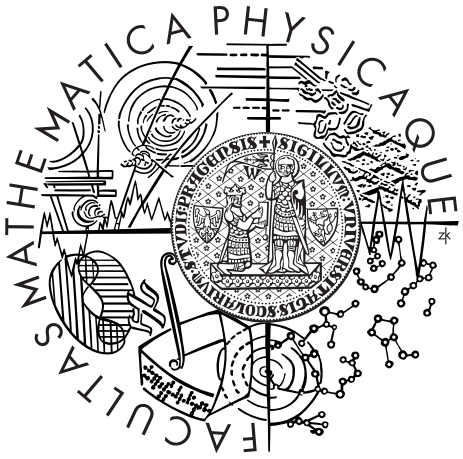
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Matouš Macháček

Measures of Machine Translation Quality

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: RNDr. Ondřej Bojar, Ph.D.
Study programme: Computer Science
Specialization: Mathematical Linguistics

Prague 2014

I would like to thank my supervisor, RNDr. Ondřej Bojar, Ph.D., for his advice and guidance. He taught me a lot over the years of my study and I am very grateful to him for that.

Many thanks go to annotators of machine translation quality. Without them, there would be no manual evaluation experiment in my thesis.

I would like to also thank Markéta Šteflová for proofreading some parts of my thesis.

Finally, I would like to give special thanks to my girlfriend Bára for her love, support, encouragement and patience throughout the countless hours I have spent on this thesis. I would like to dedicate this thesis to her.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, 31th of July 2014

Název práce: Měření kvality strojového překladu

Autor: Matouš Macháček

Ústav: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: RNDr. Ondřej Bojar, Ph.D.

Abstrakt: V této práci zkoumáme manuální i automatické metody pro vyhodnocování kvality strojového překladu. Navrhujeme manuální metodu evaluace, ve které anotátoři hodnotí místo celých vět pouze krátké úseky strojového překladu, což zjednodušuje a zefektivňuje anotaci. Provedli jsme anotační experiment a vyhodnotili jsme systémy strojového překladu podle této metody. Získané výsledky jsou velmi podobné těm z oficiálního vyhodnocení systémů v rámci sotěže WMT14. Získanou databázi anotací dále používáme k evaluaci nových, neviděných systémů a k ladění parametrů statistického strojového překladače. Evaluace nových systémů ale dává nepřesné výsledky a v práci proto analyzujeme důvody tohoto neúspěchu. V rámci zkoumání automatických metod evaluace jsme dvakrát po sobě organizovali soutěž strojových metrik v rámci workshopu WMT. V této práci uvádíme výsledky z poslední soutže, diskutujeme různé metody metaevaluace a analyzujeme některé zúčastněné metriky.

Klíčová slova: strojový překlad, vyhodnocování kvality, automatické metriky, anotace

Title: Measures of Machine Translation Quality

Author: Matouš Macháček

Department: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Ondřej Bojar, Ph.D.

Abstract: We explore both manual and automatic methods of machine translation evaluation. We propose a manual evaluation method in which annotators rank only translations of short segments instead of whole sentences. This results in easier and more efficient annotation. We have conducted an annotation experiment and evaluated a set of MT systems using this method. The obtained results are very close to the official WMT14 evaluation results. We also use the collected database of annotations to automatically evaluate new, unseen systems and to tune parameters of a statistical machine translation system. The evaluation of unseen systems, however, does not work and we analyze the reasons. To explore the automatic methods, we organized Metrics Shared Task held during the Workshop of Statistical Machine Translation in years 2013 and 2014. We report the results of the last shared task, discuss various metaevaluation methods and analyze some of the participating metrics.

Keywords: machine translation, evaluation, automatic metrics, annotation

Contents

Chapter 1

Introduction

In the globalized world we live in, there is a need for translating from one human language to another. The translation itself is often not easy even for people. They have to be trained for that and well paid. There is therefore a very high demand for cheap and high-quality machine translation. A lot of researchers and companies try to satisfy this demand and constantly improve their translation systems. One of the most important thing when improving your system is that you know how to measure the improvement.

There are a lot of situations in which you need to evaluate machine translation. If you are a customer who would like to buy a machine translation system you want to buy the system which will best suit you. There are many criteria of MT systems (speed, price, memory consumption, scalability, etc.) but the most important criterion is usually the machine translation quality. Therefore, you would like to evaluate this quality on a sample of documents you are going to translate.

If you are a developer of a machine translation system you need to evaluate your system during various phases of the development process. Let us go through the phases which need an evaluation in reversed order. From time to time, you have a ready-to-use system and you would like to have a comparison to systems of your competitors or another researchers. You can evaluate your system yourself in a way that the obtained score is comparable to other scores, or you can participate in a shared evaluation campaign, for instance NIST OpenMT[1] or at Workshop on Statistical Machine Translation[2] (WMT). During the day-to-day development process of your system, you try various configuration, new components and features. You have to evaluate your system with every change to know how much the change improved (or sometimes worsened) the translation quality. Recently, new automatic methods which tune parameters of your system to directly optimize an evaluation measure emerged.

Unlike other applications of natural language processing, for instance speech recognition, the evaluation in machine translation is not easy at all. The main reason for that is that when translating an average sentence, there is no single correct solution, in fact there are hundreds of thousands correct translations (Bojar et al., 2013).

There are two fundamental approaches in the machine translation evaluation:

[1] `http://www.nist.gov/itl/iad/mig/openmt12.cfm`

[2] `http://www.statmt.org/wmt14/translation-task.html`

a manual evaluation and an automatic evaluation (Bojar, 2012). In the manual evaluation, human judges assess the quality of each sentence manually. The most widely used methodology in the past was to assign values from two five point scales representing fluency and adequacy. However, it was shown that a) people had difficulties with separating these two aspects of translation and b) there was a very small inter-annotator agreement, since each annotator had different expectation of a good translation. In later evaluation campaigns started by Callison-Burch et al. (2007), annotators ranked translations relatively to each other.

In the automatic evaluation, an output of a machine translation system is compared to a reference translation. However, unlike other machine learning problems, you cannot simply compute the proportion of translated sentences which match the reference translation. If a translated sentence is not identical to its reference counterpart it does not have to be a bad translation because there is the very large number of correct translations. Automatic metrics[3] are used to measure the similarity between the candidate and reference translations. The more similar they are, the better the translation is considered.

As you can see, the evaluation of machine translation is very important and at the same time, very difficult. We explore both manual and automatic evaluation in this thesis.

1.1 Motivation and Goals

Manual evaluation is of course considered as the only source of truth which metrics try to approximate. However, it suffers from many disadvantages. Since it includes manual labour, it is very costly and slow. Moreover, manual evaluation is not reproducible; human judges have different criteria for comparing candidates and even an individual judge is not consistent with himself in time. Human evaluation is therefore most often used in shared evaluation campaigns and sometimes used when you want to evaluate a new component of your system. It is definitely not feasible to directly use human evaluation in an automatic method for tuning your model's parameters because these methods require to evaluate millions of sentences.

Automatic metrics, on the other hand, are fast, reproducible and cost almost nothing. However, they are only proxies to human evaluation and their correlation to human judgments varies a lot. The first goal of this thesis is therefore to conduct a large-scale comparison of available automatic metrics in the terms of correlation with human judgments. For this purpose, we have organized Metrics Shared Task within Workshop on Statistical Machine Translation in two consecutive years 2013 (Macháček and Bojar, 2013) and 2014 (Macháček and Bojar, 2014). We also compare the metrics in other more subjective criteria.

It would be very useful if we have an evaluation method which would take advantages from both manual and automatic evaluations. Recently, there actually emerged such methods on the boundary of manual and automatic evaluation. These methods usually require a large manual annotation effort at the beginning

[3]Automatic metrics for machine translation evaluation are not actually metrics according to the mathematical definition. For example, the triangle inequality usually does not hold and some metrics are not even symmetric. However, we are going to use this traditional term in this thesis.

to create a database and then they use the collected database in an automatic way during evaluation. The second goal of this thesis is to propose a method which could be used to manually evaluate a set of systems and the database collected during this manual evaluation could be later reused to automatically evaluate new, unseen systems and to tune a system. This goal includes, besides proposing the method, also developing an annotation application, conducting a real evaluation experiment and experiment with reusing the collected database.

1.2 Outline

The thesis is organized as follows. In Chapter 2 we propose the manual evaluation method and report the annotation experiment we have conducted. We explore the possibility of reusing the collected database to evaluate new systems and to tune a system in Chapter 3. In this chapter, we also analyze the results in more details and try to explain them. The automatic metrics are described, compared and evaluated in terms of correlation with human judgments in Chapter 4. Related work is summarized in Chapter 5. We conclude our work in Chapter 6.

There are three appendices, all of them relate to the contents of the attached DVD-ROM. In Appendix A, you can find the documentation of the metrics task package which is used to compute the metrics task results. In Appendix B, you can find the user documentation of the annotation application called *SegRanks* which is used in the annotation experiment. You can find the development documentation of this application in Appendix C.

Chapter 2

SegRanks: Manual Evaluation Based on Short Segments

In the following two chapters, we propose a novel manual evaluation method in which we create a database of human annotations. This database could be reused later to automatically evaluate similar but unseen translations or even to tune systems. This method therefore lies on the boundary of automatic and manual evaluation methods.

The proposed method consists of two parts, which we describe and experiment with in the following two chapters. In this chapter, we describe the way of collecting the database of human judgements and report the human annotation experiment which we conducted using the proposed method. In the following chapter, we present some methods which exploit the collected database: besides the evaluation of annotated systems, we experiment with extrapolating the database to evaluate unseen translations and with tuning a machine translation system using the database.

In the WMT official human evaluation, humans judge whole sentences. They get five candidate translations of a given source sentence and their task is to rank these candidates relatively to one another (ties are allowed). One of the disadvantages of this method is that the sentences are quite long and therefore quite hard to remember for the judge to compare them. Also, when comparing longer sentences, there are many more aspects in which one sentence can be better or worse than another, and therefore it is more difficult for judges to choose the better of the candidates.

To avoid these disadvantages, we propose the following method. Instead of judging whole sentences, we extract short segments[1] from candidates and give them to judges to rank them. In order to extract meaningful segments with the same meaning from all candidates, we do the following procedure: First, we parse the source sentence and then recursively descend the parsed tree and find nodes which cover source segments of a given maximum length. This is described exactly in Algorithm 1. Finally, we project these extracted source segments to their counterpart segments in all candidate sentences using an automatic alignment. You can find the whole process illustrated in Figure 2.1. This extraction method is inspired by Zaidan and Callison-Burch (2009) and by the WMT07 manual

[1]The term 'segment' is sometimes used in the literature to refer a sentence. In this thesis, we will use the term 'segment' for a phrase of few words.

Algorithm 1 Short Segment Extraction From Source Side Parse Tree

```
1: function ExtractSegments(treeNode, minLength, maxLength)
2:     extractedSegments ← list()
3:     leaves ← treeNode.leaves()
4:     if length(leaves) ≤ maxLength then
5:         if lenth(leaves) ≥ minLength then
6:             extractedSegments.append(leaves)
7:     else
8:         for node in treeNode.children() do
9:             segments ← ExtractSegments(child, minLength, maxLength))
10:            extractedSegments.extend(segments)
       return extractedSegments
```

evaluation (Callison-Burch et al., 2007).

In the segment evaluation in (Callison-Burch et al., 2007), these extracted segments are only highlighted and shown to judges together with the rest of the sentence. Judges are asked to rank the highlighted segments in the context of the whole candidate sentences.

We use a different approach here which is more similar to that used by Zaidan and Callison-Burch (2009). We show the extracted segments without any context and ask judges to rank them. The only additional information provided to annotators is the whole source sentence with the source segment highlighted. Judges are told that they can imagine the rest of the sentence in which the ranked segment fits best. They are instructed to penalize only those segments for which they cannot imagine any appropriate rest of the sentence.

While we are aware that this approach has some disadvantages (which we summarize below) there is one significant advantage: it is much more likely that two systems produce the same translation of a short segment than that they would produce the same translation of a whole sentence. Because we do not show the sentence context to annotators, we can merge identical segment candidates into one, so the annotators have fewer candidate segments to rank. This also allows us to reuse already collected human judgements later to evaluate a new system which was not in the set of annotated systems or to tune parameters of a system.

The following list summarizes disadvantages of this method, which we are aware of. However, we believe that the advantages still outweight the problems and that our method is worth exploration.

- A system could translate shorter segments quite well but it can fail to combine them properly when creating the whole sentence translation. For instance, a system may fail to preserve the subject – verb agreement, which is very important in the Czech language. In their paper, Zaidan and Callison-Burch (2009) suggest to go up the parse tree and extract also the longer segments which consist of already extracted shorter segments. However, if we use this approach the amount of annotation work would multiply several times. Furthermore, the longer segments are more difficult to rank and the chance that systems' candidates will be identical (so that we can merge them for annotation) is lower.

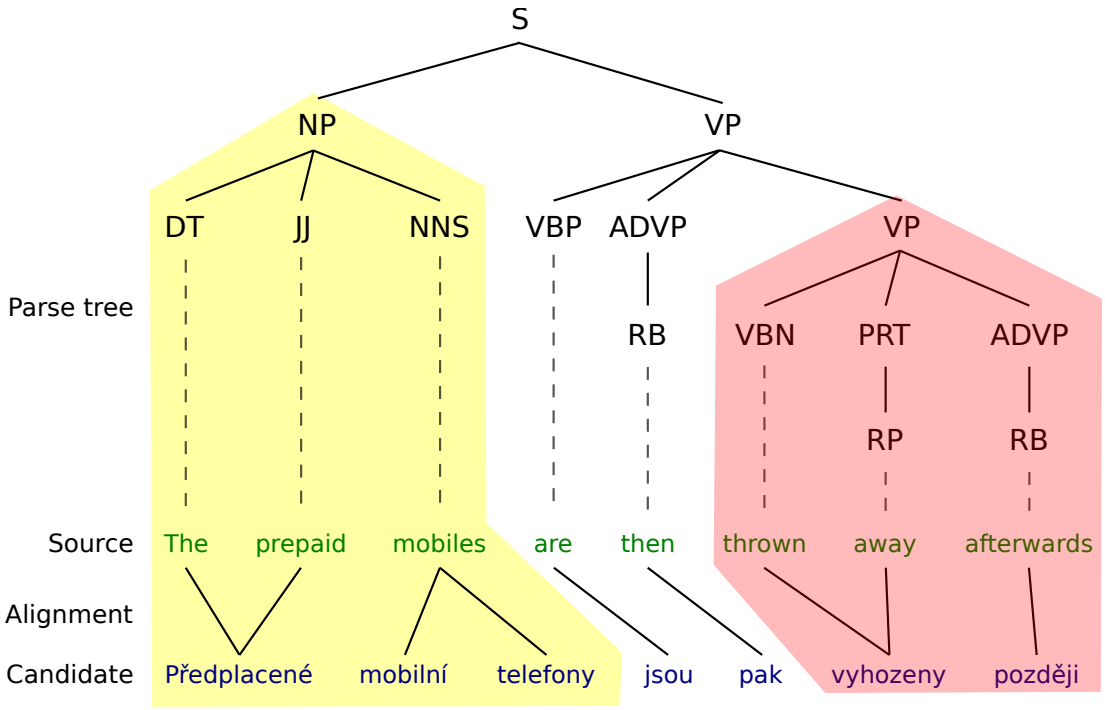


Figure 2.1: An illustration of candidate segments extraction process. For a given MT system, two segments were extracted from this sentence. The `maxLength` constant was set to the value 4 here, to illustrate that not all of the words are always covered by the extracted segments.

- Annotators do not see the whole context of annotated short segments. They are instructed to imagine any suitable context of the segment. However, they can fail to imagine a suitable context even if there exists one and wrongly penalize the segment. To partially remedy this disadvantage we give all extracted short segments to annotators to judge at once, so they can at least imagine the context.

- Extracted short segments do not cover the whole sentence. For example in Figure 2.1, the words 'jsou' and 'pak' are not part of any extracted segment. We would avoid this problem if we set the variable `minLength` to zero. This, however, would generate a high number of short segments to annotate.

- Some segment candidates are much more important to convey the meaning of a sentence than others, and therefore should not have equal weights when being interpreted. When an annotator ranks system A better than system B in two of three ranked segments, and system B better than system A in the third segment, it does not always mean that he would have ranked system A better than system B when ranking whole sentences. The third segment could be much more important for the quality of translation than the first two. We are afraid that it is not possible to easily avoid this problem. However, we also believe that this problem is not so severe and that possible differences in the importance of individual segments cancel out.

2.1 Data and Segment Preparation

We have conducted an annotation experiment using the proposed method. In this section, we describe what data we have used and how we prepared it for the annotation experiment.

We used English to Czech part of the WMT14 (Bojar et al., 2014) test set. We chose this data set to be able to compare experiments' results with the official WMT14 human evaluation.

The testset consists of 3003 sentences (68866 tokens). It contains both source sentences and reference translations. Roughly a half of the sentences were originally in Czech and were translated by human translators into English. The second half of the sentences was translated in the opposite direction. Besides the source and reference translations, we also used candidate translations of 10 systems which participated in the WMT14 translation task. All systems are listed in Table 2.1.

ID	Type	Team
CU-DEPFIX	hybrid	Charles University, Prague (Tamchyna et al., 2014)
CU-BOJAR	hybrid	
CU-FUNKY	hybrid	
CU-TECTO	hybrid	
UEDIN-PHRASE	statistical	University of Edinburgh (Durrani et al., 2014)
UEDIN-UNCNSTR	statistical	
COMMERCIAL-1	rule-based	Commercial machine translation systems
COMMERCIAL-2	rule-based	
ONLINE-A	statistical	Online statistical machine translation systems
ONLINE-B	statistical	

Table 2.1: The machine translation systems participating in the WMT14 translation task in English-Czech direction which were used in the annotation experiment

Source sentences and all candidate translations were tokenized using the script `tokenizer.perl`. Unicode punctuation characters were normalized using the script `replace-unicode-punctuation.perl`. (Both scripts are included in the Moses toolkit).

The source sentences were parsed using the Stanford parser. We used lexicalized `englishFactored` model (Klein and Manning, 2003b) which is distributed with the parser. We also tried unlexicalized `englishPCFG` (Klein and Manning, 2003a) and compared the segments extracted using the both parsers on a small random sample of sentences. The `englishFactored` model yielded subjectively better segments.

We constructed an alignment between the source sentences and the candidate translations using Giza++ (Och and Ney, 2003). Since the alignment algorithm is unsupervised and the amount of all candidate translations is relatively small (10×3003), we introduced more data by concatenating all candidate translations with 646,605 sentences taken from the Europarl parallel corpus (Koehn, 2005) and with 197,053 sentences taken from the CzEng parallel corpus (Bojar et al., 2012). The concatenated parallel corpus was lowercased before the alignment computation.

We extracted short segments from the parsed source trees using Algorithm 1. The constant `minLength` was set to the value 3 to filter out very short segments which are hard to judge without context. This also helped reduce the number of extracted segments to be annotated. The constant `maxLength` was set to the value 6 so the extracted segments were not too long to judge and at the same time it was more likely that two candidate translations of a segment were equal and therefore there would be fewer items to rank (our aim was to make annotations as easy and fast as possible). We have experimented with various settings of these two constants and the final settings seemed to generate a reasonable number of meaningful segments.

From the 3003 source sentences, we have extracted 8485 segments of length 3 to 6 tokens. That is approximately 2.83 segments per sentence on average. By projecting the source segments to the candidate sentences using the computed alignments, we got $10 \times 8485 = 84850$ candidate segments. However, after the merging of equal segments, only 50011 candidate segments were left. This represents 58.9 % of the original candidate segments, or in other words, after the merging we got 5.89 (instead of original 10) candidate segments to be ranked for each source segment on average. These prepared candidate segments were inserted into the database to be ranked by annotators.

2.2 Ranking of Segments

We have developed a new annotation application called *SegRanks* for this annotation experiment.[2] Please see Appendix B for the user documentation and Appendix C for the development documentation.

You can find an example screen shot of this application in Figure 2.2. Annotation instructions were displayed on each annotation screen. This is the English translation of these instructions:

> A number of segments are extracted from the annotated sentence. You are shown a few candidate translations for each of these segments. Your task is to distinguish acceptable candidate translations (the meaning of the segment can be guessed despite a few or more errors) from unacceptable ones (the meaning is definitely not possible to guess from the candidate segment). Also please rank the acceptable candidate translations relatively from the best ones to the worst ones. Please place better candidate translations higher and the worse ones lower. You can place candidates of the same quality on the same rank. We ask that you place unacceptable candidates to the position "Garbage"
>
> Please note that source segments and their candidate translations are chosen automatically and do not have to be perfect. Consider them only as approximate clues. If a candidate segment contains

[2]It would be probably possible to customize and use an existing annotation application, for example Appraise (Federmann, 2012). However, since the ranking of short segments is quite specific it would require a lot of customization. We therefore decided to develop our own light-weight web application which would suit our needs perfectly and allow us to optimize the efficiency of annotation.

