole system before the actual presentation started, but also for ”off the record” parts.

The connection of the presenter client to the infrastructure was established using the c binary library. On the side of the mediator the connection spawned a new client process handling the connection as described before.
In addition to that the previously described REST-Api also provided an interface to the architecture, which allowed even the streaming of prerecorded videos or audios into the

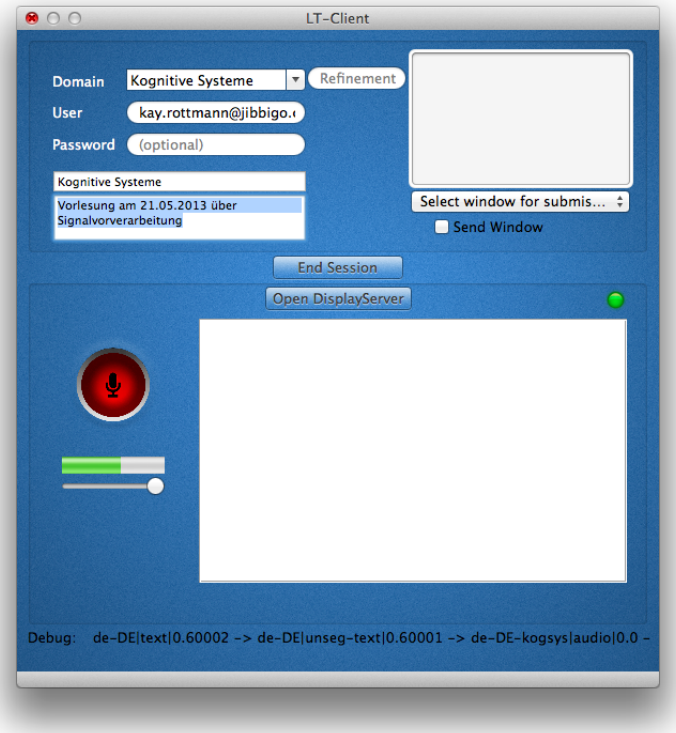

Figure 6.1.: User interface for the client as seen by the presenter

system, enabling students to follow in a different language and read the transcripts.
As a proof of concept we also built a HTML5 client, which provided the complete same information as the standalone client, but did not require an installation by the presenter. This client allowed the usage of the lecture translation system on different plattforms, as long as a browser which supported the necessary HTML5 features was available (which was at the time of writing the case for all major browser platforms).

6.2. Result Display

The goal of our system was to be used in university classes. So the presentation of results had to be available for multiple students at the same time. Those students did not necessarily speak the same languages, making the usage of a single presentation display for every student unfeasible. For that reason we developed a webserver that connected to the mediator as a display server process. A website allowed us to keep the development of the presentation on our own side without the need of anyone to install tools to follow a lecture. It also removed the necessity to develop for different operating systems and support of different versions of the protocol. And finally this also allowed us to target the audience on mobile devices, without the need of writing dedicated apps, as long as a stan-

dard internet browser was available on that platform. This last part was really important, because most of the students have a mobile device at hand when they come into lectures, while not necessarily everyone has a notebook.
The development of the website furthermore enabled us to keep multiple students connected with different language needs, while only a single display process was collecting all the different results. This allowed a reduction of the workload on the mediator which was an important preparation for the lecture translation architecture to be used in larger scale at universities. Students who followed a lecture online by opening the lecture presentation on the webserver were able to select the corresponding lecture they wanted to get the translations for, and they could change the language into the language they wanted to have the content translated into on demand. On the backend side, the webserver was maintaining all currently needed languages for every active lecture and whenever a new language was needed, or a language was not needed anymore for a lecture, the corresponding *XML* message was sent to the connected display process on the mediator side. The mediator then took care of freeing or allocating the needed resources, changing the set of messages which were sent to the display web server by this action.
When a student opened up the website for following the lecture translation project, we presented a list of the currently active lectures, as shown in Figure 6.2. From this list, the student was able to select a lecture and login to follow the live presentation. We also developed a simple protection mechanism, which used a password for the lectures, so that the presenter was able to restrict the audience to a smaller circle if wanted.

Once logged in, the student was facing a webpage which can be seen in Figure 6.3, where on top the current running lecture was output and on the bottom of the page the student was able to select a language into which this lecture was translated. In addition to the name of the language, the drop down menu also provided a description of the stream the student was about to follow (in the given example, it is a stream called ”speech”). This was reflecting the usage of multiple input sources which we allowed in the client processes, so that students had the possibility to also follow different input streams. We also supported the transmission of image data in this architecture. When a presenter enabled the submission of images in addition to the audio, students were able for example to look at the slides, that were shown. This was especially developed with remotely following students in mind who were not present in the lecture hall at the time of the presentation.

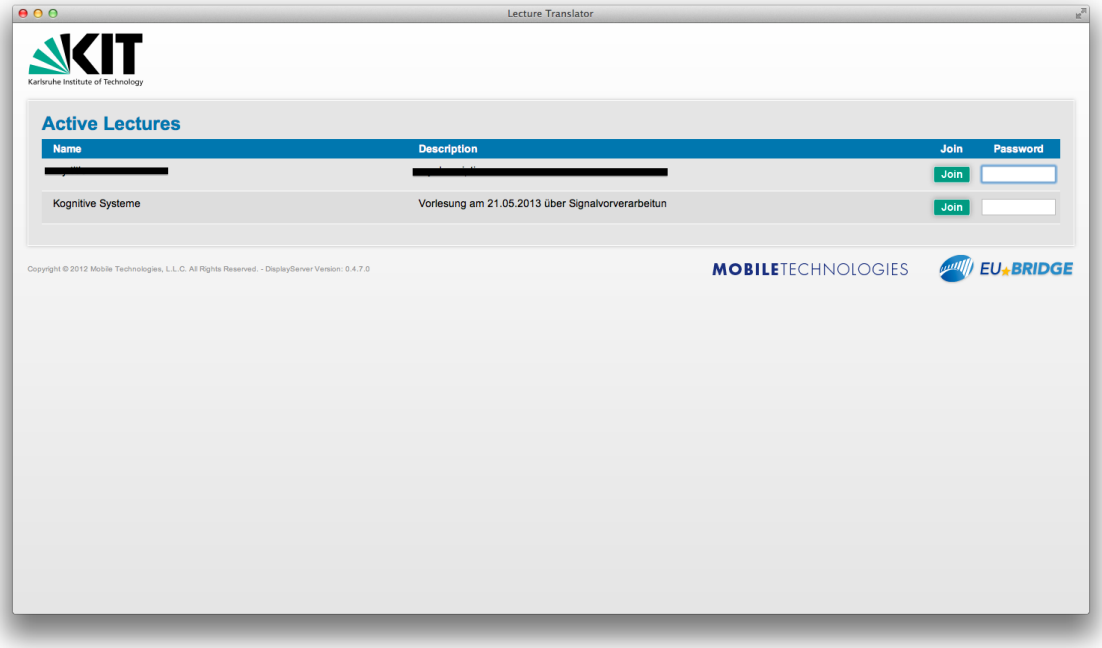

Figure 6.2.: Interface of Display which shows currently running lectures a user can join

6.3. Archiving and Crowdsourcing of Lectures

The sections above described the components which were visible to students and presenters during a lecture, but our architecture was also built for students as a learning tool and crowd sourcing feedback for transcription and translations of past lectures to improve the underlying machine translation system as described in Chapter 5.

We described in Chapter 4 that the mediator based architecture directly stored the lectures in a database, and this section now describes how this was used as a basis for the crowdsourcing efforts.