Figure 2.2: A screenshot of the annotation application. Annotators rank the candidate segments by dragging and dropping them into the ranks. Annotators see all annotated segments of a sentence on a single screen.

> an extra word, which does not correspond to the source segment but otherwise could be in the translated sentence, you do not have to rank such candidate any worser. If something is missing in the candidate translation you should consider it an error.

Our goal was to make the annotation as efficient and user friendly as possible. Annotators rank all the source segments of a sentence on a single screen (so that they have to read the whole source sentence and reference translation only once). For each annotated segment they see the source sentence repeated, with the annotated segment highlighted. Annotators rank the segment candidates by dragging and dropping them to appropriate rank positions. When all the candidates of all the source segments of the sentence are ranked, annotators are allowed to submit their annotations to the server. The web interface has a responsive design, so it is displayed correctly on smaller screens, and the drag-and-drop works also on touch screens. Annotators were therefore able to rank segments on a tablet.

The very annotation experiment was conducted during May/June 2014 and it lasted exactly one month. During this time, 17 annotators ranked segments of 2765 sentences, which is more than 92 % of the prepared English-Czech test set.

2.2.1 Annotator Agreements

To measure the reliability and robustness of the proposed annotation method, we have computed intra- and inter-annotator agreements. A reasonable degree of these agreements supports the suitability of this method for machine translation evaluation.

We measured the agreements using Cohen's kappa coefficient (κ) (Cohen, 1960). Let $P(A)$ be the proportion of times the annotators agree and $P(E)$ be the proportion of time that they would agree by chance. Then the Cohen's κ is computed using the following formula:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

Simply put, κ is the proportion of times the annotators agrees of all the times they would not agree by chance. Note that κ is a normalized version of $P(A)$; it considers how difficult it is to agree without knowing anything about the task. Values of κ can be therefore compared in principle across various annotation experiments. The maximum value is 1 which would mean that annotators always agree. A value of zero would mean that annotators agree as often as they would by chance.

In our case, $P(A)$ and $P(E)$ are computed in the context of pairwise comparisons. Approximately 5 % of the annotated sentences were annotated twice by two different annotators (for the inter-annotator agreement). Another 5 % of the sentences were annotated twice by the same annotator (for the intra-annotator agreement). From all the segments of these double annotated sentences, we extracted pairwise comparisons of candidate segments. Then we computed $P(A)$ as the proportion of pairwise comparisons in which annotations match.

We computed the expected agreement by chance as

$$P(E) = P(A > B)^2 + P(A = B)^2 + P(A < B)^2$$

	our method	Bojar et al. (2014)
intra-annotator κ	0.593	0.448
inter-annotator κ	0.397	0.360

Table 2.2: κ scores measuring intra-annotator and inter-annotator agreements. We also report corresponding κ scores from official WMT translation task for comparison. Please see Table 2.3 for the annotator agreements computed for individual annotators

where $P(A > B)$, $P(A = B)$ and $P(A < B)$ were computed empirically as the relative frequencies of cases where the two segments A, B are ranked $>$, $=$, or $<$ respectively, across all annotations of the pair A, B, regardless the sentence or annotator. The value of $P(E)$ in our experiment is 0.394, which means that the probability of the outcomes $A > B$, $A = B$ and $A < B$ is not uniform.

The final values of inter-annotator and intra-annotator κ can be found in Table 2.2 You can also compare them to the corresponding κ values from WMT14 translation task (Bojar et al., 2014), which were computed similarly on the same testset. The exact interpretation of the Kappa coefficient is difficult, but according to Landis and Koch (1977), 0 – 0.2 is slight, 0.2 – 0.4 is fair, 0.4 – 0.6 is moderate, 0.6 – 0.8 is substantial and 0.8 – 1.0 is almost perfect. You can see that we get both κ scores better than those of WMT14. However, they are still quite low and we expected them to be higher, since the annotation task was designed to be much simpler than the ranking full segments in the official WMT human evaluation.

We also computed the agreements κ for individual annotators and report them in Table 2.3. As you can see, these scores are varying a lot. Based on these scores, we could filter out unreliable annotators. However, we do not do that in order to maintain comparability to the WMT14 experiment.

ID	n_{sent}	n_{seg}	t	$\frac{t}{n_{seg}}$	κ_{intra}	κ_{inter}
1	78	232	4:53:43	75	0.838	0.495
4	87	248	5:00:46	74	0.283	0.145
6	378	1160	16:53:13	52	0.657	0.453
7	243	663	11:16:26	62	0.763	0.417
8	6	20	0:21:06	63		
9	12	50	0:45:58	55	0.686	
10	98	274	5:04:45	66	0.673	0.320
12	3	7	0:15:03	129	0.281	
13	424	1260	1 day, 2:26:14	75	0.459	0.308
15	106	282	5:17:58	67	0.611	0.401
16	224	627	5:00:11	28	0.655	0.614
17	26	74	2:40:44	130	0.734	0.269
18	83	234	7:02:10	111	0.676	0.385
19	15	50	1:31:52	110		
21	483	1443	1 day, 13:44:37	94	0.663	0.383
22	117	338	6:00:36	64	0.714	0.431
23	477	1371	22:07:35	58	0.376	0.363
Total	2773	8333	6 days, 14:22:57	68	0.593	0.397

Table 2.3: The list of annotators and their statistics. The annotators are anonymized using their ID. The table contains the number of annotated sentences n_{sent}, the number of annotated source segments n_{seg} (this is not the number of ranked segment candidates), the time spent annotating t, the average time in seconds spent annotating one source segment $\frac{t}{n_{seg}}$, the normalized intra-annotator agreement κ_{intra} and the normalized inter-annotator agreement κ_{inter}.

Chapter 3

Using Short Segments in MT Development

In this chapter, we describe several experiments with the collected database of annotations.

3.1 Overall Ranking of Annotated Systems

In the first experiment, we would like to show that the proposed method can be used to produce overall ranking of the annotated systems which will be very similar to the official human evaluation in WMT.

The obtained database contains a list of annotations for each extracted source segment from each source sentence. The list can be empty (not all of the sentences were annotated), it can contain more than one annotation (some segments were annotated twice by an annotator, some segments were annotated by multiple annotators), but most of the time it contains only one annotation.

An annotation is a mapping from the set of candidate segments to the set of ranks $1 \ldots N+1$, where N is the count of unique candidate segments. (To discriminate the relative quality of all segments, annotators had available N ranks which could be all occupied when there were no ties. The rank $N+1$ represents the "garbage" category from the annotation application. However in all of our experiments reported in this thesis we consider this category simply as one more rank, the worst one). A lower rank of a segment means that the candidate segment was ranked better. This is an example **segment annotation** of a source segment "the huge volume":

```
{
  'velkému objemu' : 1,
  'obrovské hlasitosti' : 5,
  'obrovský objem' : 2,
  'obrovské množství' : 2,
}
```

We expanded these segment annotations with the information about the systems which produced the segments. The ranks in the annotations are now indexed by the system names. If more systems translated a source segment as the same

candidate segment, the candidate segment's rank is copied to all the systems. At this point we take advantage of ranking short segments which are often translated identically. The following is the **system annotation** after expansion of the above **segment annotation**:

```
{
  'uedin-unconstrained' : 2,
  'commercial1' : 5,
  'commercial2' : 2,
  'CU-TectoMT' : 1,
  'onlineB' : 2,
  'onlineA': 2,
  'cu-funky' : 2,
  'cu-bojar' : 2,
  'uedin-wmt14': 2,
  'cu-depfix': 2
}
```

These rankings are now very similar to those obtained in the official WMT human evaluation. These annotations are mostly interpreted as pairwise systems' comparisons (for each combination of size 2 of all systems we have a pairwise comparison), where the absolute values of the ranks and their absolute differences between them are not considered. From the above **system annotation**, the following **pairwise comparisons** are extracted (only a few extracted pairwise comparisons are listed here for the sake of brevity, generally $N \times (N-1)/2$ pairwise comparisons are extracted, where N is number of all systems):

```
[
  'uedin-unconstrained' < 'commercial1',
  'uedin-unconstrained' = 'commercial2',
  'uedin-unconstrained' > 'CU-TectoMT',
  ...
  'commercial1' > 'commercial2',
  'commercial1' > 'CU-TectoMT',
  ...
]
```

The interpretation of these pairwise comparisons has changed several times during the WMT workshops. Here, we use **Ratio of wins (ignoring ties)** method, which was introduced by Bojar et al. (2011) and used in WMT12 workshop (Callison-Burch et al., 2012). This method is based on a method used in several WMT workshops before WMT12 and it is quite easy to compute and interpret results.

For a given set C of segment-level extracted pairwise comparisons (s_1, s_2, c), where

$$c = \begin{cases} win & \text{if } rank(s_1) < rank(s_2) \\ loss & \text{if } rank(s_1) > rank(s_2) \\ tie & \text{if } rank(s_1) = rank(s_2) \end{cases}$$

System	Score
cu-depfix	0.5777
onlineB	0.5642
uedin-unconstrained	0.5626
cu-bojar	0.5606
cu-funky	0.5566
uedin-wmt14	0.5498
onlineA	0.5007
CU-TectoMT	0.4485
commercial1	0.3992
commercial2	0.3492

(a) Short segments judgements

System	Score
cu-depfix	0.6101
cu-bojar	0.6011
uedin-unconstrained	0.5967
cu-funky	0.5823
onlineB	0.5439
uedin-wmt14	0.5285
onlineA	0.5039
CU-TectoMT	0.4473
commercial1	0.3617
commercial2	0.2780

(b) Official WMT14 judgements

Table 3.1: Overall rankings of systems according to **Ratio of wins (ignoring ties)** score. You can see the results computed on short segments judgements and the results computed on the official WMT14 human judgements side by side to compare the differences.

we define for each system s the total number of wins, losses and ties:

$$\begin{aligned} win(s) &:= |\{(s, \bar{s}, c) \in C; c = win\}| + |\{(\bar{s}, s, c) \in C; c = loss\}| \\ loss(s) &:= |\{(s, \bar{s}, c) \in C; c = loss\}| + |\{(\bar{s}, s, c) \in C; c = win\}| \\ ties(s) &:= |\{(s, \bar{s}, c) \in C; c = tie\}| + |\{(\bar{s}, s, c) \in C; c = tie\}| \end{aligned}$$

Then the **Ratio of wins (ignoring ties)** for a given system s is computed using the following formula:

$$E_{win}(s) = \frac{win(s)}{win(s) + loss(s)}$$

3.1.1 Results

The overall ranking of systems which participated in WMT14 Translation Task in English-Czech direction, according to the **Ratio of wins (ignoring ties)** computed on the short segments judgements, is reported in Table 3.1a.

To compare our method with the classic method of judging whole sentences, we have also computed the **Ratio of wins (ignoring ties)** on the judgements collected during WMT14 manual evaluation. You can see these results in Table 3.1b. The Pearson correlation coefficient between the short segments score and official human scores is 0.978 (please see Section 4.3 for details of how we computed this correlation).

You can notice that the range of scores computed on the short segments judgements is much more narrow (0.35 — 0.58) than the range of scores computed on the sentence judgements (0.28 — 0.61). This can be explained by the following: if system A beats system B in a sentence-level judgement of a particular sentence it does not necessarily mean that in segment-level judging system A will be better

than system B on all segments of the sentence. System A will be probably better on a majority of the segments (but even that does not have to be always true). When computing the ratio of wins on the sentence-level judgements, system A gets one win and system B gets one loss. However, when computing the ratio of wins on the segment-level system A gets for instance two wins and one loss, system B one win and two losses. It should be clear now that computing expected wins on the sentence-level judgements is coarser while our method is more fine-grained.

The overall rankings of the systems obtained by both of the methods are very similar. However, there are two changes when comparing to the sentence-level judgments results: system online-B is better and system cu-bojar is worse according to the segments-level judgments results. We try to explain this in Section 3.1.2 below.

3.1.2 Analysis

To see the difference between the segment-level judgements and sentence-level judgements, we have computed Kendall tau rank correlation coefficient, also known as Kendall's τ, between segment-level pairwise comparisons and sentence-level pairwise comparisons. This coefficient is used to measures how often a set of pairwise rankings agrees with another set of pairwise rankings. The basic formula for Kendall's τ is:

$$\tau = \frac{|Concordant| - |Discordant|}{|Concordant| + |Discordant|}$$

where $Concordant$ is the set of pairwise combination where both sets of pairwise rankings agrees with each other and $Discordant$ is the set of pairwise combination where the sets do not agree in pairwise ranking. We will discuss the Kendall's τ in more detail in chapter 4. Here, we computed $|Concordant|$ as the number of pairwise segment-level judgments which agrees with the corresponding sentence-level judgment. Similarly, $|Discordant|$ is the number of those which do not agree with the corresponding sentence-level judgment. In this section, we do not consider any tied pairwise comparisons.

The computed correlations are given in Table 3.2. The order of the systems in Table 3.2a is quite similar to the order of systems in the overall rankings (Table 3.1); better systems have higher correlation than worse systems. This is somehow expected: Better systems, which were more often ranked better in the sentence-level judgements are also more likely to be ranked better in the segment-level judgments. You can also read the table in the following way: If a sentence of system cu-funky was ranked better on sentence-level, it is very likely (71%) that the segments of the sentence were ranked better also. On the other hand, if system commercial2 was ranked better on sentence-level only a little more than half of the segments were ranked better also. Table 3.2b is similar but in reversed order.

The influence of a system's quality should be canceled out in Table 3.2c. You can see that systems cu-bojar and onlineB have the Kendall's τ lower (although not the lowest). This, together with the fact that both systems lie in a cluster of systems of very similar quality, is consistent with their change in the overall

System	τ
cu-funky	0.428
cu-depfix	0.405
onlineB	0.398
cu-bojar	0.394
uedin-uncon.	0.390
uedin-wmt14	0.363
onlineA	0.312
CU-TectoMT	0.270
commercial1	0.166
commercial2	0.123

(a) Better systems

System	τ
commercial2	0.492
commercial1	0.441
CU-TectoMT	0.385
onlineA	0.328
uedin-uncon.	0.260
uedin-wmt14	0.257
cu-funky	0.230
cu-bojar	0.227
cu-depfix	0.225
onlineB	0.225

(b) Worse systems

System	τ
commercial2	0.394
cu-funky	0.348
commercial1	0.345
uedin-uncon.	0.341
cu-depfix	0.335
CU-TectoMT	0.333
cu-bojar	0.328
onlineB	0.321
onlineA	0.320
uedin-wmt14	0.317

(c) All systems

Table 3.2: Kendall's τ correlations between the segment-level and sentence-level judgments. For a given system we computed the correlation on all pairwise comparisons including the given system. Table 3.2a contains correlations computed on sentence-level judgments where the given system was better, the Table 3.2b is computed on sentence-level comparisons where the given system was worse. Finally, Table 3.2c was computed on all pairwise comparisons including the system. The systems are sorted by the correlation in reverse order.

ranking of systems.

For the explanation of the differences between the overall rankings computed on sentence-level and segment-level judgments we have to look into the data. We want to find candidate translations for which the sentence judgments disagree as much as possible with the segment judgments. To quantify this property we have defined **disagreement quotient**:

$$q_d(s,n) = \frac{win_{seg}(s,n)/(win_{seg}(s,n) + loss_{seg}(s,n))}{win_{sent}(s,n)/(win_{sent}(s,n) + loss_{sent}(s,n))}$$

where s is a system, n is a sentence number, $win_{seg}(s,n)$ is the number of segment-level comparisons where the n-th candidate sentence translated by system s won and $loss_{sent}(s,n)$ being the number of comparisons in which the candidate sentence loses. Finally, $win_{sent}(s,n)$ and $loss_{sent}(s,n)$ are defined similarly for the sentence-level comparisons.

Using this measure we have found candidate sentence translations which were ranked high by segment-level judgments but ranked low by the sentence-level judgments. We have listed candidate sentences with the highest **disagreement quotient** and tried to analyze the cause of the disagreement between the sentence-level and segment-level judgments. In the following, we present some of these sentences with comments. The extracted segments which were ranked are in bold.

Source: Airlines began charging for **the first and second checked bags** in 2008.
Candidate: Letecké linky začaly nabíjení pro **první a druhý odbavených zavazadel** v roce 2008.
(Sentence 715, online-B)

The translation of the segment is relatively good, the case of the noun phrase is wrong but the meaning could be understood. The reason why the whole sentence was ranked poorly is probably the word "nabíjení" ("Charging a battery" in English), which is obviously a wrong lexical choice of the MT system. Unfortunately this word is not covered by the only ranked segment. A similar problem is also in the following sentence:

Source: I want to fulfil **my role of dad and husband**.
Candidate: Chci, aby splnil **svou roli táty a manžela**.
(Sentence 559, cu-bojar)

The translation of the segment is perfect but the subject of the candidate translation is wrongly expressed as the third person. The whole sentence is therefore correctly ranked as a poor translation. And again, this is not covered by the extracted segments.

Source: **Samsung, Huawei and HTC** all manufacture phones that operate on **Google's Android operating system**, which competes fiercely **with Apple and Microsoft mobile products**.
Candidate: **Samsung, Huawei a HTC** všechny výrobní telefony, které pracují **android operační systém Google**, který konkuruje **zuřivě a Applu Microsoftu mobilním produktům**.
(Sentence 484, CU-TectoMT)

The problem here is again in the predicate. The verb "manufacture" is wrongly translated as the adjective "výrobní" ("manufactured" in English).

In all the previous examples, a predicate was wrongly translated but unfortunately it was not covered in the extracted segments and therefore reflected in segment-level judgments. This seems to be the main disadvantage of this method, extracted segments sometimes do not cover predicates which are very important for annotator when judging the overall quality of the sentence.

We have also listed candidate translations with the lowest **disagreement quotient** (a candidate was highly ranked by sentence-level judgments but ranked low by system-level judgments). However, these candidates were not so interesting and we cannot see any general pattern there. We feel that the sentences which are ranked much better in segment-level judgments than in sentence-level judgments are a much more severe problem.

3.2 Evaluating New Systems

Machine translation systems often translate a given short source segment identically. This was one of our main motivations for ranking translations of short

segments. As you saw in the previous section, evaluating the annotated systems using the short segments annotations works reasonably well (despite there are some shortcomings as described above). It would be very useful if we could use the database of annotations to evaluate also unseen systems. The more annotated systems we have in the database the more likely it is that an unseen MT system produces a translation of a short segment which is already in the database. We will call this situation a **hit**. If the translated segment is not already annotated, we will call it a **miss**.

Because we didn't have any spare systems which were not annotated, we did the following trick: in each step, we choose one system and removed its segments from the database of annotations. Then we consider this system as unseen and tried to match the system's segments with the segments left in the database. We call this trick **leave-one-out**. In essence, it is very similar to standard cross validation.

3.2.1 Exact Matching of candidate segments

The most obvious way to evaluate an unseen translation is to compute the **Ratio of wins (ignoring ties)** of all the systems (including the unseen one) only on the hit segments. We extracted the pairwise comparisons from all the segment annotations where the segment of the unseen system was hit and computed the **Ratio of wins (ignoring ties)** only on such extracted comparisons. We performed this experiment for all the systems using the **leave-one-out** trick.

You can see the results of this experiment in Table 3.3. We also report **hit** rate, which is the ratio of hits to all relevant segments (miss + hits).

The average hit rate is 58.8 % which is above our expectations. However, as you can see, the hit rate varies a lot across the systems that were left out. This is caused by the fact that some systems are very similar; they use the same tools and/or training data. For example all the systems cu-bojar, cu-depfix, cu-funky are based on the Moses SMT toolkit and their hit rates are very high (0.74 — 0.93).

As you can see, the obtained orderings of the systems are not very good. The winning system in each of the tables is the one that was left out, which is obviously wrong. Besides that, systems similar to the left-out one get also a much better rank. For example, when the left-out system is one of the systems cu-bojar, cu-depfix and cu-funky, the other two of this group are right below the left-out system. This could be explained by the following statement: MT systems are more likely to agree on a good translation of a segment than on a bad translation.