With the possibility of reviewing past lectures, students got a better opportunity to learn the curriculum of classes, and therefore we expected students to be willing to use such systems resulting in a situation where students were willing to give feedback and we were able to learn automatically from these translations.

Our way of achieving this was that we developed another server, which served an archive website to the students, where they were able to see past lectures and listen to them again while seeing the transcripts or the translations. The interface the students normally faced is seen in Figure 6.4. When we developed the backend archive, we also developed an indexing mechanism, which indexed the textual content of a presentation, as soon as text output was generated (in whatever language). This allowed students to search within lectures.

Figure 6.3.: Presentation view of a live lecture

In the Figure 6.4 we show, the situation where a student is seeing a list of all lectures, where the German word *Multiplikation* is used - the search itself was case insensitive. The student can then select the desired lecture and read and listen to it again.
For the maintainer of a lecture, the view of the lectures he owned was similar, with a difference, that it was possible to delete or share lectures which is reflected in the two buttons next to the lecture link in Figure 6.4. In this case, the meaning of sharing a lecture was, that the owner was able to generate a direct link to this past lecture, which allowed users to listen and read the content, even without a user account or password. This was helpful for situations, where presenters taught the same class multiple times over years, but they did not want the content to be always available to all students, but just for themself. A direct link was created in those cases, and when the presenter did not want this direct access anymore, he was able to simply delete this direct link to prevent future access to those who used this link, but keeping the content of the lecture still available in the database for future usage.

When a lecture was selected from this list, another view was opened in the browser and the user was able to see the transcription and translation of the audio. The play button on the left side of the screen allowed the user to start the audio again, which caused our system to stream the recorded audio back to the browser of the user, so that it was possible to hear the lecture again. Since we wanted to motivate the user to correct errors our translation or transcription made, we also highlighted as an additional help for the user the currently heard transcription and audio. This situation is shown in Figure 6.5.
In this presentation mode, the text of the two columns, is scrolling according to the

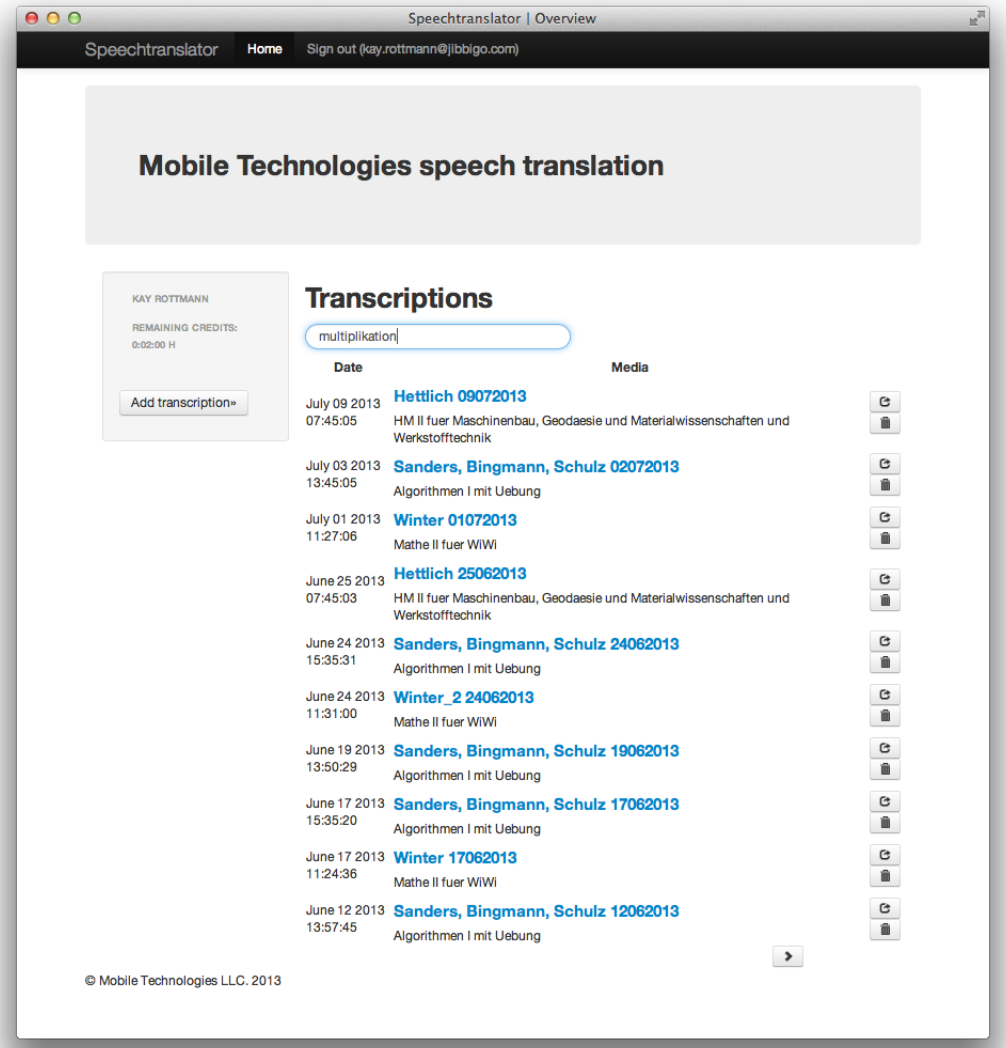

Figure 6.4.: Overview over lectures available in the lecture archive containing the word ”multiplikation”

currently heard audio, so that the user does not have to interact with the website, while following the recording along.

When the user of this system however observed an error in either the transcription or translation, he was able to stop the audio playback with any key press and directly make a correction to the current segment. We took great care this was possible without any mouse interaction at all, so that a user who was willing to put a lot of effort into the correction was able to work with the keyboard alone making the corrections faster.

In Figure 6.6, we show the screen, the user was facing the moment a correction was happening. In this correction screen, it was possible to listen to the corresponding audio