To support this statement we have performed an analysis of some sentences from the test set. In the following examples of sentences, we have marked the hit segments by **green** and the missed segments by **red**:

system	score
uedin-uncnstr.	0.633
cu-depfix	0.580
uedin-wmt14	0.576
onlineB	0.571
cu-bojar	0.564
cu-funky	0.560
onlineA	0.499
CU-TectoMT	0.425
commercial1	0.377
commercial2	0.329

(a) uedin-uncnstr., hits: 0.67

system	score
commercial1	0.581
uedin-uncnstr.	0.557
onlineB	0.552
uedin-wmt14	0.540
cu-depfix	0.535
cu-funky	0.525
cu-bojar	0.523
onlineA	0.472
CU-TectoMT	0.422
commercial2	0.346

(b) commercial1, hits: 0.28

system	score
commercial2	0.570
onlineB	0.552
uedin-uncnstr.	0.548
cu-depfix	0.535
cu-bojar	0.529
cu-funky	0.524
uedin-wmt14	0.523
onlineA	0.457
CU-TectoMT	0.423
commercial1	0.386

(c) commercial2, hits: 0.28

system	score
CU-TectoMT	0.649
cu-depfix	0.600
cu-bojar	0.584
cu-funky	0.575
onlineB	0.528
uedin-uncnstr.	0.522
uedin-wmt14	0.502
onlineA	0.450
commercial1	0.373
commercial2	0.339

(d) CU-TectoMT, hits: 0.45

system	score
onlineB	0.689
uedin-uncnstr.	0.584
cu-depfix	0.578
cu-funky	0.567
cu-bojar	0.567
uedin-wmt14	0.564
onlineA	0.511
CU-TectoMT	0.406
commercial1	0.360
commercial2	0.320

(e) onlineB, hits: 0.52

system	score
onlineA	0.649
onlineB	0.584
uedin-uncnstr.	0.577
uedin-wmt14	0.568
cu-depfix	0.566
cu-bojar	0.555
cu-funky	0.546
CU-TectoMT	0.411
commercial1	0.365
commercial2	0.318

(f) onlineA, hits: 0.47

system	score
cu-funky	0.630
cu-depfix	0.600
cu-bojar	0.582
uedin-uncnstr.	0.559
onlineB	0.557
uedin-wmt14	0.542
onlineA	0.491
CU-TectoMT	0.446
commercial1	0.378
commercial2	0.331

(g) cu-funky, hits: 0.74

system	score
cu-bojar	0.588
cu-depfix	0.582
cu-funky	0.564
onlineB	0.562
uedin-uncnstr.	0.559
uedin-wmt14	0.545
onlineA	0.498
CU-TectoMT	0.449
commercial1	0.392
commercial2	0.346

(h) cu-bojar, hits: 0.88

system	score
cu-depfix	0.587
cu-bojar	0.570
cu-funky	0.566
onlineB	0.562
uedin-uncnstr.	0.561
uedin-wmt14	0.548
onlineA	0.498
CU-TectoMT	0.449
commercial1	0.394
commercial2	0.345

(i) cu-depfix, hits: 0.93

Table 3.3: The results of evaluating unseen systems using the exact matching and the **leave-one-out** trick. Each subtable is marked by the left-out system. You can also see the hit rates. The table for the system uedin-wmt14 is omitted for the sake of brevity.

Source: So, **still no Words** With Friends, the **online Scrabble-type game** that **actor Alec Baldwin** was playing **on his smartphone in 2011 when he was famously** booted off **an American Airlines jet** for refusing to turn off the device while the plane **was parked at the gate**.

Candidate: Takže **stále žádná slova** s přáteli, **online hra Scrabble typ** že **herec Alec Baldwin** si hrál **na jeho smartphone v roce 2011**, **kdy mu byl slavně** spuštěn z **American Airlines jet** za odmítnutí vypněte zařízení, zatímco letadlo **bylo zaparkováno u brány**.
(Sentence 2976, onlineA)

You can see that the green hits are translated relatively correctly and are understandable. We cannot say the same about the missed segments.

Source: **Amongst other things**, it showed that the Americans even monitored **the mobile phone of German Chancellor Angela Merkel**.

Candidate: **Kromě jiných věcí** ukázalo, že Američané i sledovali **mobilní telefon Germana kancléře Angely Merkelové**.
(Sentence 2945, CU-TectoMT)

The missed segment "Kromě jiných věcí" in this sentence is translated quite well (so the above statement does not hold here). However, the only hit segment here "mobilní telefon" is translated correctly and the missed segment "Germana kancléře Angely Merkelové" is not translated correctly. Judging how easy a segment is to translate, is even more difficult than judging the translations. Nevertheless you may agree with us that the hit segment "the mobile phone" is easy to translate and the missed segment "of German Chancellor Angela Merkel" is much more difficult to translate. We can see this also in the last example:

Source: They had **searched frantically for their missing dog** and posted appeals **on social networking sites** after she had ran **into the quarry following the minor accident**.

Candidate: Měli zoufale **hledal své chybějící psa** a odvolání **na sociálních sítích** poté, co se dostal **do lomu po drobné nehody**.
(Sentence 77, uedin-wmt14)

Both of the hit segments are translated very well and we can say that they are also very easy to translate. On the other hand, the missed segments are understandable but not perfect. Compared to the green segment, they are also more difficult to translate.

Following the manual analysis we can conclude that MT systems are much more likely to agree on the better translations. This however prevents us from matching the segments exactly, because it gives us very overestimated scores for the unseen candidates.

3.2.2 Matching the Closest Segment by Edit Distance

We would like to compute the scores on all the annotated segments to avoid the problem stated in the previous subsection. A natural way to approximate the "correct" ranks of unseen segments is to use the rank of the segment from the database with the closest edit distance. We use the character based Levenshtein distance which is defined as the minimum number of insertions, deletions and substitutions of characters required to transform a given string to another:

$$lev(a,b) = \min_{e \in E(a,b)} (D(e) + S(e) + I(e))$$

where $E(a,b)$ denotes a set of all sequences of edit operations which transform the string a to the string b, and $D(e)$, $S(e)$, $I(e)$ denote number of deletions, substitutions and insertions respectively in the sequence s. If more segments in the database have the same minimal distance to the "unseen" segment, we compute the average of their ranks.

Similarly to the previous experiment we extracted the pairwise comparisons and computed the **Ratio of wins (ignoring ties)**. You can see the results tabulated in Table 3.4. For each left-out system, we also report the average edit distance (AED) of the unseen segments to the closest segments in the database.

The overall rankings of the systems are much more reasonable compared to the exact matching, although they are still not perfect. The scores of systems which were left out are not always the best in the obtained rankings but they are still heavily overestimated. This shows not only that systems are more likely to agree on the better translations than on the worse ones, but also that they produce translations which are closer to better translations than to other translations of similar quality. Our notion of that idea is that a good translation is a point in a high-dimensional space and candidate translations are points around the good translation. Now, let's have a few points around the good translation representing the translations in the database. When we get a new point it is quite likely in this high-dimensional space that the closest point will be the good translation. You can see an illustration of this situation in two dimensions in Figure 3.1. If we do not have enough candidates in the database it is quite likely that for a given unseen candidate the closest candidate is the good translation and not any other candidate translation of similar quality.

The average edit distances vary a lot. Systems cu-bojar, cu-depfix and cu-funky have very low AED (0.2 — 1.7), because they are very similar to each other. Systems CU-TectoMT, onlineA and onlineB are in the middle of the AED range (3.1 — 3.9) and since they are quite solitary in the set of ranked systems we can consider their AEDs as representative values. Systems commercial1 and commercial2 have the highest AEDs. This could be explained by the fact that both of the systems are rule based and produce dissimilar translations to those generated by the statistical based systems. It is interesting, however, that their translations are not even similar to each other. You can see the distribution of the absolute counts with respect to the edit distance to the closest segment plotted in Figure 3.2.

To support the above statement (that the closest segment to an unseen candidate is more likely to be of better quality) we have performed the following analysis: For each left-out system we computed how often the closest segment

system	score
uedin-uncnstr.	0.592
cu-depfix	0.576
onlineB	0.562
cu-bojar	0.558
cu-funky	0.555
uedin-wmt14	0.548
onlineA	0.498
CU-TectoMT	0.446
commercial1	0.397
commercial2	0.347

(a) uedin-uncnstr., AED: 2.0

system	score
cu-depfix	0.566
onlineB	0.553
uedin-uncnstr.	0.551
cu-bojar	0.548
cu-funky	0.545
uedin-wmt14	0.537
commercial1	0.495
onlineA	0.488
CU-TectoMT	0.433
commercial2	0.334

(b) commercial1, AED: 5.4

system	score
cu-depfix	0.559
onlineB	0.549
uedin-uncnstr.	0.546
cu-bojar	0.541
cu-funky	0.539
uedin-wmt14	0.533
commercial2	0.484
onlineA	0.483
CU-TectoMT	0.430
commercial1	0.379

(c) commercial2, AED: 5.7

system	score
cu-depfix	0.566
CU-TectoMT	0.557
onlineB	0.554
uedin-uncnstr.	0.552
cu-bojar	0.548
cu-funky	0.545
uedin-wmt14	0.539
onlineA	0.489
commercial1	0.387
commercial2	0.337

(d) CU-TectoMT, AED: 3.9

system	score
onlineB	0.614
cu-depfix	0.574
uedin-uncnstr.	0.559
cu-bojar	0.556
cu-funky	0.552
uedin-wmt14	0.546
onlineA	0.496
CU-TectoMT	0.444
commercial1	0.395
commercial2	0.345

(e) onlineB, AED: 3.1

system	score
onlineA	0.581
cu-depfix	0.570
onlineB	0.556
uedin-uncnstr.	0.555
cu-bojar	0.553
cu-funky	0.549
uedin-wmt14	0.542
CU-TectoMT	0.441
commercial1	0.391
commercial2	0.341

(f) onlineA, AED: 3.4

system	score
cu-funky	0.597
cu-depfix	0.575
onlineB	0.561
uedin-uncnstr.	0.559
cu-bojar	0.557
uedin-wmt14	0.546
onlineA	0.497
CU-TectoMT	0.445
commercial1	0.396
commercial2	0.346

(g) cu-funky, AED: 1.7

system	score
cu-bojar	0.580
cu-depfix	0.576
onlineB	0.562
uedin-uncnstr.	0.561
cu-funky	0.555
uedin-wmt14	0.548
onlineA	0.499
CU-TectoMT	0.447
commercial1	0.398
commercial2	0.348

(h) cu-bojar, AED: 0.4

system	score
cu-depfix	0.579
onlineB	0.564
uedin-uncnstr.	0.563
cu-bojar	0.561
cu-funky	0.556
uedin-wmt14	0.550
onlineA	0.501
CU-TectoMT	0.448
commercial1	0.399
commercial2	0.349

(i) cu-depfix, AED: 0.2

Table 3.4: The results of evaluating unseen systems using the edit distance matching and **leave-one-out** trick. Each subtable is marked by the left-out system. The abbreviation AED stands for average edit distance which is computed across all segments. The table for the system uedin-wmt14 is omitted for the sake of brevity.

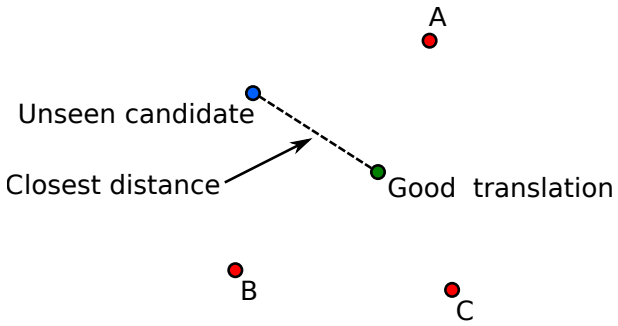


Figure 3.1: An example of a good translation with only a few candidate translations around it. If the number of dimensions is higher than the number of candidates, it is intuitively quite likely that the closest point to the new unseen candidate is the good translation.

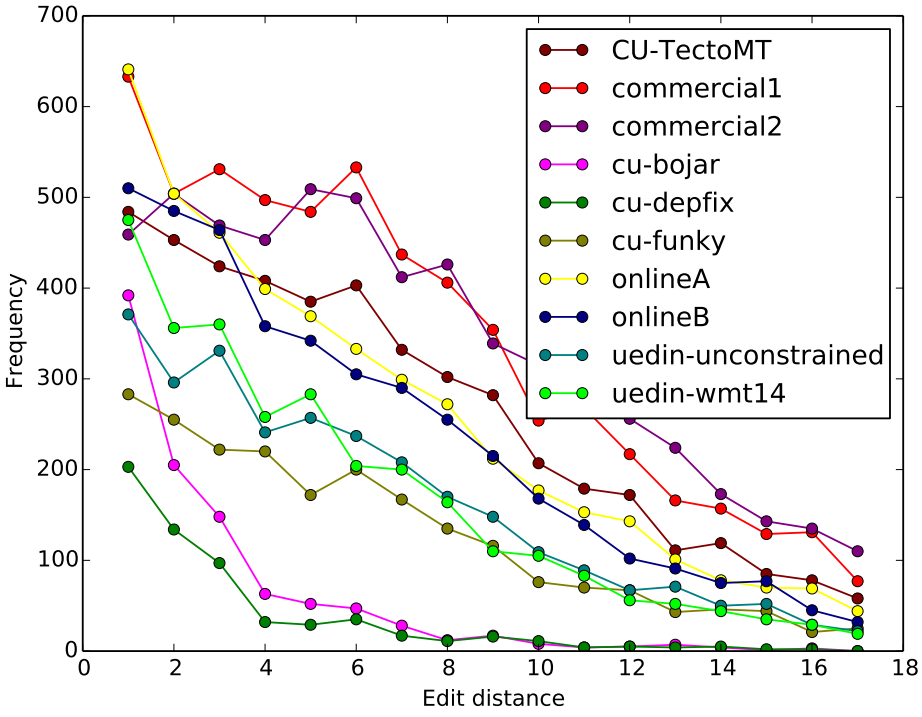


Figure 3.2: Absolute counts of segments with respect to the edit distance to the closest segment

Unseen system	Worse	Equal	Better
commercial1	28.0 %	18.2 %	53.8 %
commercial2	23.4 %	16.8 %	59.8 %
cu-bojar	22.8 %	30.5 %	46.7 %
cu-depfix	34.2 %	31.4 %	34.4 %
cu-funky	29.3 %	22.9 %	47.8 %
CU-TectoMT	26.2 %	17.8 %	56.0 %
onlineA	28.7 %	19.1 %	52.2 %
onlineB	33.5 %	19.9 %	46.6 %
uedin-unconstrained	32.8 %	21.5 %	45.7 %
uedin-wmt14	32.1 %	21.9 %	46.0 %
All	28.5 %	19.7 %	51.9 %

Table 3.5: Comparisons of the unseen and the closest segments' ranks. This table shows how often the rank of the closest segment in the database was worse, equal or better than the original rank of the "unseen" segment. These relative frequencies were computed only on the missed segments (which weren't already in the database).

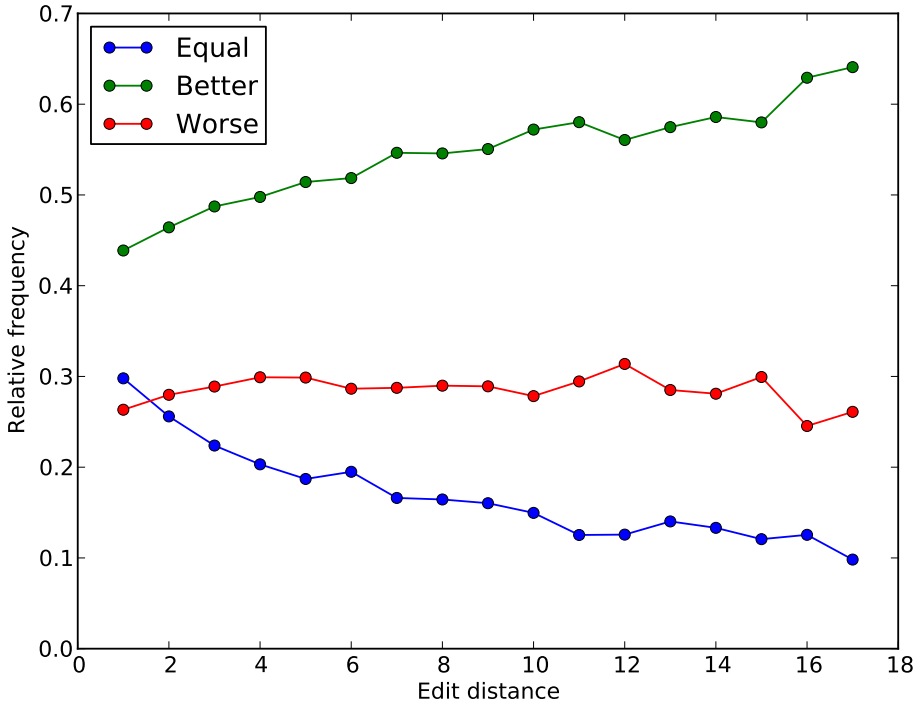


Figure 3.3: Comparisons of the unseen and the closest segments' ranks with respect to the edit distance. The results in this figure are computed on all of the systems.

Unseen segment	Closest segment	D	C
s dokumenty upřednostňujícím doprovodem hradu	se dokumenty favorizovat doprovod hradu	18	W
ze 120 domova m2	z 120 m2 domova	7	B
vaše ústa ohřívá protein	vaše ústa zahřívání bílkovin	10	B
videokonference	videokonferenci	1	B
popřel užívání kokainu a	popřela užívání kokainu a	1	W
přibližně šedesát- drah kilometru	přibližně šedesát-dráha kilometru	3	E
v Liverpoolském porotním soudu	Liverpool Korunního soudu	11	B
je nesmysl v gravitaci filmu	Je nesmysl ve filmu gravitace	15	B
Pak 64,23 % oprávněných voličů	pak 64,23 % oprávněných voličů	1	B
podle DPA agentury	podle DPA agentury té	3	W

Table 3.6: Example candidates and the closest segments. The second last column (**D**) stands for distance and contains distances of the unseen candidates to the closest segments. The last column (**C**) stands for comparison; the closest segment is either worse (**W**), equally good (**E**) or better (**B**) than the original unseen segment.

has better, equal or worse rank than the "unseen" segment (We can actually do that, because we know the true rank of the segment removed from the database). We computed these relative frequencies only on the missed segments (the closest segment was not the same segment). You can see this analysis performed for the individual systems in Table 3.5. In total, more than a half of the closest segments have a better rank than the original segments and only 20.6 % of the segments have the same rank. This is very poor because it means that our approximation method which ranks unseen translations has the accuracy of only 20.6 %. This accuracy does not differ much for individual systems but there is an expected trend of similar systems (cu-bojar, cu-depfix) having this accuracy slightly higher.

You can also see how these relative frequencies vary with the change of the edit distance in Figure 3.3. As you can see, the relative number of closest systems which are ranked better grows quite significantly with the edit distance. The relative number of the worse segments is quite stable (around 0.3) and does not change significantly with the edit distance. The relative number of closest segments which are equally ranked as the source segment is decreasing with the edit distance. For example, for the segments whose edit distance to the closest segment is 17, only 10 percent of the closest segments is equally ranked, which is very poor.

We also listed a few example candidate segments in Table 3.6 together with the corresponding closest segments from the database and their distances. We also report whether the closest segment was ranked better, equal or worse than the "unseen" one.

We have to unfortunately conclude, that the proposed method, which reuses the database for evaluating unseen translations, does not work. We analyzed the

System	Score
cu-depfix	0.305
cu-funky	0.302
uedin-unconstrained	0.302
cu-bojar	0.300
uedin-wmt14	0.296
onlineB	0.289
onlineA	0.259
CU-TectoMT	0.225
commercial1	0.176
commercial2	0.160

(a) SegRanksBLEU, correlation: 0.9745

System	Score
cu-depfix	0.221
cu-bojar	0.221
cu-funky	0.221
uedin-unconstrained	0.220
uedin-wmt14	0.215
onlineB	0.207
onlineA	0.187
CU-TectoMT	0.157
commercial1	0.114
commercial2	0.102

(b) BLEU, correlation: 0.9751

Table 3.7: Overall system ranking according to SegRanksBLEU and BLEU scores. Please see Chapter 4 for the details of computation of the reported correlations.

results and the main cause of this failure seems to be that the systems tend to agree on better translations and their translations tend to be more similar to better translations in the database so we cannot predict their rank accurately.

3.2.3 Enhancing Reference Translation

Following the conclusions from the previous two subsections, it seems that errors in machine translation are very unique. Any database of bad examples (bad translations) is therefore very sparse. It is a very well known fact that the number of possible correct translations is very high (Bojar et al., 2013). It seems, however, that the number of bad translations is much higher and therefore it makes more sense to use a database of good translations.

In the following experiment, we therefore use only the good candidate segments from the annotated database. The approach used here is, however, different from the previous experiments. We would like to measure how similar candidates are to the good translations from the database. This is very similar to what automatic metrics do when measuring similarity between a candidate and reference translations. We have therefore decided to use one of the standard metrics – BLEU – to measure this similarity. First, we are going to introduce the metric and then we are going to customize it for our experiment.

Metric BLEU was developed by Papineni et al. (2002) and it is the most used metric in the machine translation evaluation. It is defined as the geometric mean of n-gram precisions for $n \in \{1 \dots N\}$, where N is usually 4. More precisely, for a candidate c and reference translations r_i where $i \in I$, let the clipped count of an n-gram g be defined as follows:

$$\text{count}_{clip}(g, c, r) = \min\left(\text{count}(g, c),\ \max_{i \in I}\left(\text{count}(g, r_i)\right)\right)$$

where count(g, s) denotes the count of n-gram g in the sentence s. The modified precision p_n is then defined as:

$$p_n = \frac{\sum_{g \in \text{n-grams}(c)} \text{count}_{clip}(g, c, r)}{\sum_{g \in \text{n-grams}(c)} \text{count}(g, c)}$$

Using the computed n-gram precisions, we can compute the final BLEU score:

$$BLEU = BP \cdot \exp\left(\frac{1}{N}\sum_{i=1}^{N} \log p_n\right)$$

where BP is the brevity penalty (meant to penalize short outputs, to discourage improving precision at the expense of recall) defined as follows:

$$BP = \begin{cases} 1 & \text{if } |c| > |r| \\ \exp(1 - |r|/|c|) & \text{if } |c| \leq |r| \end{cases}$$

where $|c|$ and $|r|$ are the lengths of the candidate and reference translations respectively. In the case of multiple reference translations, $|r|$ could be the average length, the maximum length or the length closest to the candidate c.

This experiment consists of two steps. In the first step we select good segment translations from the short segments database. In the second step we use the selected good segment translations to enhance reference translations when computing BLEU.

Since the assigned ranks in the database are relative, we cannot know which segments are really good in terms of the absolute quality. We have to assume that there is at least one good candidate translation among the ranked candidates and consider all candidate segments with the best rank as the good translations. We select these candidate segments for each source segment for each sentence.

Finally, we use the selected good segments as the reference translations in addition to the original reference sentence translated by a human expert. Since the new references are only short segments and do not cover a whole sentence, we use only the length of the original reference sentence in the computation of the brevity penalty. To distinguish this method from the standard BLEU with the single official reference translation, we will call this method SegRanksBLEU.

Please note, that introducing the new reference translations, which do not change the brevity penalty, can only increase the clipped counts of n-grams occurring in the short segments. Candidates will be rewarded for having n-grams which are also in the good segment translations in the database.

You can see the overall rankings of the evaluated systems as given by SegRanksBLEU and BLEU in Table 3.7. As expected, the SegRanksBLEU scores are indeed much higher than BLEU. However, the reported system level correlations of these two metrics are almost equal (correlation of SegRanksBLEU is even a little bit lower).

3.3 Tuning Systems

Automatic metrics are not used only for evaluating already developed systems on test sets. They are also often used when tuning a system to choose the parameters

of a model which gives the best metric score computed on a development test set. One of the methods utilizing automatic metrics for system tuning is Minimum Error Rate Training (MERT). We will describe the theory behind MERT in the following subsection a then we will experiment with it using the short segments rank database.

3.3.1 Minimum Error Rate Training

Most of the statistical machine translation systems model the posterior probability of producing a sentence translation e given a source sentence f using a log-linear model (Och and Ney, 2002):

$$P(e \mid f) = \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(e,f)\right)}{\sum_{e'} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(e',f)\right)} = \frac{\exp\left(\boldsymbol{\lambda} \cdot \boldsymbol{h}(e,f)\right)}{\sum_{e'} \exp\left(\boldsymbol{\lambda} \cdot \boldsymbol{h}(e',f)\right)} \tag{3.1}$$

where the vector $\boldsymbol{h}(e,f) = \{h_1(e,f), h_2(e,f), \ldots, h_M(e,f)\}$ is a feature vector and the vector $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_M\}$ is a feature weight vector. It can be easily shown that when searching for the best candidate e_{best}, the above log-linear formula can be simplified to the following:

$$e_{best} = \arg\max_{e} \{P(e \mid f)\} = \arg\max_{e} \{\boldsymbol{\lambda} \cdot \boldsymbol{h}(e,f)\} \tag{3.2}$$

Now the problem with this log-linear model is how to find the weight vector $\boldsymbol{\lambda}$ which would give the best translations. And this is what the MERT method solves.

Let $F = \{f_i;\ i \in I\}$ be a held out corpus in the source language and $R = \{r_i;\ i \in I\}$ be a corresponding reference translation. Traditionally, the best weight vector $\boldsymbol{\lambda}$ can be found using the maximum likelihood principle:

$$\hat{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\lambda}} \left\{ \prod_{i \in I} P_{\boldsymbol{\lambda}}(e_i \mid f_i) \right\} \tag{3.3}$$

This optimization problem has some very nice properties: there is one global optimum and there are algorithms which converge to this optimum. Och (2003), however, argues that that there is no proof that this method gives the best weights with respect to the translation quality.

He suggested to choose the weight vector $\boldsymbol{\lambda}$ which produces the best translation of the heldout data. We can measure the quality of the translation using an automatic metric $m(E,R)$, where E is the set of candidate translations produced by the MT system and R is a set of reference translation. We are therefore going to minimize the score (or minimize it in the case the metric is error based and returns lower scores for better translations) of the metric:

$$\hat{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\lambda}} \left\{ m(\hat{E}(F, \boldsymbol{\lambda}), R) \right\} \tag{3.4}$$

where $\hat{E}(F, \boldsymbol{\lambda})$ is the corpus F translated with the weights $\boldsymbol{\lambda}$ and $\hat{e}(f, \boldsymbol{\lambda})$ is the translation of a sentence f with weights $\boldsymbol{\lambda}$:

$$\hat{E}(F, \boldsymbol{\lambda}) = \{\hat{e}(f_i, \boldsymbol{\lambda});\ i \in I\} \tag{3.5}$$

$$\hat{e}(f, \boldsymbol{\lambda}) = \arg\max_{e} \{\boldsymbol{\lambda} \cdot \boldsymbol{h}(e, f)\} \tag{3.6}$$

The criterion (3.4) is, however, not so easy to optimize. It contains an argmax operation and therefore it is not smooth and it is not possible to use a gradient based method. It has also a lot of local maxima. Och (2003) therefore suggested to use Powell's algorithm (Powell, 1964) which does not need gradient to optimize a function.

In each step, Powell's algorithm starts at a point from which it goes in several directions (possibly also in random directions) and optimizes the function along the lines which originate at the start point using a given line optimization method. The most optimal point found on the lines is then used as the new starting point in the next step.

The Powell's algorithm depends on an efficient line optimization method. Och proposed a method which takes advantage of log-linear model properties. In the following we limit the space of the optimization problem to a line defined by the following equation:

$$\boldsymbol{\lambda}(\gamma) = \boldsymbol{\lambda'} + \gamma \boldsymbol{d} \tag{3.7}$$

where $\boldsymbol{\lambda'}$ defines a starting point and the vector $\boldsymbol{d}$ defines a direction of the line. Now, we can reformulate the limited optimization problem:

$$\hat{\gamma} = \arg\max_{\gamma \in \mathbb{R}} \left\{ m(\hat{E}(F, \boldsymbol{\lambda}(\gamma)), R) \right\} \tag{3.8}$$

When we apply the line equation 3.7 in the argmax operator which chooses the best translation in the equation 3.6, we get the following.

$$\begin{aligned}
\hat{e}(f, \boldsymbol{\lambda}(\gamma)) &= \arg\max_{e} \{\boldsymbol{\lambda}(\gamma) \cdot \boldsymbol{h}(e, f)\} && (3.9)\\
&= \arg\max_{e} \{\boldsymbol{\lambda'} \cdot \boldsymbol{h}(e, f) + \gamma \boldsymbol{d} \cdot \boldsymbol{h}(e, f)\} && (3.10)\\
&= \arg\max_{e} \{t(e, f) + \gamma \cdot m(e, f)\} && (3.11)
\end{aligned}$$

where the scalars $t(e, f)$ and $m(e, f)$ are constant for a given candidate e. This means that each candidate e specifies a line given by the equation $x = t(e, f) + \gamma \cdot m(e, f)$, which computes the candidate score for the parameter γ. The argmax operator in the equation (3.11) then chose the candidate whose line is the highest in a given γ. You can see this situation illustrated in Figure 3.4. The line optimization method finds all the intersections of the lines where the best candidate (with the highest model score) changes. The intersections found for each sentence in the heldout data are then merged to obtain intervals of γ in which the γ does not change the set of the highest score candidates. For each interval, the metric score is computed and the optimal γ is chosen from the interval with the highest metric score. Since the metrics are often computed from sentence decomposable statistics, we can easily update the metric's score in each interval boundary by subtracting the old sentence's statistics and adding the new sentence's statistics.

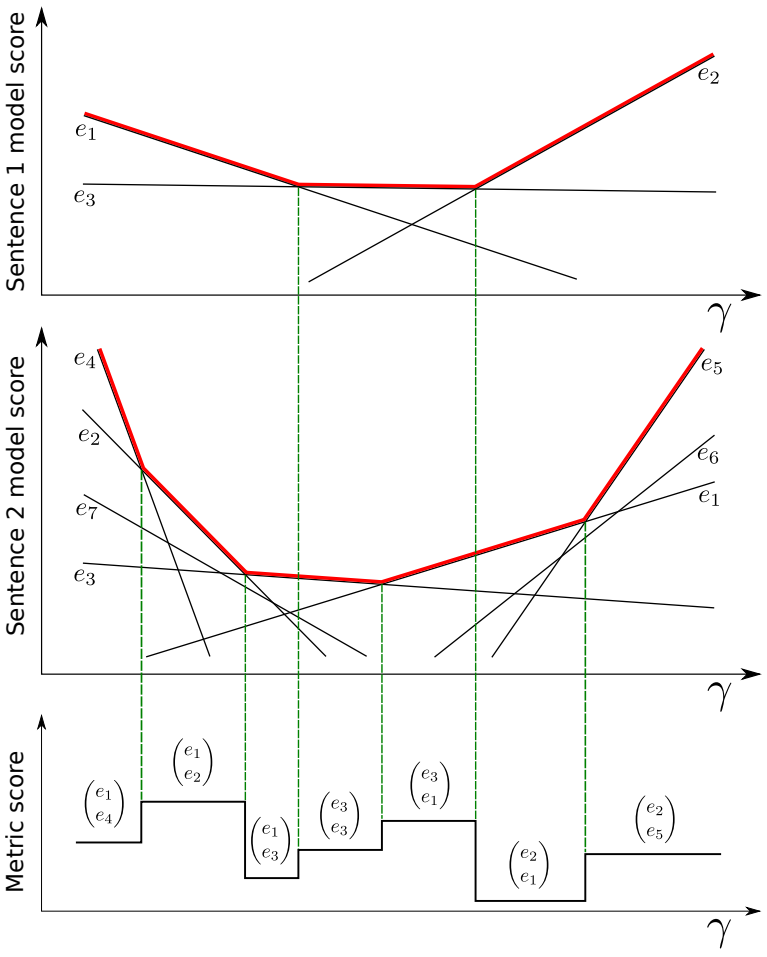


Figure 3.4: An illustration of the line optimization in MERT. You can see two sentences and their candidate translations represented by the lines. The best candidate in each interval is marked by the red line. The bottom part of the figure illustrates how the metric score depends on γ.

The last difficulty in the MERT method is that we cannot easily enumerate all the candidate sentences. For that reason the n-best list approximation is used: the decoder produces n best translations for each sentence and only these n translations are considered during the optimization. A problem arises when the optimal weight vector found during the optimization gets too far from the weight vector used to produce the initial n-best list. For that reason a new n-best list is produced with the new weights; it is merged with previous n-best lists and the optimization process is run again in new a MERT iteration. When the optimal weight vector converges or a maximum number of iterations is reached, the MERT method ends.

3.3.2 Experiments with MERT

The MERT method is most often used with BLEU metric even though it does not have the best correlation with human judgments (see Chapter 4 for more details). However, there are a lot of experiments being done utilizing also other automatic metrics in recent time. In WMT11 (Callison-Burch et al., 2011), there was a shared task in which participants tuned a common system with different metrics. The tuned systems were then evaluated by humans and some of them outperformed the baseline system tuned with BLEU.

It is not feasible to employ any sort of human evaluation directly in the MERT process. On one hand, human evaluation is very slow and expensive, on the other hand, MERT requires to evaluate very long n-best list in each iteration. There are some suggestions to do the manual evaluation in a clever way and lower the amount of manual work. For example Zaidan and Callison-Burch (2009) noticed that a lot of short segments are repeated in a n-best list and therefore suggest to extract these short segments from a n-best list in each MERT iteration and let humans rank them. (Actually our short segment extraction method was partially inspired by this work). However they did not try this method in an actual MERT run yet for lack of resources.

We believe that a much less expensive way to introduce an element of human evaluation in the MERT method is to use some sort of semi-automatic metric in which a certain amount of manual work is needed at the beginning and then the metric evaluates translations automatically. In this subsection, we therefore experiment with previously introduced metrics which rely on the collected database of short segment ranks.

Tuned System

The system we tried to tune is the system cu-bojar (Tamchyna et al., 2014), which we also used in previous sections as one of the evaluated systems. This system is Moses-based and combines several different approaches:

- factored phrase-based Moses model
- domain-adapted language model
- deep-syntactic MT system TectoMT

The parallel data used to train the phrase tables consist of 14.83 million parallel sentences taken from the CzEng corpus (Bojar et al., 2012) and 0.65 million of sentences taken from the Europarl corpus (Koehn, 2005). The monolingual data used to train language models consist of 215.93 million sentences taken from the Czech side of the CzEng corpus and from 5 sources of a news domain.

Authors of the system used their baseline systems to translate WMT test sets from years 2012–2014. These translations were then used to retrieve similar Czech sentences using information retrieval techniques which were then used to train the domain-adapted language model.

The deep-syntactic MT system TectoMT was used to translate WMT test sets from years 2007–2014. These new synthetic parallel training data were then used to train an additional phrase table.

Please note that both the domain-adapted language model and the TectoMT phrase table were also trained on the data we use as development set and test set so that tuning can assign appropriate weights to them.

There are 15 component weights to tune in total. Tamchyna et al. (2014) of course tuned their system cu-bojar when participating in the translation shared task, but we use their system in the state it was before the tuning to tune it ourselves.

We used the implementation of MERT which is distributed with Moses toolkit[1]. It is implemented in C++ and new metrics are created by subclassing the base *Scorer* class. Since the annotated database is stored in Python data structures and since a development in Python is much easier and faster, we have implemented new *PythonScorer* which is a universal wrapper around an arbitrary Python scorer class, implemented using the Python C API.

Metric Variants

Our original idea was that we will use the alignment produced by the Moses decoder when translating the n-best list to project the extracted short source segments to the target side to have candidates which would be extracted the same way as the ranked segments in the database. Unfortunately the alignment produced by the Moses decoder is very sparse and unreliable (which may be caused by a bug in the code) so we had to get along without the alignment.

The naive approximation is to just test whether the ranked candidate segments from the database also occur in evaluated sentences. The first metric we use in the MERT experiment is therefore very similar to the **Exact Matching** method in Section 3.2.1. For each ranked source segment we test if any of its candidate segment occurs in the evaluated sentence. If it does, we extract all the pairwise comparisons from the matched segment. If more candidate segments occur in the evaluated translation, we choose the longest segment (assuming that the shorter segments are just substrings of the longest one). The final score is then computed as **Ratio of wins (ignoring ties)**. We call this metric ExactMatch in the following.

Even if the hit rate (percentage of segment candidates which are already ranked in the database) computed on a whole n-best list would be 100 %, the ExactMatch metric still does not evaluate whole sentences and could be too

[1] `http://www.statmt.org/moses/`

coarse and harsh. Moreover, if the hit rate drops during the tuning, the metric score would be computed on a very small percentage of the development set and it would be very unstable. To ensure that the tuning is stable, we interpolate all the metrics in this section (if not said otherwise) together with BLEU with equal weights. We also tried to tune using the ExactMatch metric solely but the tuned system translated very badly and the hit rate dropped very low during the tuning. We could explain this by that the system was tuned to translate a few ranked segments well (so these segments were hits) but other segments were missed and translated badly. The optimal metric score was therefore computed on a very small fraction of the development set and did not reflect the overall quality of the translation.

Since we do not have the alignment and cannot extract the candidate segments exactly, we unfortunately cannot use the method introduced in Subsection 3.2.2 which matches the closest segment in the database by edit distance. To avoid the shortcomings of the ExactMatch metric (the metric is not computed on all the extracted source segments and an unseen system is more likely to cause a hit in the database with better translations), we propose another variant called PenalizeUnknown. This variant differs from ExactMatch in that it considers all missed segments as the worst translations. We agree that the assumption that all unseen and unranked segments are wrong is not correct, but this approach could increase the hit rate. The question is then, whether we prefer to have a system which produces a lot of segments which were already ranked (even badly) or a system which produces a lot of unranked segments which we hope to be of better quality.

The last variant we experiment with is SegRanksBLEU metric introduced in Subsection 3.2.3. Because this metric is already based on BLEU metric (and use the reference translation) we do not interpolate it with BLEU anymore.

Results and Analysis

You can see the results of the tuned system in Table 3.8. We used the tuned systems to translate the test set (newstest2012) and then we evaluated these translations automatically and manually. For the automatic evaluation we used metrics BLEU (Papineni et al., 2002) and CDER (Leusch et al., 2006). We have also conducted a small scale manual evaluation. For each evaluated system, we randomly sampled 100 sentences which were translated differently[2] to the baseline and by the evaluated system. Then we compared manually the sampled sentences with the corresponding translations produced by the baseline system. The task was to choose which sentence is better or whether they are of the same quality. We report how many of the sampled sentences were better and how many of them were worse than the corresponding baseline translation.

You can see that none of the metric variants outperformed the baseline system tuned solely to BLEU in the automatic evaluation. This was expected, since the best performing system according to BLEU should be the one which was tuned by BLEU. However, you can see that the system tuned by PenalizeUnknown has both the BLEU and CDER scores only a little bit lower.

[2] 58.6 % of all test set sentences were translated differently to the baseline by SegRanksBLEU, 62.4 % by PenalizeUnknown and 83.4 % by ExactMatch

Tunable metric	#Mert iterations	Automatic Evaluation BLEU	Automatic Evaluation CDER	Manual Evaluation Better	Manual Evaluation Worse
BLEU (baseline)	11	**0.1782**	**0.3855**	—	—
ExactMatch	20	0.1637	0.3674	22 %	38 %
PenalizeUnknown	8	0.1772	0.3850	**34 %**	**25 %**
SegRanksBLEU	8	0.1753	0.3835	29 %	49 %

Table 3.8: Results of systems which were optimized to a SegRanks based metric. The items in the first column specify the metric which was used when tuning the system on the development test. The columns BLEU and CDER contain just scores of these metrics computed on the test set translated with the tuned weights. The last two columns contain percentages of better and worse sentences compared to the baseline system in the manual evaluation.

In the manual evaluation, the only system which translated more better sentences than worse sentences compared to the baseline system is the system tuned to PenalizeUnknown. This means that forcing the system to produce known and evaluated segments when translating development set helps to chose better weights. This, however, also means that we discouraged the tuned system to produce unknown and maybe better translations. The best hypothetical translation according to the optimized metric which the tuned system can produce during MERT consists of the best ranked segments from the database. However, there certainly exist better translations.

We should say here, that this experiment should be performed more precisely to obtain reliable results. The problem with MERT is that it is not deterministic in two aspects: First, MERT method can converge to only a local optimum, this depends on the point it started. Second, the used implementation of Powell's algorithm optimize the metric value along the axis and also along random lines. Two MERT runs with different random seed can therefore find different weights. The fact that system tuned to PenalizeUnknown was better than baseline could be therefore caused only by a chance. To do it more precisely, we would have to run the MERT process more times starting at different points and with different random seeds (but with the same starting conditions for each optimized metric). However, we did not do that this way, because we would have to manually evaluate each of the MERT runs a we did not have enough resources for that.

To see the differences between ExactMatch and PenalizeUnknown, we have plotted the values of hit rates computed in each MERT iteration in Figure 3.5. You can see that PenalizeUnknown gets to the hit rate of 0.7 very quickly (in the sixth iteration) and it's growth is quite stable. This is however expected, since the objective function also indirectly optimizes the hit rate. The hit rate in ExactMatch also grows but much more slowly. It stabilizes around the of value 0.5. It is good that the final value is quite high so the ExactMatch is computed on a nonnegligible amount of data. However, it takes many more iterations for ExactMatch to get to the value of 0.5 that it takes for PenalizeUnknown to get to the value of 0.7.

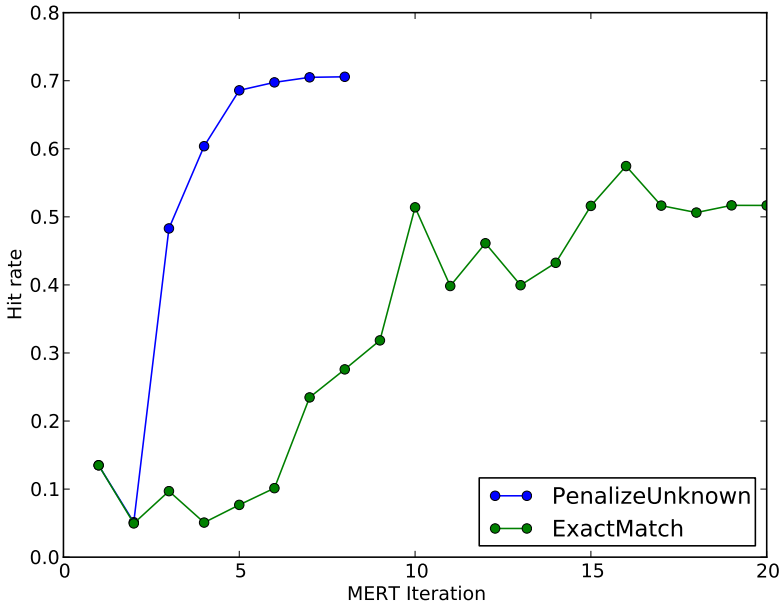


Figure 3.5: Hit rates computed on the n-best lists produced in each MERT iteration

Chapter 4

Evaluation and Comparison of Automatic Metrics

Automatic machine translation metrics play a very important role in the development of MT systems and their evaluation. There are many different metrics of diverse nature and one would like to assess their quality. For this reason, the Metrics Shared Task is held annually at the Workshop of Statistical Machine Translation[1], starting with Koehn and Monz (2006) and following up to Bojar et al. (2014). In WMT13, we took over the organization of the task (Macháček and Bojar, 2013) and continued to do that also in WMT14 (Macháček and Bojar, 2014). In this chapter, we present the results of WMT14 metrics shared task with some additional information and comments.