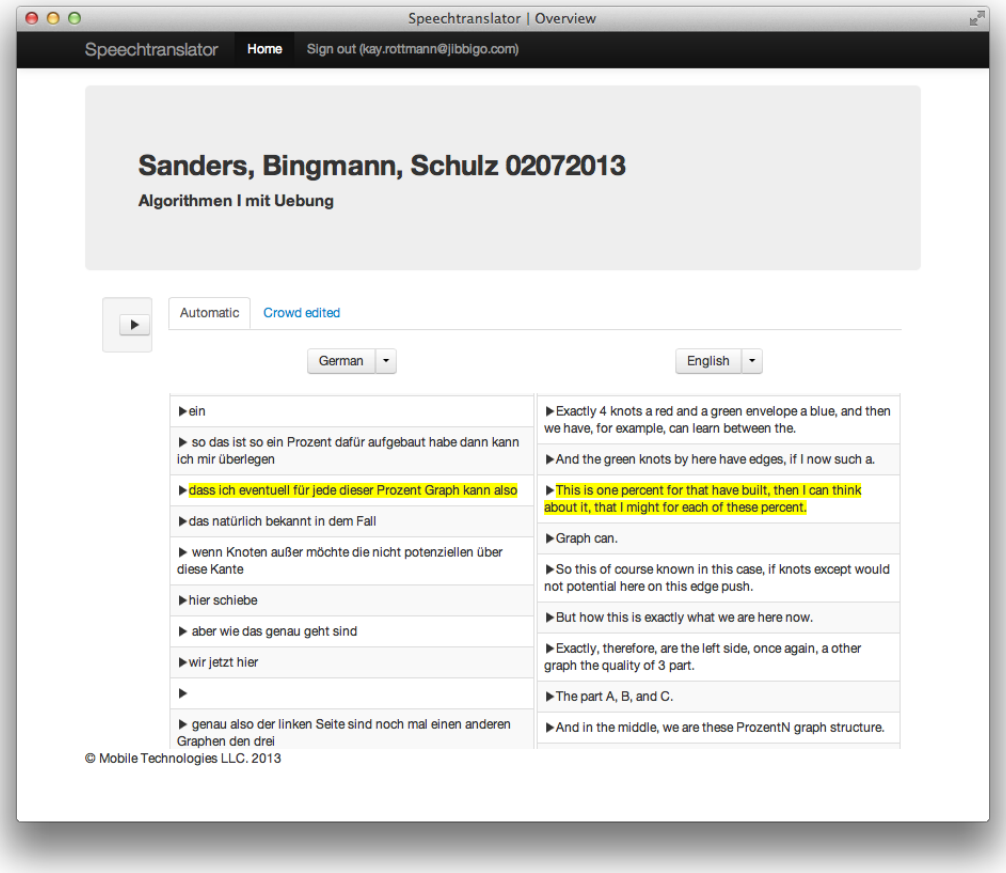

Figure 6.5.: Display of the content in a lecture that was saved in the archive

of this segment alone and seeing the automatic transcription or translation result. In addition to that, the dialogue also showed corrections, which were made by other users. The intention of this was, to allow users to agree on one reference more easily, without having the trouble of dealing with multiple same corrections that only differed in typos or small punctuation differences. To restrict the cognitive overload users are facing in this screen we reduced the number of shown alternative corrections already made to a maximum of 5.

One problem with the correction of translations however is - as stated in Chapter 3 - the problem, that the correction of a wrong translation typically takes longer than rewriting a translation from scratch. For that reason we implemented the mechanisms described in Chapter 4, where a partial input of a user provided translation is extended by machine translation and this is shown to the user as an auto complete. In (ABC+14) it is shown that those mechanisms provide a time reduction in the correction process and our experiments showed the same results.

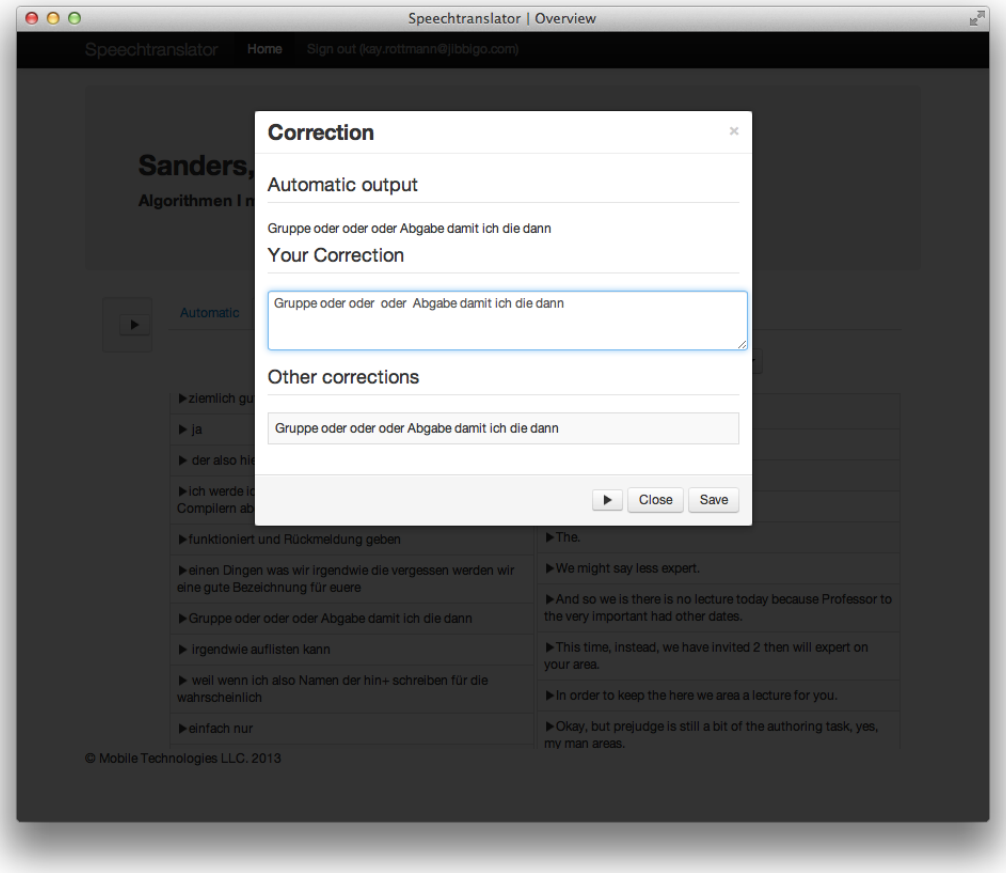

Figure 6.6.: Dialogue to obtain correction of translation or recognition

As pointed out in Chapter 4, the reduction of noise is also a very important part of the correction process. This is for two reasons. On the one hand, we expected people to give corrections, without being fully able to do so. This could result in wrong transcriptions or translations and a reason for this might be a user who is looking at a bad translation result which changed the meaning, and this user than provides a more fluent version of this misleading translation.

On the other hand, we also expected a small percentage of people who try to sabotage such a system. Therefore we thought it was necessary to identify these situations and anticipated them by developing counter measures. In this setting we used the previously described mechanism of identifying noise using a SVM based approach and used this to identify user made corrections as good or bad ones. When our system classified something as noise, this correction was not immediately fed back into the learning system, but it was stored in the archive with a marker indicating that this needed another check. If a user continued to provide ”corrections” which were classified as noise, we did not accept any further corrections from that user at all, until they were manually verified. We imple-

mented a simple threshold of more than 80% of rejected translations. However since we were unable to test out and experiment in a non artificial environment the usage of the implemented features, this is a field open for future work.

In a second step we expanded our system to add more external resources as well and we developed mechanisms to import data from elsewhere than the live lecture translation system. By the usage of the underlying REST-Api as described in Chapter 3, we allowed the users to also add videos and audio files, or even videos from publicly available URLs or youtube and vimeo video platforms for transcription and translation purposes. The interface for the user is shown in Figure 6.7. Data imported over this interface got transcribed and potentially translated and was available to the user in the lecture overview with the provided description elements entered on this side.

The transcription and translation itself was done in a server side process, which sent out an email notification to the user, once the requested transcript was ready. To deal with abuse of such a mechanism (by constantly importing external resources), we implemented an accounting mechanism, which assigned only a limited amount of processable minutes to a user. From a business perspective this mechanism was easy to extent to a payable service.