In this task, we asked metric developers to score the outputs of WMT14 Shared Translation Task (Bojar et al., 2014). We have collected the computed metrics' scores and use them to evaluate the quality of the metrics. There are actually two subtasks: a system-level task and a sentence-level task.

- **System-level task.** In this subtask, participants compute one score for the whole test set, as translated by each of the systems. We then measure the correlation of those scores with the systems' official human scores. The goal of metrics in this subtask is to give an overall ranking of the systems as close as possible to the official ranking according to human scores in each direction.

- **Sentence-level task.** In this subtask, participants compute one score for each sentence of each system's translation. We then measure the correlation of these scores with pairwise human judgements. The goal of metrics in this subtask is to compare two candidate sentences in the same way as humans would.

In the previous years, a lot of metrics performed very well in the system-level task. In some of the translation directions, they predicted the overall ranking of systems almost perfectly. However, there is still space for metrics to improve. In contrast to the system-level task, the sentence-level task is much more difficult. The correlations in previous years were very low. It is therefore very interesting to see whether metrics perform better this year.

[1] `http://www.statmt.org/wmt13`

The systems' outputs, human judgements and evaluated metrics are summarized in Section 4.1. In Section 4.2, we describe the participated metrics in more detail. The quality of the metrics in terms of system-level correlation is reported in Section 4.3. The sentence-level correlations with a detailed discussion and a slight change in the calculation compared to the previous year is reported in Section 4.4.

4.1 Data

We used the translations of MT systems involved in WMT14 Shared Translation Task together with reference translations as the test set for the Metrics Task. This dataset consists of 110 systems' outputs and 10 reference translations in 10 translation directions (English from and into Czech, French, German, Hindi and Russian). For most of the translation directions each system's output and the reference translation contain 3003 sentences. For more details please see the WMT14 overview paper (Bojar et al., 2014).

4.1.1 Manual MT Quality Judgements

During the WMT14 Translation Task, a large scale manual annotation was conducted to compare the systems. We used these collected human judgements for the evaluation of the automatic metrics.

The participants in the manual annotation were asked to evaluate the system outputs by ranking translated sentences relative to each other. For each source sentence that was included in the procedure, the annotator was shown the outputs of five systems to which he or she was supposed to assign ranks. Ties were allowed.

These collected rank labels for each five-tuple of systems were then interpreted as 10 pairwise comparisons of systems and used to assign each system a score that reflects how high that system was usually ranked by the annotators. Please see the WMT14 overview paper for details on how this score is computed. You can also find inter- and intra-annotator agreement estimates there.

4.1.2 Participants of the Metrics Shared Task

Table 4.1 lists the participants of WMT14 Shared Metrics Task, along with their metrics. We have collected 23 metrics from a total of 12 research groups.

In addition to that we have computed the following two groups of standard metrics as baselines:

- **Mteval.** The metrics BLEU (Papineni et al., 2002) and NIST (Doddington, 2002) were computed using the script `mteval-v13a.pl`[2] which is used in the OpenMT Evaluation Campaign and includes its own tokenization. We run `mteval` with the flag `--international-tokenization` since it performs slightly better (Macháček and Bojar, 2013).

- **Moses Scorer.** The metrics TER (Snover et al., 2006), WER, PER and CDER (Leusch et al., 2006) were computed using the Moses scorer which

[2] `http://www.itl.nist.gov/iad/mig//tools/`

Metric	Participant
AMBER	National Research Council of Canada (Chen and Kuhn, 2011)
APAC	Hokkai-Gakuen University (Echizen'ya, 2014)
BEER	ILLC – University of Amsterdam (Stanojevic and Sima'an, 2014)
BLEU-NRC	National Research Council of Canada (Chen and Cherry, 2014)
DiscoTK-*	Qatar Computing Research Institute (Guzman et al., 2014)
ELEXR	University of Tehran (Mahmoudi et al., 2013)
LAYERED	IIT, Bombay (Gautam and Bhattacharyya, 2014)
Meteor	Carnegie Mellon University (Denkowski and Lavie, 2014)
Parmesan	Charles University in Prague (Barančíková, 2014)
RED-*	Dublin City University (Wu and Yu, 2014)
tBLEU	Charles University in Prague (Libovický and Pecina, 2014)
UPC-IPA	Technical University of Catalunya (Gonzàlez et al., 2014)
UPC-STOUT	Technical University of Catalunya (Gonzàlez et al., 2014)
VERTa-EQ	University of Barcelona (Comelles and Atserias, 2014)
VERTa-W	University of Barcelona (Comelles and Atserias, 2014)

Table 4.1: Participants of WMT14 Metrics Shared Task

is used in the Moses model optimization. To tokenize the sentences we used the standard tokenizer script as available in the Moses toolkit.

We have normalized all metrics' scores such that better translations get higher scores.

4.2 Descriptions of the Metrics

In this section, we present short descriptions of some interesting metrics which participated in the shared task. Please see the cited papers for more details about the metrics. Note that the description of the BLEU metric has been already presented in Subsection 3.2.3.

4.2.1 AMBER

Metric AMBER is developed by Chen and Kuhn (2011). The abbreviation stands for "A Modified Bleu, Enhanced Ranking metric". This metric is based on BLEU metric because, as the authors explain, BLEU

- is language independent,
- can be computed quickly (important for tuning),
- seems to be the best tuning metric.

They want to improve the correlation with human judgments while preserving the advantages which make BLEU so popular. The basic formula for this metrics follows:

$$AMBER = score \cdot penalty$$

where *penalty* is a weighted geometric average of 10 various penalty measures and *score* is given by:

$$score = \theta_1 \cdot AvgP + \theta_2 \cdot Fmean + \theta_3 \cdot AvgF$$

where $AvgP$ is a geometric average of n-gram precisions (this is the same as in BLEU), $Fmean$ is the F-measure computed on the average n-gram precision and average n-gram recall (averages are computed across $n \in \{1, \ldots, N\}$) and $AvgF$ is an average (across $n \in \{1, \ldots, N\}$) of F-measures computed on n-gram precision and n-gram recall for given n.

This is the basic computation but the authors propose many more enhancements, for example they propose 8 different preprocessing (normalization and tokenization) methods which are used to compute AMBER metric on each method separately and then return the average of scores computed after applying all the methods. All free parameters of this metric were manually tuned on a development set.

While we agree that the authors managed to preserve the key properties of BLEU mentioned above, we think that one of the most popular properties of BLEU is its implementation simplicity which is definitely not preserved in AMBER.

4.2.2 BEER

This metric is proposed by Stanojevic and Sima'an (2014). The abbreviation stands for "BEtter Evaluation as Ranking". Their metric model can employ various features which measure the similarity of candidate and reference translations from various aspects. Individual features are then combined using a simple linear interpolation of feature functions:

$$score(h, r) = \sum_i w_i \cdot \phi_i(h, r) = \boldsymbol{w} \cdot \boldsymbol{\phi}(h, r)$$

They propose two groups of features. For adequacy features, they use precision, recall and F1-score features for each of the following entities: function words, content words, all words and character n-grams for $n \in \{1, \ldots, 6\}$. The total number of all adequacy features is 27.

The second group of features are ordering features. They represent orderings as permutations. One of the ordering features is Kendall's *tau* distance to monotone permutation. Other 5 ordering features are based on permutation trees introduced by Zhang and Gildea (2007).

They tune the feature weights to get the best correlation with human sentence-level judgments. Let h_{good} and h_{bad} be two hypothesis translations of a source sentence with reference translation r and, let h_{good} be ranked better than h_{bad} by human judgments. Given that the metric's model is linear, one can derive:

$$\begin{aligned} score(h_{good}, r) > score(h_{bad}, r) \quad &\Leftrightarrow \quad \boldsymbol{w} \cdot \boldsymbol{\phi}(h_{good}, r) > \boldsymbol{w} \cdot \boldsymbol{\phi}(h_{bad}, r) \\ &\Leftrightarrow \quad \boldsymbol{w} \cdot \boldsymbol{\phi}(h_{good}, r) - \boldsymbol{w} \cdot \boldsymbol{\phi}(h_{bad}, r) > 0 \\ &\Leftrightarrow \quad \boldsymbol{w} \cdot (\boldsymbol{\phi}(h_{good}, r) - \boldsymbol{\phi}(h_{bad}, r)) > 0 \end{aligned}$$

Using the last equation, a binary classification problem can be formulated. For each human pairwise comparison we have one positive training instance with the feature vector $\boldsymbol{\phi}(h_{good}, r) - \boldsymbol{\phi}(h_{bad}, r)$ and one negative training instance with the feature vector $\boldsymbol{\phi}(h_{bad}, r) - \boldsymbol{\phi}(h_{good}, r)$. A regression is then used to train the weight vector $\boldsymbol{w}$.

Please note that once the weight vector is trained, there is no need to classify a pair of hypothesis when evaluating. A score of a single evaluated hypothesis h is computed as $\boldsymbol{w} \cdot \boldsymbol{\phi}(h, r)$.

There are two properties of this metric we really like. First, except for the lists of function words, the metric is language independent and requires almost no language resources. Therefore it participated in all the language directions. Second, the weights training method is, unlike other methods, very elegant. Most of the parametrized metrics we know about either do not tune parameters at all (UPC-STOUT and UPC-IPA), tune them manually (AMBER, VERT*) or use generic optimization techniques like hill climbing which does not have to find the optimum (METEOR).

4.2.3 BLEU-NRC

The original BLEU is not very suitable for evaluating single sentences. It can easily happen that for higher values of n, no n-gram from the candidate matches the reference translation. The precision is then zero and since the BLEU score is computed as the geometric average, it is also zero. A lot of sentences therefore get zero scores and even they can be meaningful. There is also a problem with very short sentences for which the precisions of higher n values could be undefined. To compute BLEU on sentence level, one has to smooth the precision values.

Let m_n be the clipped count of n-grams occurring in both the reference and candidate translations, and let l_n be the total count of n-grams in the candidate. In the original BLEU, the precision is computed as $p_n = m_n/l_n$. When smoothing, a modified clipped count m'_n is computed and then the precision is computed as $p_n = m'_n/l_n$.

Chen and Cherry (2014) compare various smoothing methods and choose the method which correlates best with sentence-level judgments. Their best performing smoothing method is a combination of two methods. They compute first modified counts m'_n and based on them, they compute second modified counts m''_n.

The first modified counts are computed using the following algorithm:

1: $invcnt \leftarrow 1$
2: **for** n **in** 1 **in** N **do**
3: **if** $m_n = 0$ **then**
4: $invcnt \leftarrow invcnt \cdot \frac{K}{\ln(\text{len}(c))}$
5: $m'_n \leftarrow 1/invcnt$
6: **else**
7: $m'_n \leftarrow m_n$

where K is set empirically (they use $K = 5$). The assignment in line 4 means that for shorter sentences, the smoothed clipped counts are smaller. The motivation for this is the fact that the denominator in the precision computation is smaller for shorter sentences and therefore the numerator should be also smaller so that

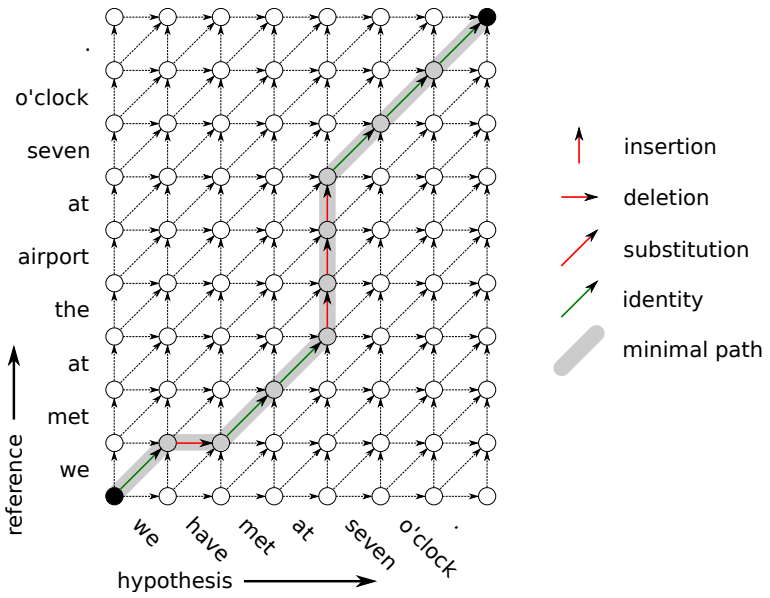


Figure 4.1: An example of WER alignment grid

precisions for shorter sentences are not inflated. However, we think that this allows to game the metric by producing very long sentences with no n-gram matched: if $\operatorname{len} c > e^K$, than the variable *invcnt* will decrease and the clipped counts m'_n will therefore increase. For each positive real number, there exists a sentence length which would ensure the metric score to be higher than the given number. We, however, do not consider this as a serious problem.

The second smoothing technique is used on top of the first method, and is inspired by the intuition that matched counts for similar values of n should be similar. This method therefore computes the modified matched counts as an average of neighbouring clipped counts. The authors define $m''_0 = m'_1 + 1$ and calculate m''_n for $n > 0$ as follows:

$$m''_n = \frac{m''_{n-1} + m'_n + m'_{n+1}}{3}$$

Finally, the n-gram precisions and the BLEU score are computed from the modified clipped counts m''_n.

For completeness we also report how the BLEU metric is smoothed in the Moses MERT implementations since it is used for computing a baseline metric sentBLEU in the sentence-level results. To both the numerator and denominator in the precision computation, a one is added:

$$p_n = \frac{m_n + 1}{l_n + 1}$$

4.2.4 WER

Word Error Rate metric is based on an edit distance. It is similar to the Levenshtein distance but the edit operations work with tokens instead of characters and the edit distance is normalized by the length of the reference r:

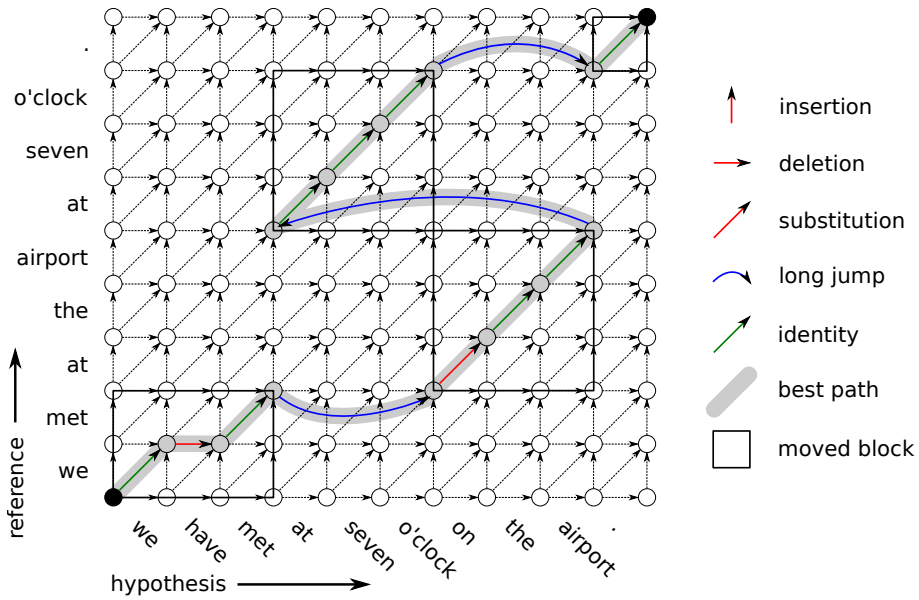


Figure 4.2: An example of CDER alignment grid with long jumps

$$WER(c,r) = \frac{\min_{e \in E(c,r)} \left(D(e) + S(e) + I(e)\right)}{|r|}$$

where $E(c,r)$ denotes a set of all sequences of edit operations which transform the candidate c to the reference r, and $D(e)$, $S(e)$, $I(e)$ denote the number of deletions, substitutions and insertions of tokens respectively in the sequence s.

The minimal sequence can be easily computed using a dynamic programing algorithm. This computation can be visualized with an alignment grid which you can see in Figure 4.1.

4.2.5 CDER

The abbreviation of this metric stands for "Cover Disjoint Error Rate" and it was developed by Leusch et al. (2006). This metric is similar to WER in that it is also an edit distance based metric. In addition to the insertions, substitutions and deletions, CDER also allows long jumps. The long jump is an operation in which we move in an alignment grid to any position in the candidate (we can get to any position in the current row of the alignment grid for a unit price). This simulates movements of blocks (longer sequences of tokens) and it is motivated by that block movements should not be penalized so harsh. The CDER is computed as follows:

$$CDER(c,r) = \frac{\min_{e \in E(c,r)} \left(D(e) + S(e) + I(e) + LJ(e)\right)}{|r|}$$

where $E(c,r)$ denotes a set of all sequences of edit operations which transform the candidate c to the reference r, and $D(e)$, $S(e)$, $I(e)$, $LJ(e)$ denote the number of deletions, substitutions, insertions and long jumps respectively in the sequence s. You can see an example of an alignment grid with long jumps in Figure 4.2.

4.2.6 DiscoTK

DiscoTK metric family was developed by Guzman et al. (2014). Its abbreviation stands for "Discourse Tree Kernel". The basic principle is that both reference and candidate translations are parsed to discourse parse trees which are then compared using convolution tree kernels. There are 5 different discourse tree representations; all of them are compared and normalized kernels scores are then averaged with uniform weights to get the DiscoTK-light score. Please see the paper for more details.

One of the weaknesses of the above discourse-based metrics is that they use unigram lexical information, which does not capture reordering. To make a more robust metric, Guzman et al. (2014) combined the above five measures with twelve more metrics implemented by Giménez and Màrquez (2010) in the Asiya toolkit.[3]. The 17 measures in total are then averaged with uniform weights to get DiscoTK-party score. The second variant of this metric, DiscoTK-party-tuned, averages the measures with tuned weights.

4.2.7 LAYERED

LAYERED was developed by Gautam and Bhattacharyya (2014). Its name comes from combining metrics on various linguistic layers:

- **Lexical Layer.** The authors decided to use ordinary BLEU from this layer.

- **Syntactic Layer.** LAYERED employs three metrics working on this layer, all of them are permutation based metrics and therefore consider only reordering of words. Permutations are constructed from source-candidate and source-reference alignments. The first two metrics are designed by Birch and Osborne (2011).

  The first metric is Hamming Score (HAMMING), which is a normalized Hamming distance defined as the number of disagreements between two permutations: $|\{i \in \{1 \dots n\} \mid \pi(i) \neq \sigma(i)\}|$.

  The second metric is Kendall's τ distance (KTD), which measures the minimum number of transpositions of two adjacent symbols necessary to transform one permutation into another.

  The last metric is Spearman rank correlation coefficient (SPEARMAN), which measures how much the permutation between the candidate and the reference is monotone. (We also use Spearman rank correlation coefficient for the metrics evaluation. You can find the exact definition later in this chapter).

- **Semantic Layer.** There are two metrics working on this layer. Both of them compute precision and recall of matched dependencies in candidate and reference parse trees and then average them to get a single score.[4] The

[3] `http://nlp.lsi.upc.edu/asiya/`

[4] A question is why the authors do not use F-measure which is usually used to combine precision and recall. They do not even use the terms precision and recall and instead define the concepts of precision and recall using a textual entailment concept.

first metric (SHALLOW) uses the Stanford dependency parser (de Marneffe et al., 2006) to generate the dependencies. The second metric (DEEP) uses the UNL dependency graph generator to construct the dependencies.

All the metrics from all the layers are then linearly combined (using tuned weights) to get a final LAYERED score:

$$\begin{aligned} LAYERED &= 0.26 \cdot BLEU + 0.13 \cdot HAMMING + 0.03 \cdot KTD \\ &+ 0.04 \cdot SPEARMAN + 0.28 \cdot SHALLOW + 0.26 \cdot DEEP \end{aligned}$$

4.2.8 Meteor

Meteor (Denkowski and Lavie, 2014) evaluates candidate translations in two phases. In the first phase, it aligns a candidate translation to a reference translation. In the second phase, a sentence-level similarity score is computed using the constructed alignment.