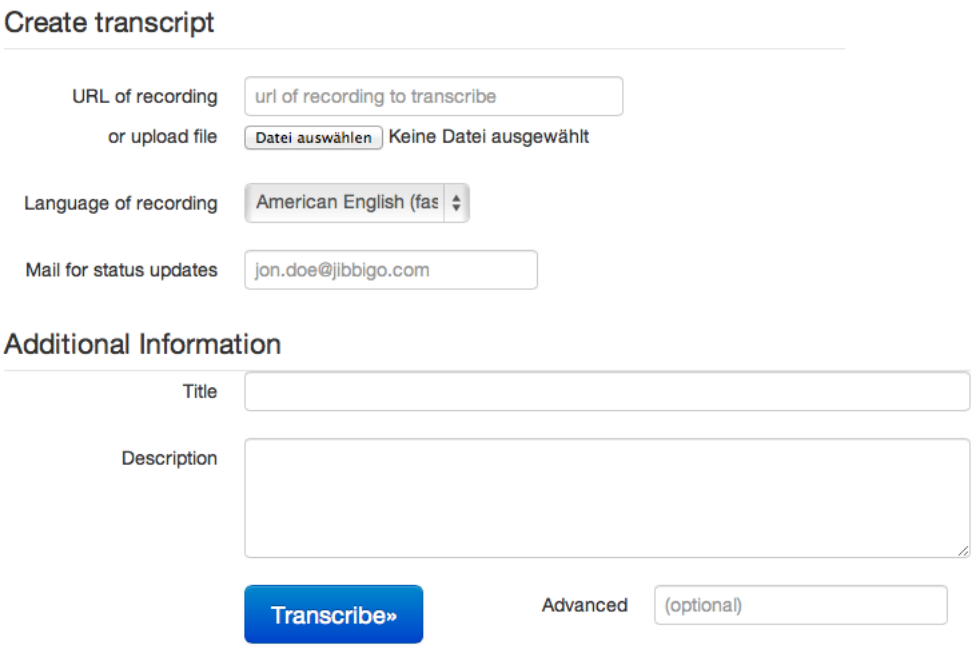

Figure 6.7.: Import of external resources

This transcription of external resources also played very well with the mechanism of sharing direct links to such resources. So it was easily possible to give limited time access to the

resource within this system and rejecting the access, when the owner of the resource decided to.

Instead of using the website, the underlying use of the REST-Api also provided a direct programming interface to developers, who then were able to write their own applications to connect to the architecture and work on lectures or external resources.

6.4. Conclusion

In this chapter we presented the complete end to end system we developed and deployed which proved the usability and functionality of the work presented in the previous chapters.

We developed techniques to provide students in a lecture setting with real time translations, allowing foreign students to follow a lecture in their own language. We furthermore developed a system that allowed students to revisit past lectures, with the possibility of giving feedback on corrections to the automatic transcriptions and translations. This feedback was then depending on a quality check either used to immediately update translation models, or it was hold back because of bad quality. We also integrated the developed mechanisms for autocompleting translation feedback, to make it easier for people to give feedback and support them in this task.

With the introduction of user accounts, the possibility of sharing content to users without accounts and a credit based transcription interface for content accessible on the web, we also opened up this tool for broader usage beyond the lecture hall.

7. Conclusion

In this thesis we pointed out the need for fast updates to machine translation models and the problem of current batch training approaches and their long update cycles.
This lead us to develop a distributed system and the algorithms for fast incremental updates to statistical phrase based machine translation systems in a distributed speech translation setup. The algorithms we developed, allowed us to have our machine translation models being updated while machine translation was actively in use, making crowdsourcing for corrected translations a better experience for the users who are willing to do this, because they were able to immediately see that their previously made corrections were used in the future.
To make this possible, we had to find efficient ways of storing and looking up the data needed for the model modification and we had to develop the mechanisms to update models in a fast and efficient manner. Usually the building of models for phrase based statistical machine translation systems relies on huge training corpora and involves multiple hours of compute time to build, our work showed that we were able to learn new information for models and update them in milliseconds.
We then developed a distributed system which can be used in speech translation tasks where crowdsourcing is one aspect, while the deliverance to multiple users at the same time was another important point. Since we focused on the usage in a lecture translation environment, we especially looked at the requirements of easy extendability to other language pairs which need to be translated and to the ability of being able to support multiple users at the same time, while staying dynamic enough to be able to react to changes during a speech translation session.
The system we developed was not restricted to speech or text data only, but we also showed

that the same underlying principles work for image data as well. Allowing for example the translation of slides during presentations, or also to apply the whole framework to completely different not speech related tasks.

We also developed the necessary requirements for allowing the learning from crowdsourced data, and we developed mechanisms where corrections were able to be used beyond the currently looked at language pairs the correction was generated for.

Since the interaction of other people with our system played an important role for the crowd sourced feedback, we also developed algorithms to support people the moment they were giving feedback by having auto completion for translations the user was typing and we ensured with automatic evaluation of user feedback, that bad translations were identified and not used for the model improvement. We also evaluated our developed algorithms on crowd sourced data (the TED lectures with their crowd sourced translations) and showed that we were able to achieve significant gains in translation quality by usage of the crowd sourced feedback. We showed that our method is fast enough to allow on the fly improvements in a realtime live translation setup and in comparisons with the batch training approach, we were able to continuously achieve similar translation quality by the usage of hybrid techniques, mixing the incremental model training with repeated batch training.

In further experiments we also looked at the micro level of the changes in the translations generated by our incremental models. We proved that our incremental model was able to improve the translation quality even for the rest of a single talk and we saw that these improvements were beyond the simple learning of previously out of vocabulary words, which would be possible with simpler approaches. In these evaluations we also showed that the most changes in translation quality happen when corrections are provided for the beginning of a paragraph dealing with a new domain.

For the identification of bad translations, we also showed, that we were able to reduce the training corpus size by a significant number of sentence pairs, while the translation quality did remain within the same range and in one case even improved over the original translation score. This showed that filtering out bad training examples was able to not only save computation time but in addition to that was able to improve the overall quality.

In the end we combined the developed techniques and built a reference implementation for in the form of a distributed lecture translation system.

We developed the user facing components like websites for the crowd sourcing, presentation of live lecture translation services, interfaces for the automatic transcription of arbitrary videos and the tools for presenters in a lecture environment. We combined these with the developed architecture and thus built a lecture translation system for students of the Karl-

sruhe Institute of Technology, which was presented in 2012. The underlying infrastructure is still in regular use at the KIT.

7.1. Future Work

While it was difficult to work on crowd sourced data in a lecture environment during the creation of this thesis, it is now possible with the regular usage of our system in lectures at the Karlsruhe Institute of Technology.

With this work we built the foundation for continuous research and have the opportunity to prove that our made assumptions also hold for the user generated content in a lecture environment. This was not possible in this thesis, because of the lack of data and therefore we simulated this by using TED data, which is very similar to the situation we face at university lectures and was available as thousands of crowd sourced translations and transcripts. Something still not available in the lecture translation setting at the KIT and it will take years to generate a corpus as large as the one available in the TED data.

Related to this is also the idea of combining past research on active learning(Amb11) with our method of focusing on the real usage. The question that remains open, is if there is a way, to push users into correcting the most important sentences, to minimize the overall work needed, or with other words, if the corrections would be restricted to $x\%$ of all sentences, which sentences should be selected and is this possible to do online, highlighting to users at the moment of generation that this sentence should be corrected.

For the incremental training, we were able to make it possible on normal desktop or notebook computers with just 2 GB of RAM. But we belief, that with focusing on even more restricted settings, we will be able to improve the algorithms and efficiency even more and we think, that we can develop similar algorithms to be used on translation engines running on todays smartphones, without any loss compared to our current solution.

For the architecture of the lecture translation system we have plans to make it also possible to automatically incorporate nodes which collect data from different sources (fan in). This is already possible by manual description of the graph, but we plan to develop automatic support for finding data paths with such nodes, which then optimize for the best possible outcome by applying systemcombinations of the right systems at the right moment.