When aligning the candidate and the reference translations, the words (sometimes phrases) in the candidate are matched to the words (phrases) in the reference using the following matchers:

- **Exact:** Matches words if their surface forms are identical.
- **Stem:** Matches words if their stem is identical. A language appropriate stemmer is used.
- **Synonym:** Matches words if both of them are in any synonym set according to the WordNet database.
- **Paraphrase:** Matches phrases if they are listed in a language appropriate paraphrase table. These tables can be automatically extracted from ordinary phrase tables used in statistical machine translation.

Since there could be more possible alignments, the final alignment is then resolved as the largest subset of all matches which meets the following criteria:

1. Each word in each sentence has to be covered by zero or one match.
2. The total number of matched words in each sentence is maximal.
3. The number of chunks is minimal (chunk is a contiguous sequence of candidate words which is monotonically mapped to a contiguous sequence of reference words).
4. An alignment with a smaller total sum of absolute distances between the positions of aligned words is preferred.

The METEOR for an aligned sentence is then computed as follows. Let h_c and h_f be content words and function words respectively in the hypothesis (candidate translation), and similarly let r_c and r_f be content and function words in the reference. For each of the matchers m_i defined above, let $m_i(h_c)$, $m_i(h_f)$ be

the counts of matched content and function words in the hypothesis and similarly let $m_i(r_c)$, $m_i(r_f)$ be the counts of matched content and function words in the reference. Finally a weighted precision and recall are computed:

$$P = \frac{\sum_i w_i \cdot (\delta \cdot m_i(h_c) + (1-\delta) \cdot m_i(h_c))}{\delta \cdot |h_c| + (1-\delta) \cdot |h_f|}$$

$$R = \frac{\sum_i w_i \cdot (\delta \cdot m_i(r_c) + (1-\delta) \cdot m_i(r_c))}{\delta \cdot |r_c| + (1-\delta) \cdot |r_f|}$$

where $w_1 \ldots w_n$ are weights of the matchers and δ is a weight controlling the importance of content words over function words. Using the computed precision and recall, a F mean is computed:

$$F_{mean} = \frac{P \cdot R}{\alpha \cdot P + (1-\alpha) \cdot R}$$

Finally, the METEOR score is computed as follows:

$$METEOR = (1 - Pen) \cdot F_{mean}$$

where Pen is a fragmentation penalty calculated as

$$Pen = \gamma \cdot \left(\frac{ch}{m}\right)^{\beta}$$

where ch is the number of chunks (as defined above) and m is the total number of matched words. All the parameters α, β, γ, δ and $w_1 \ldots w_n$ are tuned to maximize the correlation with human judgments.

Some of the METEOR components need language specific resources. While some of the resources are limited to one or a few languages, other resources can be easily trained from bilingual or monolingual data. For example, the list of function words consists of all words with relative frequency above a certain treshold. Paraphrase tables are constructed with a simple method from a phrase table. The weights are tuned for each language direction independently, however developers of METEOR have also tuned a universal set of weights which can be used for any language direction. This is the case, for example, for directions from Russian and Hindi. The results for these directions are also very poor.

4.2.9 UPC-STOUT and UPC-IPA

Gonzàlez et al. (2014) combine various metrics implemented in the ASIYA toolkit.[5]. The set of the metrics they use contains 6 lexical based metrics (WER, PER, TER, METEOR-EXACT, etc.), 2 shallow parsing based metrics, 5 constituency parsing based metrics, 5 dependency parsing based metrics, 4 semantic roles based metrics, 1 explicit-semantic metric, 3 alignment metrics and 5 source-based metrics.

One of the interesting metrics is the explicit-semantic metric ESA. It requires large collection of Wikipedia articles W. For both the candidate and reference

[5] `http://nlp.lsi.upc.edu/asiya/`

translations, the cosine similarity (borrowed from the field of Information Retrieval) between the translation and each Wikipedia article is computed:

$$\begin{aligned} \boldsymbol{r} &= (\mathrm{sim}(r, w); \forall w \in W) \\ \boldsymbol{c} &= (\mathrm{sim}(c, w); \forall w \in W) \end{aligned}$$

The ESA score is then computed as the cosine similarity of these two vectors:

$$ESA = \mathrm{sim}(\boldsymbol{r}, \boldsymbol{c})$$

The first variant, UPC-STOUT, is defined as the uniformly weighted linear interpolation of the best performing metrics (measured on previous years metrics data). The second variant, UPC-IPA is a light version of UPC-STOUT. It considers only the metrics which require no or little language resources.

4.3 System-Level Metric Analysis

When evaluating metrics on the system level, we would like to reward metrics which predict the ordering of the systems as similar as possible to the ordering given by the human scores.

While the Spearman's ρ correlation coefficient was used traditionally as the main measure of system-level metrics' quality in the past, we have decided to use Pearson correlation coefficient as the main measure this year. We give reasons for this change in the next subsection.

Pearson correlation coefficient is a metaevaluation metric which measures how much two variables are linearly correlated. In our case, the first variable is the human score and the second variable is the metric score. We use the following formula to compute the Pearson's r for each metric and translation direction:

$$r = \frac{\sum_{i=1}^{n}(H_i - \bar{H})(M_i - \bar{M})}{\sqrt{\sum_{i=1}^{n}(H_i - \bar{H})^2}\sqrt{\sum_{i=1}^{n}(M_i - \bar{M})^2}} \tag{4.1}$$

where H is the vector of human scores of all the systems translating in the given direction and M is the vector of the corresponding scores as predicted by the given metric. $\bar{H}$ and $\bar{M}$ are their means respectively.

The Pearson's r ranges from -1 to 1. A value of 1 implies that a linear equation describes the relationship between the human scores and the metric scores and the metric score increases as the human score increases. A value of -1 also implies a linear relationship but this time the metric score decreases as the human score decreases. A value of 0 implies that there is no relationship between the human scores and the metric scores.

Since we have normalized all metrics such that better translations get a higher score, we consider metrics with values of Pearson's r closer to 1 as better.

For the sake of completeness and for the possibility to compare the results to previous years, we also report Spearman's rank correlation coefficient in the tables. This metaevaluation metric measures how much the human score and the metric score are mononotonically related. This measure does not consider the absolute differences between the values so the relationship between the two

variables is not penalized for not being linear. To compute Spearman's ρ, we must first transform the variables to ranks. If there are some equal values, we assign to them an average of corresponding ranks. Once we have the ranks, we can compute Spearman's ρ as the Pearson correlation coefficient between the rankings. Since there are no tied scores, we use the following simplified formula:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where d_i is the difference between the human rank and metric's rank for system i and n is number of the systems. The possible values of ρ range between 1 and -1. A value of 1 implies that the metric scores give the same ranking of the systems as the official human scores. A value of -1 implies that the metric scores give a reversed ranking.

The reported empirical confidence intervals of system-level correlations were obtained through bootstrap resampling of 1000 samples (confidence level of 95 %).

4.3.1 Reasons for Pearson Correlation Coefficient

In the translation task, there are often similar systems with human scores very close to each other. It can therefore easily happen that even a good metric compares two similar systems differently from humans. We believe that the penalty incurred by the metric for such a swap should somehow reflect that the systems were hard to separate.

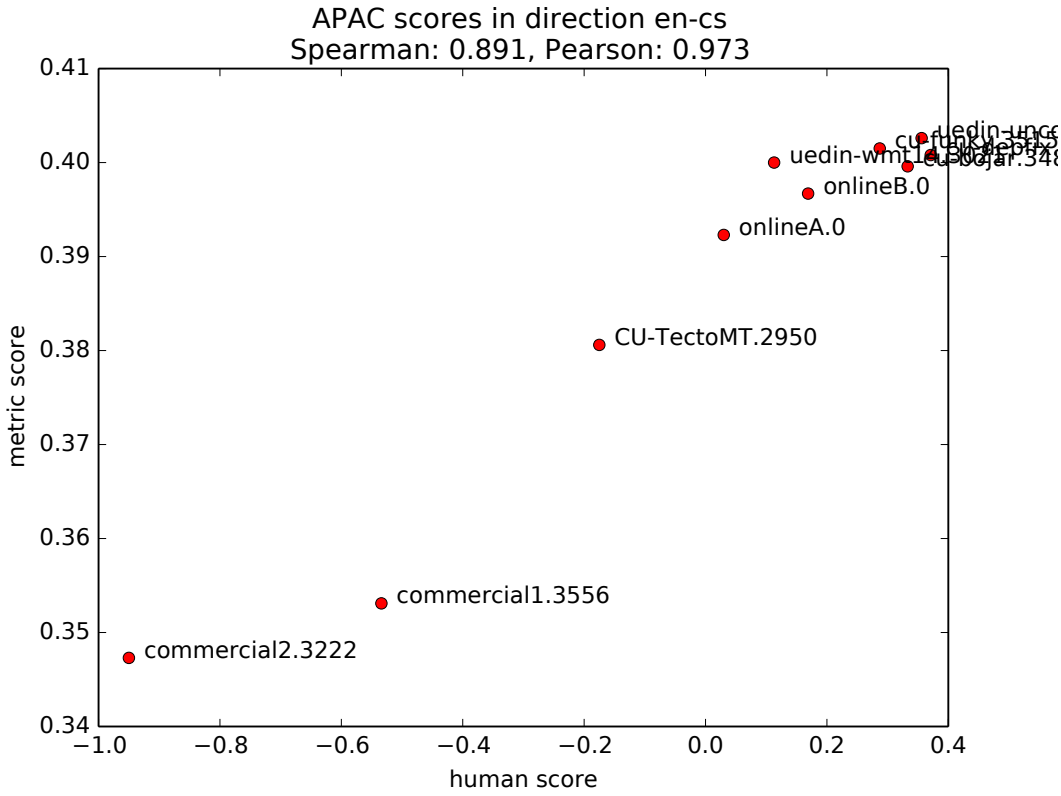


Figure 4.3: Motivation for Pearson: Metric vs. human scores scatterplot

Since the Spearman's ρ converts both human and metric scores to ranks and therefore disregards the absolute differences in the scores, it does exactly what we feel is not fair. The Pearson correlation coefficient does not suffer from this problem. We are aware of the fact that Pearson correlation coefficient also reflects whether the relation between manual and automatic scores is linear (as opposed to e.g. quadratic). We don't think this would be negatively affecting any of the metrics since overall, the systems are of a comparable quality and the metrics are likely to behave linearly in this small range of scores.

You can see an example situation in Figure 4.3. There is a cluster of very similar systems in upper right corner. In this cluster, some systems are not ranked correctly by the metric. For instance, the swap of the uedin-wmt14 and onlineB systems is not penalized so harsh in the Pearson score. However in Spearman score, it is penalized just the same as a swap of very distant systems would be penalized, for instance commercial1 and CU-TectoMT.

Moreover, the general agreement to adopt Pearson instead of Spearman's correlation coefficient was already apparent during the WMT12 workshop. This change just did not get through for WMT13.

4.3.2 Results

You can find the system-level correlations for translations into English in Table 4.2 and for translations out of English in Table 4.3. Each row in the tables contains correlations of a metric in each of the examined translation directions. The metrics are sorted by average Pearson correlation coefficient across the translation directions. The best results in each direction are in bold.

Into-English Results Analysis

As in previous years, a lot of metrics outperformed BLEU in system level correlation. In into-English directions, the metric DiscoTK-party-tuned has the highest correlation in two language directions and it is also the best correlated metric on average according to both Pearson and Spearman's coefficients. The second best correlated metric on average (according to Pearson) is LAYERED which is also the single best metric in Hindi-to-English direction. Metrics REDSys and REDSysSent are quite unstable, they win in French-to-English and Czech-to-English directions respectively but they perform very poorly in other directions.

You can see that the metrics which combine many features or many metrics on different linguistic layers perform generally much better than other metrics (the first six metrics in directions into English are of this type). Most of the lexical based metrics (NIST, TER, WER, PER) perform very poorly. However, DiscoTK-light which combines only parsing based metrics also performs very poorly. This suggest that a good metric has to combine features on various linguistic layers. The tuning of parameters is very important as you can see in the case of metrics DiscoTK-party-tuned and DiscoTK-party.

Except Meteor, none of the participants took part in the last year's metrics task. We can therefore compare current and last year's results only for Meteor and baseline metrics. Meteor, the winner last year, performs generally well

Correlation coefficient	**Pearson Correlation Coefficient**						**Spearman's**
Direction	**fr-en**	**de-en**	**hi-en**	**cs-en**	**ru-en**	**Average**	**Average**
Considered Systems	8	13	9	5	13		
DiscoTK-party-tuned	.977 ± .009	**.943 ± .020**	.956 ± .007	.975 ± .031	**.870 ± .022**	**.944 ± .018**	**.912 ± .043**
LAYERED	.973 ± .009	.893 ± .026	**.976 ± .006**	.941 ± .045	.854 ± .023	.927 ± .022	.894 ± .047
DiscoTK-party	.970 ± .010	.921 ± .024	.862 ± .015	.983 ± .025	.856 ± .023	.918 ± .019	.856 ± .046
UPC-STOUT	.968 ± .010	.915 ± .025	.898 ± .013	.948 ± .040	.837 ± .024	.913 ± .022	≀ .901 ± .045
VERTa-W	.959 ± .011	.867 ± .029	.920 ± .011	.934 ± .050	.848 ± .024	.906 ± .025	.868 ± .045
VERTa-EQ	.959 ± .011	.854 ± .031	.927 ± .010	.938 ± .048	.842 ± .024	.904 ± .025	.857 ± .046
tBLEU	.952 ± .012	.832 ± .034	.954 ± .007	.957 ± .040	.803 ± .027	.900 ± .024	.841 ± .056
BLEU-NRC	.953 ± .012	.823 ± .035	.959 ± .007	.946 ± .044	.787 ± .028	.894 ± .025	≀ .855 ± .056
BLEU	.952 ± .012	.832 ± .034	.956 ± .007	.909 ± .054	.789 ± .027	.888 ± .027	.833 ± .058
UPC-IPA	.966 ± .010	.895 ± .027	.914 ± .010	.824 ± .073	.812 ± .026	.882 ± .029	≀ .858 ± .044
CDER	.954 ± .012	.823 ± .034	.826 ± .016	.965 ± .035	.802 ± .027	.874 ± .025	.807 ± .050
APAC	.963 ± .010	.817 ± .034	.790 ± .016	.982 ± .026	.816 ± .026	.874 ± .022	.807 ± .049
REDSys	**.981 ± .008**	.898 ± .026	.676 ± .022	.989 ± .021	.814 ± .026	.872 ± .021	.786 ± .047
REDSysSent	.980 ± .008	.910 ± .024	.644 ± .023	**.993 ± .018**	.807 ± .027	.867 ± .020	.771 ± .043
NIST	.955 ± .011	.811 ± .035	.784 ± .016	.983 ± .025	.800 ± .027	.867 ± .023	≀ .824 ± .055
DiscoTK-light	.965 ± .011	.935 ± .022	.557 ± .025	.954 ± .038	.791 ± .027	.840 ± .024	.774 ± .046
Meteor	.975 ± .009	.927 ± .022	.457 ± .027	.980 ± .029	.805 ± .026	.829 ± .023	≀ .788 ± .046
TER	.952 ± .012	.775 ± .038	.618 ± .021	.976 ± .031	.809 ± .027	.826 ± .026	.746 ± .057
WER	.952 ± .012	.762 ± .038	.610 ± .021	.974 ± .033	.809 ± .027	.821 ± .026	.736 ± .058
AMBER	.948 ± .012	.910 ± .026	.506 ± .026	.744 ± .095	.797 ± .027	.781 ± .037	.728 ± .051
PER	.946 ± .013	.867 ± .031	.411 ± .025	.883 ± .063	.799 ± .028	.781 ± .032	.698 ± .047
ELEXR	.971 ± .009	.857 ± .031	.535 ± .026	.945 ± .044	−.404 ± .045	.581 ± .031	.652 ± .046

Table 4.2: System-level correlations of automatic evaluation metrics and the official WMT human scores when translating into English. The symbol "≀" indicates where the Spearman's ρ average is out of sequence compared to the main Pearson average.

Correlation coefficient	**Pearson Correlation Coefficient**						**Spearman's**
Direction	**en-fr**	**en-hi**	**en-cs**	**en-ru**	**Average**	**en-de**	**Average**
Considered Systems	13	12	10	9		18	(excl. en-de)
NIST	.941 ± .022	.981 ± .006	.985 ± .006	.927 ± .012	**.959 ± .012**	.200 ± .046	**.850 ± .030**
CDER	.949 ± .020	.949 ± .010	.982 ± .006	.938 ± .011	.955 ± .012	.278 ± .045	.840 ± .036
AMBER	.928 ± .023	**.990 ± .004**	.972 ± .008	.926 ± .012	.954 ± .012	.241 ± .045	.817 ± .041
METEOR	.941 ± .021	.975 ± .007	.976 ± .007	.923 ± .013	.954 ± .012	.263 ± .045	.806 ± .039
BLEU	.937 ± .022	.973 ± .007	.976 ± .007	.915 ± .013	.950 ± .012	.216 ± .046	≀ .809 ± .036
PER	.936 ± .023	.931 ± .011	**.988 ± .005**	**.941 ± .011**	.949 ± .013	.190 ± .047	≀ .823 ± .037
APAC	.950 ± .020	.940 ± .011	.973 ± .008	.929 ± .012	.948 ± .013	.346 ± .044	.799 ± .041
TBLEU	.932 ± .023	.968 ± .008	.973 ± .008	.912 ± .013	.946 ± .013	.239 ± .046	≀ .805 ± .039
BLEU-NRC	.933 ± .022	.971 ± .007	.974 ± .008	.901 ± .014	.945 ± .013	.205 ± .046	≀ .809 ± .039
ELEXR	.885 ± .029	.962 ± .009	.979 ± .007	.938 ± .011	.941 ± .014	.260 ± .044	.768 ± .036
TER	.954 ± .019	.829 ± .017	.978 ± .007	.931 ± .012	.923 ± .014	.324 ± .045	.745 ± .035
WER	**.960 ± .018**	.516 ± .026	.976 ± .007	.932 ± .011	.846 ± .016	**.357 ± .045**	.696 ± .037
PARMESAN	n/a	n/a	.962 ± .009	n/a	.962 ± .009	n/a	.915 ± .048
UPC-IPA	.940 ± .021	n/a	.969 ± .008	.921 ± .013	.943 ± .014	.285 ± .045	.785 ± .050
REDSYSSENT	.941 ± .021	n/a	n/a	n/a	.941 ± .021	.208 ± .045	≀ .962 ± .038
REDSYS	.940 ± .021	n/a	n/a	n/a	.940 ± .021	.208 ± .045	.962 ± .038
UPC-STOUT	.940 ± .021	n/a	.938 ± .011	.919 ± .013	.933 ± .015	.301 ± .044	.713 ± .040

Table 4.3: System-level correlations of automatic evaluation metrics and the official WMT human scores when translating out of English. The symbol "≀" indicates where the Spearman's ρ average is out of sequence compared to the main Pearson average.

in some directions but it suffers horribly when evaluating translations from non-Latin script (Russian and especially Hindi). For the baseline metrics the results are quite similar across the years. In both years BLEU performs best among baseline metrics, closely followed by CDER. NIST is in the middle of the list in both years. The remaining baseline metrics TER, WER and PER perform significantly worse.

Out-of-English Results Analysis

The results into German are markedly lower and have broader confidence intervals than the results in other directions. This could be explained by a very high number (18) of participating systems of similar quality. Both human judgements and automatic metrics are negatively affected by these circumstances. To preserve the reliability of overall metrics' performance across languages, we decided to exclude English-to-German direction from the average Pearson and Spearman's correlation coefficients.

In other out-of-English directions, the best correlated metric on average according to Pearson coefficient is NIST, even though it does not win in any single direction. CDER is the second best according to Pearson and the best metric according to Spearman's. Again it does not win in any single direction. The metrics PER and WER are quite unstable. Each of them wins in two directions but performs very badly in others.

Compared to the last year results, the order of metrics participating in both years is quite similar: NIST and CDER performed very well both years, followed by BLEU. The metrics TER and WER are again at the end of the list. An interesting change is that PER performs much better this year.

4.4 Sentence-Level Metric Analysis

We measure the quality of metrics' sentence-level scores using Kendall's τ rank correlation coefficient. In this type of evaluation, the metric is expected to predict the result of the manual pairwise comparison of two systems. Note that the golden truth is obtained from a compact annotation of five systems at once, while an experiment with text-to-speech evaluation techniques by Vazquez-Alvarez and Huckvale (2002) suggests that a genuine pairwise comparison is likely to lead to more stable results.

In the past, slightly different variations of Kendall's τ computation were used in the Metrics Tasks. Also some of the participants have noticed a problem with ties in the WMT13 method. Therefore, we discuss several possible variants in detail in this thesis.

4.4.1 Notation for Kendall's τ Computation

The basic formula for Kendall's τ is:

$$\tau = \frac{|Concordant| - |Discordant|}{|Concordant| + |Discordant|} \tag{4.2}$$

where *Concordant* is the set of all human comparisons for which a given metric suggests the same order and *Discordant* is the set of all human comparisons for which a given metric disagrees. For an example of Kendall's τ computation, let us have the following comparisons:

Human	Metric
$A < B$	$A < B$
$C > A$	$C > A$
$C > B$	$C < B$

There are two concordant comparisons and one discordant:

$$\tau = \frac{2-1}{2+1} = \frac{1}{3}$$

In the original Kendall's τ, comparisons with human or metric ties are considered neither concordant nor discordant. However in the previous Metrics Tasks (Callison-Burch et al. (2012) and earlier), comparisons with human ties were considered as discordant. To easily specify which pairs are counted as concordant and which as discordant, we have developed the following tabular notation. This is, for example, the WMT12 method:

		Metric		
WMT12		$<$	$=$	$>$
Human	$<$	1	-1	-1
	$=$	X	X	X
	$>$	-1	-1	1

Given such a matrix $C_{h,m}$ where $h, m \in \{<, =, >\}$[6] and a metric we compute the Kendall's τ the following way: We insert each extracted human pairwise comparison into exactly one of the nine sets $S_{h,m}$ according to human and metric ranks. For example the set $S_{<,>}$ contains all comparisons where the left-hand system was ranked better than right-hand system by humans and it was ranked the other way round by the metric in question. To compute the numerator of Kendall's τ, we take the coefficients from the matrix $C_{h,m}$, use them to multiply the sizes of the corresponding sets $S_{h,m}$ and then sum them up. We do not include sets for which the value of $C_{h,m}$ is X. To compute the denominator of Kendall's τ, we simply sum the sizes of all the sets $S_{h,m}$ except those where $C_{h,m} = \text{X}$. To define it formally:

$$\tau = \frac{\sum\limits_{\substack{h,m \in \{<,=,>\} \\ C_{h,m} \neq \text{X}}} C_{h,m}|S_{h,m}|}{\sum\limits_{\substack{h,m \in \{<,=,>\} \\ C_{h,m} \neq \text{X}}} |S_{h,m}|}$$

For an example of WMT12 computation, let us have the following comparisons:

[6]Here the relation $<$ always means "is better than" even for metrics where the better system receives a higher score.

Human	Metric
$A < B$	$A < B$
$A < B$	$A < B$
$A > B$	$A = B$
$A = B$	$A > B$

The Kendall's τ score is then computed in the following way:

$$\tau = \frac{2-1}{2+1} = \frac{1}{3}$$

4.4.2 Discussion on Kendall's τ Computation

In 2013, we thought that metric ties should not be penalized and we decided to excluded them like the human ties. We will denote this method as WMT13:

		Metric		
WMT13		$<$	$=$	$>$
Human	$<$	1	X	-1
	$=$	X	X	X
	$>$	-1	X	1

It turned out, however, that it was not a good idea: metrics could game the scoring by avoiding hard cases and assigning lots of ties. A natural solution is to count the metrics ties also in the denominator to avoid the problem. We will denote this variant as WMT14:

		Metric		
WMT14		$<$	$=$	$>$
Human	$<$	1	0	-1
	$=$	X	X	X
	$>$	-1	0	1

The WMT14 variant does not allow for gaming the scoring like the WMT13 variant does. Compared to WMT12 method, WMT14 does not penalize ties.

We also considered getting human ties involved. The most natural variant would be the following variant denoted as HTIES:

		Metric		
HTIES		$<$	$=$	$>$
Human	$<$	1	0	-1
	$=$	0	1	0
	$>$	-1	0	1

Unfortunately this method allows for gaming the scoring as well. The least risky choice for metrics in hard cases would be to assign a tie because it cannot worsen the Kendall's τ and there is quite a high chance that the human rank is also a tie. Metrics could be therefore tuned to predict ties often but such metrics are not very useful. For example, the simplistic metric which assigns the same score to all candidates (and therefore all pairs would be tied by the metric) would get the score equal to the proportion of ties in all human comparisons. It would become

one of the best performing metrics in WMT13 even though it is not informative at all.

We have decided to use WMT14 variant as the main evaluation measure this year, however, we are also reporting average scores computed by other variants.

4.4.3 Kendall's τ Results

The final Kendall's τ results are shown in Table 4.4 for directions into English and in Table 4.5 for directions out of English. Each row in the tables contains correlations of a metric in given directions. The metrics are sorted by the average correlation across translation directions. The highest correlation in each column is in bold. Table 4.6 and Table 4.7 contain average Kendall's τ computed by other variants including the variant WMT13 used last year. Metrics which did not compute scores in all directions are at the bottom of the tables. The possible values of τ range between -1 (a metric always predicted a different order than humans did) and 1 (a metric always predicted the same order as humans did). Metrics with a higher τ are better.

We also computed empirical confidence intervals of Kendall's τ using bootstrap resampling. We varied the "golden truth" by sampling from human judgments. We have generated 1000 new sets and report the average of the upper and lower 2.5 % empirical bound, which corresponds to the 95 % confidence interval.

Analysis

In directions into English (Table 4.4), the strongest correlated sentence-level metric on average is DiscoTK-party-tuned followed by BEER. Unlike the system-level correlation, the results are much more stable here. DiscoTK-party-tuned has the highest correlation in 4 of 5 language directions. Generally, the ranking of metrics is almost the same in each direction.

The only two metrics which also participated in last year metrics task are Meteor and sentBLEU. In both years, Meteor performed quite well unlike sentBLEU which was outperformed by most of the metrics.

The metric DiscoTK-light-kool is worth mentioning. It is deliberately designed to assign the same score for all systems for most of the sentences. It obtained scores very close to zero (i.e. totally uninformative) in the WMT14 variant. In WMT13 though, it reached the highest score.

In directions out of English (Table 4.5), the metric with highest correlation on average across all directions is Beer, followed by Meteor.

The trends in the sentence level are quite similar to the trends in the system level. The metrics which combine features on various linguistic layers are much better than metrics which do not. The metrics which tune their weights (DiscoTK-party-tuned, BEER, RED* and Meteor) are significantly better better than metrics which do not (UPC-*, VERTa-*). An interesting contrast is between the DiscoTK-party-tuned and DiscoTK-party metrics, which differ only in the tuning. It shows that the tuning is very important. Comparing DiscoTK-party to DiscoTK-light shows that discourse tree based metrics are not very useful in MT evaluation.

Direction	**fr-en**	**de-en**	**hi-en**	**cs-en**	**ru-en**	**Avg**
Extracted-pairs	26090	25260	20900	21130	34460	
DiscoTK-party-tuned	**.433 ± .012**	**.380 ± .013**	.434 ± .013	**.328 ± .015**	**.355 ± .011**	**.386 ± .013**
BEER	.417 ± .013	.337 ± .014	**.438 ± .013**	.284 ± .016	.333 ± .011	.362 ± .013
REDcombSent	.406 ± .012	.338 ± .014	.417 ± .013	.284 ± .015	.336 ± .011	.356 ± .013
REDcombSysSent	.408 ± .012	.338 ± .014	.416 ± .013	.282 ± .014	.336 ± .011	.356 ± .013
Meteor	.406 ± .012	.334 ± .014	.420 ± .013	.282 ± .015	.329 ± .010	.354 ± .013
REDSysSent	.404 ± .012	.338 ± .014	.386 ± .014	.283 ± .015	.321 ± .010	.346 ± .013
REDSent	.403 ± .012	.336 ± .014	.383 ± .014	.283 ± .015	.323 ± .011	.345 ± .013
UPC-IPA	.412 ± .012	.340 ± .014	.368 ± .014	.274 ± .015	.316 ± .011	.342 ± .013
UPC-STOUT	.403 ± .012	.345 ± .014	.352 ± .014	.275 ± .015	.317 ± .011	.338 ± .013
VERTa-W	.399 ± .013	.321 ± .015	.386 ± .014	.263 ± .015	.315 ± .011	.337 ± .014
VERTa-EQ	.407 ± .013	.315 ± .014	.384 ± .013	.263 ± .015	.312 ± .011	.336 ± .013
DiscoTK-party	.395 ± .013	.334 ± .014	.362 ± .013	.264 ± .016	.305 ± .011	.332 ± .013
AMBER	.367 ± .013	.313 ± .014	.362 ± .013	.246 ± .016	.294 ± .011	.316 ± .013
BLEU-NRC	.382 ± .013	.272 ± .014	.322 ± .014	.226 ± .016	.269 ± .011	.294 ± .013
sentBLEU	.378 ± .013	.271 ± .014	.300 ± .013	.213 ± .016	.263 ± .011	.285 ± .013
APAC	.364 ± .012	.271 ± .014	.288 ± .014	.198 ± .016	.276 ± .011	.279 ± .013
DiscoTK-light	.311 ± .014	.224 ± .015	.238 ± .013	.187 ± .016	.209 ± .011	.234 ± .014
DiscoTK-light-kool	.005 ± .001	.001 ± .000	.000 ± .000	.002 ± .001	.001 ± .000	.002 ± .001

Table 4.4: Sentence-level Kendall's τ correlations of automatic evaluation metrics and the official WMT human judgements when translating into English.

Direction	**en-fr**	**en-de**	**en-hi**	**en-cs**	**en-ru**	**Avg**
Extracted-pairs	33350	54660	28120	55900	28960	
BEER	.292 ± .012	**.268 ± .009**	.250 ± .013	**.344 ± .009**	**.440 ± .013**	**.319 ± .011**
METEOR	.280 ± .012	.238 ± .009	.264 ± .012	.318 ± .009	.427 ± .012	.306 ± .011
AMBER	.264 ± .012	.227 ± .009	**.286 ± .012**	.302 ± .009	.397 ± .013	.295 ± .011
BLEU-NRC	.261 ± .012	.202 ± .009	.234 ± .013	.297 ± .009	.391 ± .012	.277 ± .011
APAC	.253 ± .012	.210 ± .008	.203 ± .012	.292 ± .009	.388 ± .013	.269 ± .011
SENTBLEU	.256 ± .012	.191 ± .009	.227 ± .012	.290 ± .009	.381 ± .013	.269 ± .011
UPC-STOUT	.279 ± .011	.234 ± .008	n/a	.282 ± .009	.425 ± .013	.305 ± .011
UPC-IPA	.264 ± .012	.227 ± .009	n/a	.298 ± .009	.426 ± .013	.304 ± .011
REDSENT	**.293 ± .012**	.242 ± .009	n/a	n/a	n/a	.267 ± .010
REDCOMBSYSSENT	.291 ± .012	.244 ± .009	n/a	n/a	n/a	.267 ± .010
REDCOMBSENT	.290 ± .012	.242 ± .009	n/a	n/a	n/a	.266 ± .010
REDSYSSENT	.290 ± .012	.239 ± .008	n/a	n/a	n/a	.264 ± .010

Table 4.5: Sentence-level Kendall's τ correlations of automatic evaluation metrics and the official WMT human judgements when translating out of English.

Metric	WMT14	WMT12	WMT13	HTIES
DiscoTK-party-tuned	**.386 ± .013**	**.386 ± .013**	.386 ± .013	.306 ± .010
BEER	.362 ± .013	.358 ± .013	.363 ± .013	≀ **.318 ± .011**
REDcombSent	.356 ± .013	.346 ± .013	.360 ± .013	.317 ± .011
REDcombSysSent	.356 ± .013	.346 ± .013	.359 ± .013	.316 ± .010
Meteor	.354 ± .013	.341 ± .013	.359 ± .013	≀ .317 ± .010
REDSysSent	.346 ± .013	.335 ± .013	.350 ± .013	.309 ± .010
REDSent	.345 ± .013	.334 ± .013	.349 ± .013	.308 ± .010
UPC-IPA	.342 ± .013	≀ .340 ± .014	.343 ± .014	.300 ± .011
UPC-STOUT	.338 ± .013	.336 ± .013	.339 ± .013	.294 ± .011
VERTa-W	.337 ± .014	.320 ± .014	≀ .342 ± .014	≀ .304 ± .011
VERTa-EQ	.336 ± .013	≀ .323 ± .013	.341 ± .013	.302 ± .011
DiscoTK-party	.332 ± .013	≀ .332 ± .013	.332 ± .013	.263 ± .011
AMBER	.316 ± .013	.302 ± .013	.321 ± .014	≀ .286 ± .011
BLEU-NRC	.294 ± .013	.267 ± .014	.303 ± .014	.271 ± .011
sentBLEU	.285 ± .013	.258 ± .014	.293 ± .014	.264 ± .011
APAC	.279 ± .013	.243 ± .014	.290 ± .014	.261 ± .011
DiscoTK-light	.234 ± .014	.234 ± .014	.234 ± .014	.184 ± .011
DiscoTK-light-kool	.002 ± .001	−.996 ± .001	≀ **.676 ± .256**	≀ .211 ± .005

Table 4.6: This table contains average sentence-level Kendall's τ computed by other variants in directions into English. The average correlation of WMT14 variant is repeated in the first column. The symbol "≀" indicates where the averages of other variants are out of sequence compared to the WMT14 variant.

Metric	WMT14	WMT12	WMT13	HTIES
BEER	**.319 ± .011**	**.314 ± .011**	**.320 ± .011**	.272 ± .009
Meteor	.306 ± .011	.283 ± .011	.313 ± .011	≀ **.273 ± .008**
AMBER	.295 ± .011	.269 ± .011	.303 ± .011	.266 ± .009
BLEU-NRC	.277 ± .011	.235 ± .011	.289 ± .011	.256 ± .009
APAC	.269 ± .011	.217 ± .011	.285 ± .011	.252 ± .008
sentBLEU	.269 ± .011	≀ .232 ± .011	.280 ± .011	.246 ± .009
UPC-STOUT	.305 ± .011	.300 ± .010	.306 ± .011	.256 ± .008
UPC-IPA	.304 ± .011	.292 ± .011	≀ .308 ± .011	≀ .259 ± .008
REDSent	.267 ± .010	.246 ± .010	.273 ± .011	.257 ± .008
REDcombSysSent	.267 ± .010	≀ .249 ± .010	.272 ± .010	.256 ± .008
REDcombSent	.266 ± .010	.248 ± .010	.271 ± .011	.256 ± .008
REDSysSent	.264 ± .010	.235 ± .010	≀ .273 ± .010	≀ .257 ± .008

Table 4.7: This table contains average sentence-level Kendall's τ computed by other variants in directions out of English. The average correlation of WMT14 variant is repeated in the first column. The symbol "≀" indicates where the averages of other variants are out of sequence compared to the WMT14 variant.

Chapter 5

Related Work

This chapter surveys related work on the boundary of automatic and manual evaluation. At the end, we also report related work to the automatic metric evaluation.

5.1 Feasibility of Human Evaluation in MERT

The work of Zaidan and Callison-Burch (2009) was the main inspiration for our **SegRanks** method. They develop a new metric called RYPT to use it primarily in the MERT method. This metric takes human judgments into account, but requires manual labour only at the beginning to build a database that can be reused later to evaluate unseen candidates. The core idea is to extract segments from source parsed tree and then using an alignment produced by a decoder project these source side segments to segments in n-best list candidates. The target side segments are then evaluated by humans and stored to a database, which is used later when scoring n-best list. The authors claim that this evaluation is done only once before the first iteration of MERT, however they do not specify how new, unseen segments from n-best lists produced in later MERT iterations would be evaluated.

Despite the RYPT metric is designed to be used in the MERT method, Zaidan and Callison-Burch (2009) actually have not done any experiment with MERT for a lack of resources. Only a pilot study is reported in the paper. They tried the method only on a relatively small sample of sentences from n-best list produced with already tuned weights. The reason why we could afford to do the experiment with MERT with comparable resources is that we do not extract candidate segments from the whole n-best list.

From their paper, we adopted mainly the short segment candidate extraction process. The annotation process, scoring the candidates and conducted experiments are, however, quite different to our work. The main difference is that they extract the short segments for evaluation directly from an n-best list, while we extract them from the evaluated systems' translations and hope that they will cover also the n-best list. The difference in the annotating short segments is that annotators in the paper of Zaidan and Callison-Burch (2009) do not rank candidate segments relatively to each other, but they use absolute labels `YES`, `NO` and `NOT SURE` to judge whether a candidate segment is an acceptable translation. The next difference is in the scoring, while we compute **Ratio of wins (ignoring**

ties), they compute the proportion of short segments labeled `YES`. We decided to do these changes in our method to have the annotation more similar to the official WMT human evaluation.

5.2 Extrapolating Score from Similar Sentences

Nießen et al. (2000) have developed a tool for manual evaluation. Annotators select for each evaluated sentence a rank from an absolute point scale. Each evaluated sentence is then stored to a database with its rank. The authors use their tool for everyday evaluating of new variants of their system which often translate differently only a small percentage of a development test set[1]. Identically translated sentences are therefore not evaluated again and are automatically assigned a rank from the database. Only the new translations are evaluated by humans and stored into the database with their rank.

When the database is large enough, there is an option to evaluate new translations automatically by extrapolating ranks of candidates from the database. For an evaluated candidate sentence, the rank of the closest sentence by edit distance is assigned. If there is more sentences in the database with equal edit distance, the average rank is used. This is similar to the matching the closest segment which we do in Section 3.2.2.

The authors present a few statistics related to their database, such as an average of absolute differences between the real score and the extrapolated score computed using the method similar to our **leave-one-out** trick. However, they do not show how good the extrapolated scores are and if they also do not suffer from overestimation. One of their collected database contains 42.9 candidate translations per a source sentence on average. This is much higher than in our database (the maximum number of candidates for one source segment is 10), so we could speculate that their space of candidates is much more dense and therefore may not be so affected by the overestimation.

5.3 Scratching the Surface of Possible Translations

The work by Bojar et al. (2013) is quite different to the previous two works. Their longterm goal is to improve automatic evaluation by significantly enlarging the set of reference translations. Any metric that can compare a candidate to multiple references can be then used for evaluation. The idea is that if we have a very large set of references, then there will be higher chance that either the evaluated candidate will be in the reference set, or there will be a reference very similar to the candidate. In both of the cases, an automatic metric will predict the quality much more accurately.

To systematically construct the very large set of reference translations, Bojar et al. (2013) propose compact representation in which annotators create many translations of smaller units, called bubbles, and specify conditions under which

[1]This paper was actually published before the MERT method was introduced. When it is used, it changes most of the translations.

the translated bubbles can combine together to create the whole reference translation. All possible combinations are generated and added to the set of reference translations. A single annotator could for a given source sentence produce hundreds of thousands reference translations using this method in two hours of work.

The authors show that BLEU computed on a test set of 50 sentences with all the produced references achieves better correlation with human judgments than BLEU computed on a test set of 3003 sentences with single reference translation. It would be interesting to experiment with many references when tuning a system using MERT method.

5.4 Metaevaluation

Metrics Shared Task (also sometimes called Evaluation Task) is held annually within Workshop on Statistical Machine Translation starting by Callison-Burch et al. (2008). Until the year 2012, the tasks' results used to be reported in the main overview paper. In the years 2013 and 2014, it was organized by Macháček and Bojar (2013) and Macháček and Bojar (2014) and reported in dedicated papers.

Besides the shared task within WMT, there were also MetricsMATR evaluation campaign in years 2008[2] and 2010[3].

[2] `http://www.itl.nist.gov/iad/mig/tests/metricsmatr/2008/`

[3] The task was joint with the WMT task this year, `http://www.nist.gov/itl/iad/mig/metricsmatr10.cfm`

Chapter 6

Conclusion

6.1 Manual and Semiautomatic Evaluation

In this thesis, we proposed a new method for manual evaluation, called **SegRanks**, in which annotators rank short segments (up to six words) of a translated sentence relatively to each other. The ranking of short segments is easier for annotators, since they do not have to read and remember whole sentences at once. The most promising benefit of this method is that short segments are often translated identically. We can take advantage of this in two ways: First, annotators are shown identical segments only once so that they do not have to rank them multiple times. Second, the evaluated segments can be stored together with their ranks in a database, which can be used later to automatically evaluate unseen sentences or to tune a system's parameters. We also discussed disadvantages of this method. The most severe ones are that the extracted segments do not always cover the whole sentence and that the segments are evaluated without their sentence context.

We developed an easy-to-use and modern annotation interface and conducted a manual evaluation experiment using the proposed method. We evaluated the systems which participated in the English-Czech direction in WMT Translation Task. The measured inter- and intra-annotator κ scores (the normalized agreements) are higher than the corresponding values in the WMT manual evaluation, which means that our evaluation method is more robust.

To get a final score for each system's translation, we computed how often the segments of the system were ranked better than other segments (in the context of the pairwise comparisons). The results of the evaluated systems are quite similar to the results obtained by the official WMT judgments. However, our method is not able to correctly distinguish some systems with very similar quality. We manually analyzed the sentences which were ranked high in the short segment judgments but ranked low in the official WMT judgments to explain the difference. In most of these sentences, there was a badly translated part which was, however, not covered with the evaluated short segments. The uncovered parts often contained verbs which have a significant impact on the translation quality.