Another point of research we would like to further investigate is the question of sharing and handling multiple users and their feedback. We would like to share feedback between groups, while other users adapt their systems towards their own requirements. Here we are also interested in how these differences can be used among these groups and how improvements can be used for general model updates without raising privacy concerns by

making things available which should not have been available to other users.

With a long term deployment of our incremental learning, we will also be able to run more experiments on the social aspects on users confronted with such a system. Interesting questions to look at are

- Can we measure the impact of such a system on the overall performance of students attending a class?
- How does the base translation quality influence the willingness of users providing feedback?
- How does a good ”karma-system” look like which motivates users to give more feedback?
- How many saboteurs are to expect in such a system?
- Will the user rating system of translations be enough to eliminate the influence of saboteurs?

These are all interesting questions which are important for the impact of such a system. We hope that the full deployment of an automatically improving lecture translation system is able to help answering these questions.

Bibliography

[ABC+14] V. Alabau, C. Buck, M. Carl, F. Casacuberta, M. Garcıa-Martınez, U. Germann, J. González-Rubio, R. Hill, P. Koehn, L. Leiva *et al.*, "Casmacat: A computer-assisted translation workbench," *EACL 2014*, p. 25, 2014.

[Amb11] V. Ambati, "Active learning and crowdsourcing for machine translation in low resource scenarios," Ph.D. dissertation, University of Southern California, 2011.

[AMCb+05] A. Axelrod, R. B. Mayne, C. Callison-burch, M. Osborne, and D. Talbot, "Edinburgh system description for the 2005 iwslt speech translation evaluation," in *In Proc. International Workshop on Spoken Language Translation (IWSLT*, 2005.

[AOMSC10] V. Alabau, D. Ortiz-Martínez, A. Sanchis, and F. Casacuberta, "Multimodal interactive machine translation," in *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction.* ACM, 2010, p. 46.

[AVW+96] J. Armstrong, R. Virding, C. Wikstr, M. Williams *et al.*, "Concurrent programming in erlang," 1996.

[BHF09] N. Bertoldi, B. Haddow, and J.-B. Fouet, "Improved minimum error rate training in moses," *The Prague Bulletin of Mathematical Linguistics*, vol. 91, no. 1, pp. 7–16, 2009.

[BL05] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005, pp. 65–72.

[BPPM93] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: parameter estimation,"

Comput. Linguist., vol. 19, pp. 263–311, June 1993. [Online]. Available: http://portal.acm.org/citation.cfm?id=972470.972474

[BSC+14] N. Bertoldi, P. Simianer, M. Cettolo, K. Wäschle, M. Federico, and S. Riezler, "Online adaptation to post-edits for phrase-based statistical machine translation," *Machine Translation*, vol. 28, no. 3-4, pp. 309–339, 2014.

[CB09] C. Callison-Burch, "Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, 2009, pp. 286–295.

[CBF+14] M. Cettolo, N. Bertoldi, M. Federico, H. Schwenk, L. Barrault, and C. Servan, "Translation project adaptation for mt-enhanced computer assisted translation," *Machine Translation*, vol. 28, no. 2, pp. 127–150, 2014.

[CDLS11] J. H. Clark, C. Dyer, A. Lavie, and N. A. Smith, "Better hypothesis testing for statistical machine translation: Controlling for optimizer instability," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 2011, pp. 176–181.

[CF12] C. Cherry and G. Foster, "Batch tuning strategies for statistical machine translation," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2012, pp. 427–436.

[CFNV08] F. Casacuberta, M. Federico, H. Ney, and E. Vidal, "Recent efforts in spoken language translation," *Signal Processing Magazine, IEEE*, vol. 25, no. 3, pp. 80–88, 2008.

[CG91] K. W. Church and W. A. Gale, "A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams," *Computer Speech & Language*, vol. 5, no. 1, pp. 19–54, 1991.

[CG96] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1996, pp. 310–318.

[Dod02] G. Doddington, "Automatic evaluation of machine translation quality using n-gram co-occurrence statistics," in *Proceedings of the second international conference on Human Language Technology Research.* Morgan Kaufmann Publishers Inc., 2002, pp. 138–145.

[Eid12] V. Eidelman, "Optimization strategies for online large-margin learning in machine translation," in *Proceedings of the Seventh Workshop on Statistical Machine Translation.* Association for Computational Linguistics, 2012, pp. 480–489.

[Erl06] T. Erl, *Service-oriented Architecture: Concepts, Technology, and Desing.* Pearson Education India, 2006.

[ES08] M. Eck and T. Schultz, "Developing deployable spoken language translation systems given limited resources." Ph.D. dissertation, Karlsruhe Institute of Technology, 2008.

[FKJ06] G. Foster, R. Kuhn, and J. H. Johnson, "Phrasetable smoothing for statistical machine translation," 2006.

[FT02] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology (TOIT)*, vol. 2, no. 2, pp. 115–150, 2002.

[FWK07] C. Fügen, A. Waibel, and M. Kolss, "Simultaneous translation of lectures and speeches," *Machine Translation*, vol. 21, no. 4, pp. 209–252, 2007.

[FWS+01] C. Fügen, M. Westphal, M. Schneider, T. Schultz, and A. Waibel, "Lingwear: a mobile tourist information system," in *Proceedings of the first international conference on Human language technology research.* Association for Computational Linguistics, 2001, pp. 1–5.

[GLQH11] Q. Gao, W. Lewis, C. Quirk, and M.-Y. Hwang, "Incremental training and intentional over-fitting of word alignment," *Proceedings of the Thirteenth Machine Translation Summit*, pp. 106–113, 2011.

[GV08] Q. Gao and S. Vogel, "Parallel implementations of word alignment tool," in *Software Engineering, Testing, and Quality Assurance for Natural Language Processing.* Association for Computational Linguistics, 2008, pp. 49–57.

[HDO+98] M. A. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *Intelligent Systems and their Applications, IEEE*, vol. 13, no. 4, pp. 18–28, 1998.

[HMNW11] T. Herrmann, M. Mediani, J. Niehues, and A. Waibel, "The karlsruhe institute of technology translation systems for the wmt 2011," in *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2011, pp. 379–385.

[Kas65] T. Kasami, "An efficient recognition and syntaxanalysis algorithm for context-free languages," DTIC Document, Tech. Rep., 1965.

[KCS08] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with mechanical turk," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 453–456. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357127

[KHB+07] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens *et al.*, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, 2007, pp. 177–180.

[Kni99] K. Knight, "Decoding complexity in word-replacement translation models," *Computational Linguistics*, vol. 25, no. 4, pp. 607–615, 1999.

[Koe05] P. Koehn, "Europarl: A Parallel Corpus for Statistical Machine Translation," in *Machine Translation Summit X*, Phuket, Thailand, 2005, pp. 79–86.

[Koe09] ——, "A web-based interactive computer aided translation tool," in *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*. Association for Computational Linguistics, 2009, pp. 17–20.

[KOM03] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 48–54. [Online]. Available: http://dx.doi.org/10.3115/1073445.1073462

[Lea10] N. Leavitt, "Will nosql databases live up to their promise?" *Computer*, vol. 43, no. 2, pp. 12–14, 2010.

[LIS68] P. M. Lewis II and R. E. Stearns, "Syntax-directed transduction," *Journal of the ACM (JACM)*, vol. 15, no. 3, pp. 465–488, 1968.

[LWER10] I. Lane, A. Waibel, M. Eck, and K. Rottmann, "Tools for collecting speech corpora via mechanical-turk," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, 2010, pp. 184–187.

[MM05] D. S. Munteanu and D. Marcu, "Improving machine translation performance by exploiting non-parallel corpora," *Computational Linguistics*, vol. 31, no. 4, pp. 477–504, 2005.

[Och02] F. J. Och, "Statistical machine translation: from single-word models to alignment templates," Ph.D. dissertation, Bibliothek der RWTH Aachen, 2002.

[Och03] ——, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2003, pp. 160–167.

[OMGVC10] D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta, "Online learning for interactive statistical machine translation," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 546–554.

[ON02] F. J. Och and H. Ney, "Discriminative training and maximum entropy models for statistical machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002, pp. 295–302.

[ONJN03] F. J. Och, H. Ney, F. Josef, and O. H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, 2003.

[Pre07] W. H. Press, *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[PRWjZ02] K. Papineni, S. Roukos, T. Ward, and W. jing Zhu, "Bleu: a method for automatic evaluation of machine translation," 2002, pp. 311–318.

[RVW06] K. Rottmann, S. Vogel, and A. Waibel, "Design of a decoder for statistical machine translation using inverse transduction grammar," Studienarbeit, Karlsruhe Institute of Technology, 2006.

[S+02] A. Stolcke *et al.*, "Srilm-an extensible language modeling toolkit." in *INTERSPEECH*, 2002.

[SDS+06] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, "A study of translation edit rate with targeted human annotation," in *Proceedings of association for machine translation in the Americas*, 2006, pp. 223–231.

[SLZ12] A. Saluja, I. Lane, and Y. Zhang, "Machine translation with binary feedback: a large-margin approach," in *Proceedings of The Tenth Biennial Conference of the Association for Machine Translation in the Americas, San Diego, CA, July*, 2012.

[SPF+11] S. Sakti, M. Paul, A. Finch, S. Sakai, T. T. Vu, N. Kimura, C. Hori, E. Sumita, S. Nakamura, J. Park *et al.*, "A-star: Toward translating asian spoken languages," *Computer Speech & Language*, 2011.

[Tat12] "Open collaborative multilingual "sentence dictionary"," http://tatoeba.org, Tatoeba.org, 2012.

[TED13] "TED conference, idea's worth spreading," http://www.ted.com/, TED Talk, 2013.

[Tie12] J. Tiedemann, "Parallel data, tools and interfaces in opus," in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, N. C. C. Chair), K. Choukri, T. Declerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, Eds. Istanbul, Turkey: European Language Resources Association (ELRA), may 2012.

[VNT96] S. Vogel, H. Ney, and C. Tillmann, "Hmm-based word alignment in statistical translation," in *Proceedings of the 16th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1996, pp. 836–841.

[VV05] A. Venugopal and S. Vogel, "Considerations in maximum mutual information and minimum classification error training for statistical machine translation," in *Proceedings of the Tenth Conference of the European Association for Machine Translation (EAMT-05)*. Budapest, Hungary May, 2005, pp. 30–31.

[Wai96] A. Waibel, "Interactive translation of conversational speech," *Computer*, vol. 29, no. 7, pp. 41–48, 1996.

[WP67] D. J. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, 1967.

[Wu97] D. Wu, "Stochastic inversion transduction grammars and bilingual parsing of parallel corpora," *Computational linguistics*, vol. 23, no. 3, pp. 377–403, 1997.

[WW97] Y.-Y. Wang and A. Waibel, "Decoding algorithm in statistical machine translation," in *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1997, pp. 366–372.

[You67] D. H. Younger, "Recognition and parsing of context-free languages in time< i> n</i>< sup> 3</sup>," *Information and control*, vol. 10, no. 2, pp. 189–208, 1967.

[ZCB11] O. Zaidan and C. Callison-Burch, "Crowdsourcing translation: Professional quality from non-professionals." in *ACL*, 2011, pp. 1220–1229.

[ZN03] R. Zens and H. Ney, "A comparative study on reordering constraints in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2003, pp. 144–151.

[ZON02] R. Zens, F. J. Och, and H. Ney, "Phrase-based statistical machine translation," in *KI 2002: Advances in Artificial Intelligence*. Springer, 2002, pp. 18–32.

[ZV02] B. Zhao and S. Vogel, "Adaptive parallel sentences mining from web bilingual news collection," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 2002, pp. 745–748.

[ZV07] Y. Zhang and S. Vogel, "Pandora: A large-scale two-way statistical machine translation system for hand-held devices," in *Proceedings of MT Summit XI*, Copenhagen, Denmark, Sep 2007.

A. Messages in the Speech Translation Architecture

For the message exchange between the components in the speech translation architecture we used XML messages. There are mainly two reasons why we decided for the usage of XML.

- easy access for third parties to build their own service without the need for integrating a library or implementing a different message schema
- easy to inspect and debug because messages are - except for the binary blobs - human readable.

A.1. Client Connections

When a connecting application spawns a new client process in the architecture by connecting to the client port, the architecture will directly send a message to the client announcing that it is connected, and it will also send a so called challenge in this message which must be solved by the connecting application to verify that this is a valid connection which is a simple way to prevent abuse of the system. The challenge is encapsulated in the `description` argument.

```
<status type="connect" description="12345"/>
```

The connecting application has to find the correct answer to the number in `description` and send the description of the session it provides as an answer, where it also provides

detailed information about the following session it opens up. `sessionid` is the argument, which is the correct "answer" to the previous request. and `password` is an optional argument, used for cases where a session should be protected by a password from access.

```
<flow sessionid="54321" password="secret" logging="true">
  <sessioninfos>
    <sessioninfo
     language="en"
     name="Title"
     description="description"/>
  </sessioninfos>
</flow>">
```

This initial response also includes the `sessioninfos` which give a description of the session. For example the name of the class would be a good `name` and the content of this specific lecture could be a reasonable `description`. Since we developed this system mainly for the use in a translation system, we also allowed the description in multiple languages, so that the creator of a session can provide the information in different languages. This would then be done by providing multiple `sessioninfo` fields with different `language` arguments.

It is possible to extend this in the future to also rely on machine translation for the languages not provided in the `sessioninfos` part.

The client will then receive a message from the server if the creation of this session worked:

```
<status type="ok" description="flow created"/>
```

Otherwise the connection will be closed. From that moment, the client process in the architecture is in the so called *FLOW* (compare with Figure 3.5) state which handles all the data exchange needed for the real usage of the service fulfilment. In this state the client was able to create additional input nodes:

```
<inputnode streamid="speech" fingerprint="en" type="audio"/>
```

These input nodes are used in the processing graph and they imply that this client is generating data coming from this input node. The data emitted by this node will be

labeled with the `streamid`, have the `fingerprint` as language label and contains data of the type `type`. Again, the creation of this input node is acknowledged by the server with a response message either confirming the creation

```
<status type="ok"/>
```

or it might also return an error - for example when an input node with the same *streamid* already exists.

```
<status type="error"
        description="input with this name already present"/>
```

The connected application was also able to continue to create a complete processing graph on its own and define which components should be used. This was achieved by sending `node` messages which trigger the creation of a node, allocating a corresponding service provider to this node (removing it from the pool of available workers).

```
<node sessionid="54321"
      service="smt"
      name="smt-node1"
      inputfingerprint="en"
      inputtype="text"
      outputfingerprint="es"
      outputtype="text"
      streamid="speech"/>
```

In this case all details for the node that is needed to create must be provided. Including the `streamid`, `fingerprints` and also the `service`. With this call, the node also becomes a named node with the name defined in the `name` argument. This name is used to reference this node. This node generating call however itself was not enough to define the processing of data, but we also required the construction of edges between nodes. For this purpose, the connected application sent `edge` messages to the server.

```
<edge from="smt-node1"
      to="tts-node1"/>
```

This message inserted an edge into the processing graph between a node with the name `smt-node1` and `tts-node1`. With these techniques a system combination was possible

which boils down to a single node receiving input from multiple nodes. It was also possible to send the same message to multiple components, for example in the use of a language identification component.
Both types of messages received status answers from the client process on the server to give feedback if the creation of a node or edge was successful.
As an alternative to this method, the connected application was able to directly send a message which output should be generated for a `streamid`.

```
<stream fingerprint="es" type="text" streamid="speech"/>
```

This call triggered the creation of a path in the processing graph, which would transform the input coming from the input node with the streamid `speech` into `text` of the fingerprint `es`. As before, this message was either confirmed to have worked or an error message was sent as a reply from the server. In the description of the architecture we mentioned, that not every session was automatically made available to connected displays. To do this, the application had to announce its session to the public by sending `<announce/>`. This triggered a message from the client process on the server side to the mediator process, that process then informed the connected displays about the session information of the client.
Compared to these control messages, the real data for processing was sent by the application to the client process in multiple packages. While the content of the data messages is free, the attributes follow a certain schema, so that the data is correctly routed in the processing graph.

```
<data streamid="speech"
      sessionid="1428938563"
      stop="2012-03-03 16:47:25.000"
      start="2012-03-03 16:47:23.000"
      creator="smt"
      fingerprint="de-DE"
      type="text">
        <text encoding="url">
          dies\%20ist\%20ein\%20test
        </text>
</data>
```

In the client process only the attributes of the data package were parsed, the content - in the example above the `text` part - was not inspected at all and the provider of the

services needed to agree on a format of this data. We provided however an protocol for the use of *text* and *audio* information as common ground for the most important data types in a speech to speech translation environment.
Data packages were forwarded corresponding to the nodes in the processing graph.

A.2. Worker Connections

When an application connected to the architecture by opening up a port which was used for worker connections, this application was treated as a service provider in the architecture. As soon as the connection was opened a worker process is spawned which handled this connection and the data exchange between the architecture and the application. Therefore it directly sent out a connection message which was similar to the one described in the Client part and also required the same validation mechanism.

```
<status type="connect" description="12345"/>
```

We decided to require the validation in addition, to make sure, that only trusted services can connect and announce they provide a service.
In contrast to the client connection, the worker connection then sent out a description of its service. In this description it provided a `service`, the `inputfingerprint` and `outputfingerprint` as well as the `inputtype` and `outputtype`.

```
<node sessionid="54321"
      service="smt"
      name="translation"
      inputfingerprint="en"
      inputtype="text"
      outputfingerprint="es"
      outputtype="text"/>
```

Again the `sessionid` was used as the verification of the worker. The `name` of the node was optional and not important, but allowed easier debugging. The connected application was able to send multiple messages of this kind, in case the application was able to serve different types of services. Multiple node messages however do not mean, that an application was able to serve multiple requests at the same time. For that purpose the application had to connect more than once to the architecture, spawning separate worker processes.

Once the application made public which services it can provide, a ready message (`<status type="ready"/>`) was sent out, which made this worker available for the use in processing graphs. Unless this message was sent out, the announced service was not available to be used in a processing graph. This allowed the service to do additional initializations. In Figure 3.4 the handling process was after receiving this `ready` message in the ready state, and waiting for a message that a client connected. When this happened, the application received a message about the service description the client was using from this worker. This message was again a node message as described earlier and allowed the worker to do additional handling to prepare for this special use case like swapping in specific user models.

From that moment on the worker needed to be ready to receive data messages. In addition to the data messages, there were also status messages exchanged between the client application and the worker applications. These messages were used for exchanging status messages about the processing. The `flush` was the signal that workers should output their results based on the current information. The `done` message was to notify that no more data was to be expected from this component and `reset` was used to notify the component that it should reset its state and disconnect:

```
<status type="done"/>
<status type="reset"/>
<status type="flush"/>
```

A.3. Display Connections

For the applications connected to the display port the messages were similar to the above ones. But we did not implement an authentication method, for allowing easier consumption of the sessions. The display, once connected, accepted data messages as described above sent to it for displaying the results.
The display process on the server side sent a message to the connected display application when a new session was made available:

```
<session type="add"
         sessionid="12345">
          <sessioninfos>
            [SessionInfos]
          </sessioninfos>
          <streams>
            [Streams]
          </streams
</session>
```

Here the `SessionInfos` were the same as sent out before by the client application. The `Streams` however was a list of possible streams that could be generated and therefore displayed by the display server. At the time of announcement of the streams, these streams however were not necessarily reflected by an existing path in the processing graph. A stream itself was described the following way:

```
<stream type="text"
        fingerprint="en"
        displayname="English-speech"
        control="*"
        streamid="speech"/>
```

Every stream provided the description of its output `fingerprint` and `type` and in addition to that a display name `displayname`, that was used to display this information. In addition to that the stream also had a control field, which was used to mark the original input stream with an *. The display application on the other side was able to register for

streams of a session, by sending out a message of the following format:

```
<register type="[add/delete]" sessionid="12345">
  <streams>
    [Streams]
  </streams>
</register>
```

where the `Streams` was a list of the streams the display application needed the data for. The register message supported two different types, one was `add` to subscribe to messages of this stream and making sure they were sent to the display in the future processing, but on the other hand it also allowed to `delete` and deregister from a stream. This operation caused the client process to remove the connection for this stream to the display in the processing graph. In an additional step this would also cause the client to reduce the processing graph by removing all non display nodes that have no additional node connected to their output. This saved resources by making them available again to the speech translation architecture when they were not needed anymore. When a session ended, the display application was informed with a message, stating which session had ended.

```
<session type="delete" sessionid="12345"></session>
```

The same message was also used, when the client application had an error that resulted in a connection loss or anything else causing a failure in the session.

A.4. Learning Messages

In addition to the obvious previous messages, we also included the storage in the messages, for controlling the learning and distribution of additional learned messages. Since there are multiple different kinds of feedback we decided for an open format of the feedback, described by the following *XSD*:

```
<xsd:element name="cm_post">
  <xsd:complexType>
      <xsd:attribute name="fingerprint"
        type="xsd:string"
        use="required"/>
      <xsd:attribute name="sessionid"
        type="xsd:string"
        use="required"/>
      <xsd:attribute name="userid"
        type="xsd:string"
        use="required"/>
      <xsd:attribute name="type"
        type="xsd:string"
        use="required"/>
      <xsd:anyAttribute/>
  </xsd:complexType>
</xsd:element>
```

The `cm_post` was the message sent either by a client or a display to store a correction in the system. The content of that message then was stored and indexed by `fingerprint`, `sessionid`, `userid` and `type` by the database server, with an additional increasing unique id, which put an order in the stored corrections. The content of the `cm_post` could be in any form, that was agreed on before, so that worker processes were able to extract the needed information from it.
In addition to the posting of corrections and new learning examples, there was also the `cm_get` message, that was used by the workers to get all the collected feedback relevant for this server. The *XSD* for that message was as follows:

```
<xsd:element name="cm_get">
  <xsd:complexType>
      <xsd:attribute name="fingerprint"
        type="xsd:string"
        use="required"/>
      <xsd:attribute name="sessionid"
        type="xsd:string"
        use="required"/>
      <xsd:attribute name="userid"
        type="xsd:string"
        use="required"/>
      <xsd:attribute name="type"
        type="xsd:string"
        use="required"/>
      <xsd:attribute name="revision"
        type="xsd:string"
        use="required"/>
  </xsd:complexType>
</xsd:element>
```

The `cm_get` message used essentially the same fields as the `cm_post`, but in addition it also defined a `revision`. This revision was related to the unique id under which the contents of the `cm_post` messages were stored and it allowed us to not always send all user made corrections, but only those which this worker did not receive up to the moment of sending this message.

B. Description of the REST Api

The REST api is protected by the authentication information in the database. So only a user who is registered to the system is able to make calls to the REST api. The api is accessible over a HTTP client, even over a browser, but the best way to access it is some dedicated HTTP library or for example `curl` commands over the command line because of the ease of using other verbs besides `GET`. The standard call to the REST Api is of the following format:

```
curl -X PUT -u [USER]:[PASS] '[ADDRESS]/[METHOD]' -d'[JSON]'
```

Here the `USER` and `PASS` are the credentials to get access to the API. The `ADDRESS` is the URL where the API is accessible and `METHOD` is the function of the API that is to be called. In the example above, a HTTP `PUT` was executed, which usually requires additional data to be sent to the API. This can be done either by encoding this data in the URL or by passing JSON encoded data. In the following we describe the functions that we provided in our REST api to access our system.

B.1. User Management

Since a valid user was needed for accessing, we start describing the REST api with the user management. The commands in table B.1 were for accessing and changing user information. While users were able to change their own password, or to join groups, some of these commands are only accessible for superusers - those users who are part of the `admin` group. For example only an administrator or the user himself was able to delete a user or add the user to a group without a password. Also the list of all available users was only available to a super user.

Method	Route	Request	Result	Code
GET	/user		[{'username': 'abc@cde.com', 'name': 'demo', 'userid': 5, 'validation': None, 'minutes': '20.0', 'groupid': 5}]	200
GET	/user		as Admin: full list of all users	200
GET	/user	name	[{'username': 'abc@cde.com', 'name': 'demo', 'userid': 5, 'validation': None, 'minutes': '20.0', 'groupid': 5}]	200
GET	/user/[id]		[{'username': 'abc@cde.com', 'name': 'demo', 'userid': 5, 'validation': None, 'minutes': '20.0', 'groupid': 5}]	200
POST	/user	name, pwd, group	{"status":"ok","id":25}	200
POST	/user	name, pwd, group	{"status":"failed", "reason":"username already taken"}	200
PUT	/user/[id]	pwd	{"status":"changed user password"}	200
PUT	/user/[id]	join, group-pwd	{"status":"user added to group"}	200
PUT	/user/[id]	group	{"status":"user added to group"}	200
DELETE	/user/[id]			204

Table B.1.: Commands for user management

To restrict access to groups, passwords were used, which were required for users to join a group and then having access to the data that belonged to that group.

B.2. Groups

For the management of groups, we selected the set of operations shown in table B.2. In the same manner as above, the deletion of groups was only possible by users in the `admin` group and also the list of all groups in the system was only returned for those users, while all other users only saw a list of the groups they are part of. In the section about the users, we already mentioned that a user was able to join a group using a password, this password was set with the corresponding group command and was later used for joining this group. This was for example helpful if multiple users planned to work or use the same resources in the archive.

Method	**Route**	**Request**	**Result**	**Code**
GET	/group		List of groups in system	200
GET	/group	name	description of group with name	200
GET	/group/[id]			
POST	/group	name [pwd]	{”status”:”ok”,”id”:14}	200
PUT	/group/[id]	pwd		204
DELETE	/group/[id]			204

Table B.2.: Commands for managing groups

B.3. Resources

The most important methods provided by the REST api were those methods that were able to create or manipulate resources. A resource was nothing else than a session and all its associated data messages. It was possible for a user to request his own resources, but also to delete or modify them. Again, only users in the group `admin` had full access to every resource. Furthermore the REST api allowed also the creation of new resources. This triggered a creation of a session in the mediator in combination with the necessary calls for workers creation of a client and streaming of the data. We successfully used this part of the REST api to provide an HTML5 interface to presenters, allowing presentations using the speech translation architecture only with the usage of these calls.

In contrast to the binary library we provided, this api also allowed the transcription of video and audio resources which were accessible in the world wide web. For this task an internal queuing mechanism was implemented which allowed the user to queue up multiple external resources for transcription which were then transcribed, as soon as the resources that can do that were available.

The triggering of such an action provided the user with a `RID` - a request id, which the user was able to use to check the status of the transcription and also abort the task.

For a `RID` you can get the status of a transcription by using the `"state":"current"` and you can stop a transcription with `"state":"stop"`.
Possible states for a `RID` were:

- `{"status":"currently downloading"}`, the data of the resource is prepared for the later processing
- `{"status":"waiting for processing"}`, this means the data is on the server waiting to be processed
- `{"status":"currently processing"}`, the data is being processed, meaning the data is being transcribed
- `{"status":"completed", "sessionid":id}`, the processing is finished, the returned id is the session id which references the results of the transcription process
- `{"status":"error", "reason":reason}`, if anything happened during processing, an error message is returned
- `{"status":"not in processing"}`, if nothing is done with this request id

When posting a new resource `il` and `ol` defined the input and output languages in the same fingerprint format described in the previous chapters. If these parameters were not provided, English was assumed. `description` and `title` allowed to give the meaningful names to the resources.

As already mentioned, we also developed mechanisms for the REST api which allowed users to use the REST api for online transcription. This was achieved, by using the binary data transfer in a POST message and chunk processing on the receiving side of our api. An example curl command for this task was:

```
curl -k -X POST -u USERNAME:PASSWORD
     -H "Content-type: application/json"
     -H 'Accept:application/json' 'APIADDRESS/resource'
     -d'{"il":"en-EU", "title":"Title of the transcript",
         "description":"Description of the transcript"}'
     --data-binary @FILENAME
```

Method	Route	Request	Result	Code
GET	/resource	limit, offset, search	[{”date”:”07/12/12-04:18:20.465”, ”userid”:X, ”readrights”:X, ”access-rights”:X, ”source”:”” ,”language”:”en” ,”password”:”NULL” ,”presenter”:”X” ,”description”:”X” ,”id”:ID},...]	200
GET	/resource/[id]		[{”date”:”05/12/12-08:29:06.976”, ”messages”:X, ”sessionid”:ID, ”type”:”text”, ”fingerprint”:”YY”, ”creator”:”X”, ”streamid”:”X”, ”readrights”:X, ”access-rights”:X, ”source”:””, ”language”:”en”, ”password”:”X”, ”presenter”:”X”, ”description”:”X”, ”id”:ID},...]	200
GET	/resource/[RID]	state:current	{”status”:state}	200
GET	/resource/[RID]	state:stop	{”status”:”ok”}	200
GET	/resource/[id]	format=url	{”url”:”http://...”}	200
GET	/resource/[id]	streamid, fingerprint, format, [start]	json or srt depending on format	200

Method	**Route**	**Request**	**Result**	**Code**
DELETE	/resource/[id]			204
PUT	/resource/[id]	start, stop, creator, fingerprint, text, time-offset, streamid		
POST	/resource	url, il, description, title	{”requestid”:RID}	200
POST	/resource	il, ol, translate	{”fingerprint”:”es”, ”start”:”2012-03-03 16:47:23.000”, ”stop”:”2012-03-03 16:47:25.000”, ”text”:”Traducir este”}	200

Table B.3.: Commands for managing resources