To explore the possibility of reusing the collected database to evaluate unseen translations, we have performed several experiments. In the first one, we evaluated unseen translations using only the ranks of the segments which were in the database. This, however, did not work as expected, because the obtained scores

of unseen systems were significantly overestimated. During the manual analysis, we identified that the evaluated systems are more likely to agree on better translations than on worse translations.

To avoid evaluating unseen translations only on a not-representative subset of short segments, we proposed another method. In this method, we evaluated unseen translations on all the extracted segments. To approximate the rank of an unseen segment, we took the rank of the closest segment by edit distance. This method, however, didn't work as well. The approximated rank was predicted correctly using the closest segment only in 19.7 % cases. In 51.9 % of the cases, the predicted rank was better than the original rank. The final scores of the unseen systems were thus overestimated again.

In another experiment, we extracted the best ranked segments from the collected database and considered them as good translations. We used them as additional reference translations for BLEU. However, it did not perform better than original BLEU with single reference.

In the last experiment with the collected database, we tried to use the database to tune a machine translation system using the MERT method. We proposed several variants of **SegRanks** based metrics adapted for the MERT tuning. The tuned systems were evaluated by humans against the baseline system tuned by BLEU. We were able to improve the tuning of the system using a technique which considered unseen segments as bad and therefore pushed the system to produce known and already evaluated segments.

Although most of the proposed methods exploiting the collected database did not work, we tried to identify and analyze root causes of the failures. The main cause seems to be the fact that errors in machine translation are unique and that segments produced by more than one system are likely to be of better quality. This is also related with the fact that translations are more likely to be closer to better translations than to equally good or worse translations. Maybe, if we had a more dense database (many more than 10 evaluated systems), these phenomena would not influence the results so adversely.

6.2 Automatic Evaluation

In the second part of this thesis, we summarized the results of the WMT14 Metrics Shared Task which we ran. The shared task assesses the quality of various automatic machine translation metrics. Judgements collected in the WMT14 human evaluation served as the golden truth and we checked how well the metrics predicted the judgements at the level of individual sentences (sentence-level task) as well as at the level of the whole test set (system-level task).

In the system-level task, we discussed differences between Spearman's rank correlation coefficient and Pearson correlation coefficient and decided to choose Pearson coefficient instead of Spearman's rank coefficient as being fairer. In the sentence-level task, we introduced a new notation which exactly specifies the details on Kendall's τ computation. We also discussed several variants of Kendall's τ used in the past and proposed and used a new variant which does not suffer from shortcomings of other variants.

We observed two general trends: First, the metrics which employ features on various linguistic layers are better than other metrics. Second, the metrics which

tune their parameters or weights are also better than other metrics.

We have implemented scripts for metrics evaluation and published them in a package together with human judgments[1]. Anyone can therefore reproduce the results and use it to evaluate his or her metric on the WMT14 data.

6.3 Future Work

The **SegRanks** method could be improved in several ways to hopefully avoid its shortcomings. Short segments should be extracted in a way that they cover either all words in a sentence or the most important parts, for example predicates. To avoid data sparseness, we should evaluate more systems or extract segments from the n-best list directly in the case of tuning. An application of machine learning techniques to predict quality of unseen short segments should be also examined.

For the evaluation of automatic metrics, the basic principle of the metrics task will very probably not change. We will also try not to change any metaevaluation measures again (like we did in WMT13 and WMT14), so that participants can rely on fixed rules and that results can be more comparable across years. We expect that new metrics will emerge to be examined in the task. There are two possible enhancements of the task we think of. First, automatic metrics should be also evaluated in terms of tuning a system's parameters. There was already Tunable Metrics Task organized at WMT11 and we should consider it again. Second, the confidence intervals are now computed by bootstrapping only the human judgments. It would be better if we could bootstrap new test sets and compute a metric score for each sampled test set. This would, however, require participants to compute their metric score for each of the sampled test sets.

[1] `http://www.statmt.org/wmt14/wmt14-metrics-task.tar.gz`

Bibliography

Barančíková, P. (2014, June). Parmesan: Improving Meteor by More Fine-grained Paraphrasing. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Birch, A. and M. Osborne (2011). Reordering metrics for MT. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 1027–1035. Association for Computational Linguistics.

Bojar, O. (2012). *Čeština a strojový překlad : strojový překlad našincum, našinci strojovému překladu*. Studies in Computational and Theoretical Linguistics. Praha, Česká republika: Ústav formální a aplikované lingvistiky.

Bojar, O., C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna (2014). Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Bojar, O., M. Ercegovčević, M. Popel, and O. F. Zaidan (2011). A Grain of Salt for the WMT Manual Evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, Stroudsburg, PA, USA, pp. 1–11. Association for Computational Linguistics.

Bojar, O., M. Macháček, A. Tamchyna, and D. Zeman (2013). Scratching the surface of possible translations. In *Text, Speech, and Dialogue*, pp. 465–474. Springer.

Bojar, O., Z. Žabokrtský, O. Dušek, P. Galuščáková, M. Majliš, D. Mareček, J. Maršík, M. Novák, M. Popel, and A. Tamchyna (2012, May). The Joy of Parallelism with CzEng 1.0. In *Proceedings of LREC2012*, Istanbul, Turkey. ELRA: European Language Resources Association. In print.

Callison-Burch, C., C. Fordyce, P. Koehn, C. Monz, and J. Schroeder (2007, June). (Meta-) Evaluation of Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, Prague, Czech Republic, pp. 136–158. Association for Computational Linguistics.

Callison-Burch, C., C. Fordyce, P. Koehn, C. Monz, and J. Schroeder (2008, June). Further Meta-Evaluation of Machine Translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Columbus, Ohio, pp. 70–106. Association for Computational Linguistics.

Callison-Burch, C., P. Koehn, C. Monz, M. Post, R. Soricut, and L. Specia (2012, June). Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, Montréal, Canada, pp. 10–51. Association for Computational Linguistics.

Callison-Burch, C., P. Koehn, C. Monz, and O. Zaidan (2011, July). Findings of the 2011 Workshop on Statistical Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, Scotland, pp. 22–64. Association for Computational Linguistics.

Chen, B. and C. Cherry (2014, June). A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Chen, B. and R. Kuhn (2011). AMBER: A Modified BLEU, Enhanced Ranking Metric. In *Proceedings of the 6th Workshop on Statistical Machine Translation*, pp. 71–77.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement 20*(1), 37.

Comelles, E. and J. Atserias (2014, June). VERTa participation in the WMT14 Metrics Task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

de Marneffe, M.-C., B. MacCartney, and C. D. Manning (2006). Generating typed dependency parses from phrase structure parses. In *LREC*, pp. 449–454.

Denkowski, M. and A. Lavie (2014, June). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, HLT '02, San Francisco, CA, USA, pp. 138–145. Morgan Kaufmann Publishers Inc.

Durrani, N., B. Haddow, P. Koehn, and K. Heafield (2014, June). Edinburgh's Phrase-based Machine Translation Systems for WMT-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA, pp. 97–104. Association for Computational Linguistics.

Echizen'ya, H. (2014, June). Application of Prize based on Sentence Length in Chunk-based Automatic Evaluation of Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Federmann, C. (2012, September). Appraise: An Open-Source Toolkit for Manual Evaluation of Machine Translation Output. *The Prague Bulletin of Mathematical Linguistics 98*, 25–35.

Gautam, S. and P. Bhattacharyya (2014, June). LAYERED: Description of Metric for Machine Translation Evaluation in WMT14 Metrics Task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Giménez, J. and L. Màrquez (2010). Linguistic measures for automatic machine translation evaluation. *Machine Translation 24* (3-4), 209–240.

Gonzàlez, M., A. Barrón-Cedeño, and L. Màrquez (2014, June). IPA and STOUT: Leveraging Linguistic and Source-based Features for Machine Translation Evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Guzman, F., S. Joty, L. Màrquez, and P. Nakov (2014, June). DiscoTK: Using Discourse Structure for Machine Translation Evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Klein, D. and C. D. Manning (2003a). Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, Stroudsburg, PA, USA, pp. 423–430. Association for Computational Linguistics.

Klein, D. and C. D. Manning (2003b). Fast Exact Inference with a Factored Model for Natural Language Parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS)*, pp. 3–10. MIT Press.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, Phuket, Thailand, pp. 79–86. AAMT: AAMT.

Koehn, P. and C. Monz (2006, June). Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proceedings on the Workshop on Statistical Machine Translation*, New York City, pp. 102–121. Association for Computational Linguistics.

Landis, J. R. and G. G. Koch (1977). The measurement of observer agreement for categorical data. *biometrics*, 159–174.

Leusch, G., N. Ueffing, and H. Ney (2006). CDER: Efficient MT Evaluation Using Block Movements. In *In Proceedings of EACL*, pp. 241–248.

Libovický, J. and P. Pecina (2014, June). Tolerant BLEU: a Submission to the WMT14 Metrics Task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Macháček, M. and O. Bojar (2013, August). Results of the WMT13 metrics shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, Sofia, Bulgaria, pp. 45–51. Association for Computational Linguistics.

Macháček, M. and O. Bojar (2014, June). Results of the WMT14 Metrics Shared Task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA, pp. 293–301. Association for Computational Linguistics.

Mahmoudi, A., H. Faili, M. Dehghan, and J. Maleki (2013). ELEXR: Automatic Evaluation of Machine Translation Using Lexical Relationships. In F. Castro, A. Gelbukh, and M. González (Eds.), *Advances in Artificial Intelligence and Its Applications*, Volume 8265 of *Lecture Notes in Computer Science*, pp. 394–405. Springer Berlin Heidelberg.

Nießen, S., F. J. Och, G. Leusch, H. Ney, et al. (2000). An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In *LREC*.

Och, F. J. (2003). Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, Stroudsburg, PA, USA, pp. 160–167. Association for Computational Linguistics.

Och, F. J. and H. Ney (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, Stroudsburg, PA, USA, pp. 295–302. Association for Computational Linguistics.

Och, F. J. and H. Ney (2003, March). A Systematic Comparison of Various Statistical Alignment Models. *Comput. Linguist. 29*(1), 19–51.

Papineni, K., S. Roukos, T. Ward, and W. jing Zhu (2002). BLEU: a method for automatic evaluation of machine translation. pp. 311–318.

Powell, M. J. D. (1964, January). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal 7*(2), 155–162.

Snover, M., B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul (2006). A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pp. 223–231.

Stanojevic, M. and K. Sima'an (2014, June). BEER: A Smooth Sentence Level Evaluation Metric with Rich Ingredients. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Tamchyna, A., M. Popel, R. Rosa, and O. Bojar (2014, June). CUNI in WMT14: Chimera Still Awaits Bellerophon. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA, pp. 195–200. Association for Computational Linguistics.

Vazquez-Alvarez, Y. and M. Huckvale (2002). The reliability of the ITU-t p.85 standard for the evaluation of text-to-speech systems. In J. H. L. Hansen and B. L. Pellom (Eds.), *INTERSPEECH*. ISCA.

Wu, X. and H. Yu (2014, June). RED, The DCU Submission of Metrics Tasks. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA. Association for Computational Linguistics.

Zaidan, O. F. and C. Callison-Burch (2009). Feasibility of Human-in-the-loop Minimum Error Rate Training. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, Stroudsburg, PA, USA, pp. 52–61. Association for Computational Linguistics.

Zhang, H. and D. Gildea (2007). Factorization of synchronous context-free grammars in linear time. In *Proceedings of the NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, pp. 25–32. Association for Computational Linguistics.

List of Figures

List of Tables

Appendix A

WMT14 Metrics Task Package Documentation

The `wmt14-metrics-task` directory located in the attached DVD-ROM contains scripts and data behind the results of WMT14 Metrics Task. The makefile in this directory is used to generate all the results. The command `make all` creates the following files which contain the results:

- `system.correlations.toEn`
- `system.correlations.fromEn`
- `segment.correlations.toEn`
- `segment.correlations.fromEn`

If you want to reproduce also the confidence intervals, change the following variables in Makefile:

```
COMPUTE_CONFIDENCE = true
SEGMENT_BOOTSTRAP_SAMPLES = 1000
```

You can use this package to evaluate your metric(s) on wmt14 data and compare it with other metrics. Create a subdirectory in `submissions/` and put there your metrics data files in the submission format as described at `http://www.statmt.org/wmt14/metrics-task/`. The file names have to end with `*.sys.score` or `*.seg.score` file extensions.

The scripts in this package have the following requirements:

- The `baselines/Makefile` requires a path to a compiled Moses, set the `MOSESROOT` env. variable.
- Scripts require Python 3 with *scipy* and *tabulate* packages installed.

Important content of this package:

- `metrics-task-paper.pdf` - the published paper with WMT14 metrics task results

- `submissions/` - the metrics data submitted by the task participants
- `baselines/` - the computation of the baseline metrics
- `compute-segment-correlations` - see `./compute-segment-correlations --help`
- `compute-system-correlations` - see `./compute-segment-correlations --help`
- `judgements-2014-05-14.csv` - the raw human judgements
- `human-2014-05-16.scores` - the official system-level human scores (TrueSkill)
- `human-2014-05-16.folded/` - the human scores computed on generated folds

Appendix B

SegRanks Application User Documentation

The `segranks` directory located in the attached DVD-ROM contains *SegRanks*, the annotation application which is used for ranking short segments of machine translation.

B.1 Installing and Running the Application

SegRanks is a Django web application written in Python and requires Python 2.7 and some additional Python dependencies. They are listed in `requirements.txt` and you can easily install them to a new virtual environment using the following command:

```
$ virtualenv /path/to/new/environment
$ . /path/to/new/environment/bin/activate
$ pip install -r requirements.txt
```

If you want to run the application locally, an *sqlite3* database is used. To initialize it and apply all database migrations, run:

```
$ ./manage.py syncdb
$ ./manage.py migrate
```

The database is now initialized and you can start the local web server

```
$ ./manage.py runserver
```

and open the application in your browser (usually `http://127.0.0.1:8000/`). However, the list of annotation project will be empty. You need to create a project and upload extracted segments. This is described in Section B.2.

The application is also ready to be deployed to Heroku cloud. To do that, you need in essence to create a new Heroku application, initialize a Git repository with *SegRanks*, add Heroku as a remote and push your commits there. It will automatically install all the dependencies. The application is configured to automatically set the database when deployed to Heroku. Please see Heroku documentation for more details.

B.2 Creating a New Annotation Project

To create a new annotation project, you need to have a file with extracted segments. The file has one row for each segment with the following tab separated fields:

1. sentence ID
2. tokenized source sentence
3. tokenized reference sentence translation
4. tokenized source segment
5. tokenized candidate segment
6. zero based indices of source segment words separated by a space (used for highlighting the segment in source sentence)

You can use the attached file `extracted.segments`, which contains segments extracted for the experiment in my thesis. To create a project, run:

```
$ ./manage.py create_project extracted.segments \
                "<Project Name>" "<Project Description>"
```

B.3 Annotating

Annotating in the application is very easy. Before you start, you need to be registered and signed in. To start annotating, select an annotation project you want to work on. You will be then shown annotation instructions and an annotated sentence. For each annotated segment, drag and drop segment candidates into the rank positions. When all the segment candidates are placed in the rank positions, the submit button is enabled and you can submit your annotation to the server. A new sentence is displayed to be annotated.

B.4 Printing Annotation Statistics

You can list annotators with various statistics (number of annotated segments, time spent annotating, agreements, etc.) with the following command:

```
$ ./manage.py statistics <project_id>
```

To get list of available projects with their IDs, run the command without the argument.

B.5 Exporting the Database

To export segments with their ranks in JSON or in Python Pickle format, use one of the following commands:

```
$ ./manage.py export_project <project_id> <out_file> json
$ ./manage.py export_project <project_id> <out_file> pickle
```

Both formats store a dictionary indexed by tuples of sentence IDs and source segments. The values of this dictionary are lists of rank dictionaries. The rank dictionary is indexed by candidate segments and its values are assigned ranks. The following listing is an example JSON output:

```
{
    "2386,Writing books saved me .": [
        {
            "Knihy psaní uložily mě .": 5,
            "Napsané knihy ušetřily mě .": 4,
            "Psaní knih mě zachránil .": 1,
            "Psaní knihy mě zachránil .": 2,
            "Psaní knihy mě zachránily .": 2,
            "Písemné knihy mě zachránily .": 5
        }
    ],
    "2755,At each station": [
        {
            "Na každé stanici": 1,
            "U každé stanice": 2,
            "V každé stanici": 2
        }
    ],
}
```

You can also find all the annotated segments from experiments in my thesis in the file `annotated.segments`.

Appendix C

SegRanks Application Development Documentation

SegRanks is a Django web application which follows the standard Django's structure and guidelines. It uses Model-View-Controller (MVC) pattern, although in Django, views are called templates and controllers are called views. We will use the Django's terminology here.

The model describes the representation of the data stored in the database. Views prepare the data that gets presented to the user and also process user requests and updates the data. Templates describe how the presented data will look.

C.1 Model

The Django's object-relational mapping (ORM) is used to access the data in the database. It allows to manipulate with data using an object interface. There is no need to write SQL queries manually, all manipulations with data objects are translated to SQL automatically on background.

The database model is implemented in `segranks/model.py` using the Django's model API. You can see the model illustrated in Figure C.1.

Annotation projects are stored in table *RankProject*. Each project contains a number of sentences stored in table *Sentence*, together with reference translations.

All extracted segments of a sentence are stored in table *Segment*. This table has two important fields. The first is *candidates_str*, which stores tab separated strings of all candidate translations of the segment. Although this is not a normalized design, it makes a lot of things much simpler (all the candidates are ranked at once anyway). The second field is *segment_indexes* which stores the indices of words of the segment in the source sentence. This is used for highlighting the segment in the interface.

Finally, each segment has zero or more annotations stored in *Annotation* table. When an annotation is submitted to the server, a new row for each segment in the sentence is created in this table. The most important field in this table is *ranks* which stores ranks of the segment candidates separated by tabs as a string. The reason for breaking the database normalization is the same as before. There are convenient getters and setters for these unnormalized data, so this is only an

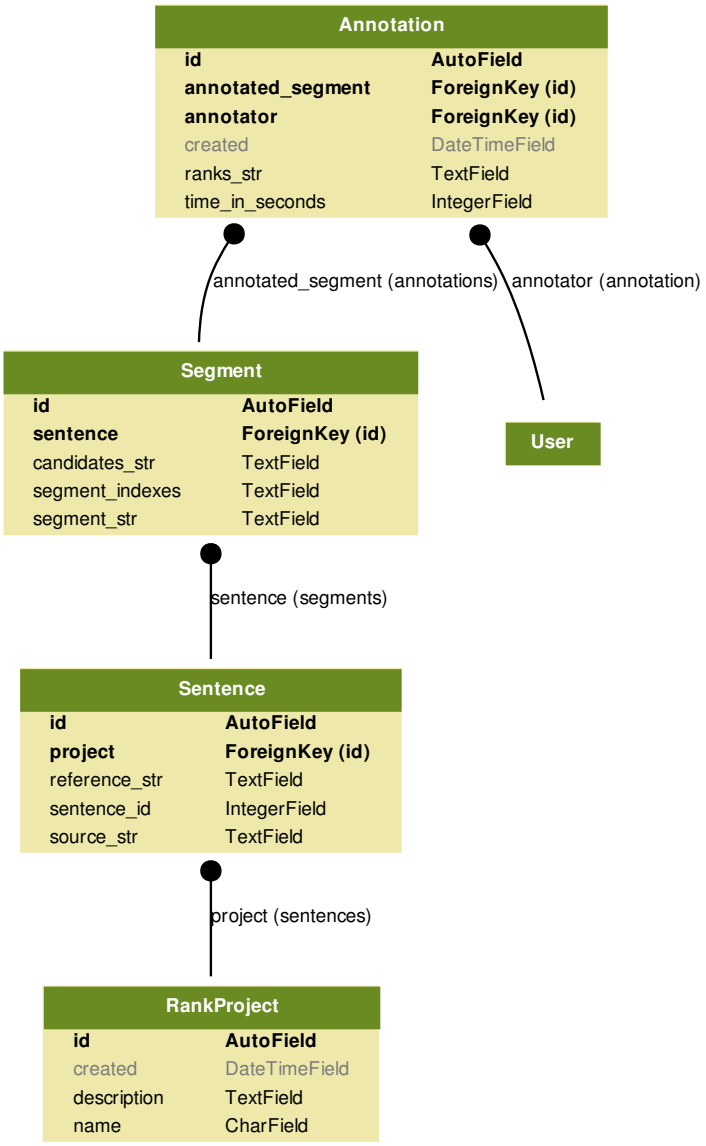


Figure C.1: Database model

implementation detail, hidden from the rest of the application. The table also stores some metadata (who created the annotation, how long did it take, etc.).

Because there were some changes of the database model during the development, we use *South* to track the database changes and easily migrate the data.

C.2 Views

Application URLs are mapped to individual views in file `segranks/urls.py`. All views are implemented as classes in file `segranks/views.py`. There are three important views.

The simplest one is *ProjectListView*, which lists all available annotation projects. It is implemented as a subclass of Django's generic *ListView* class and does not implement any method.

AnnotateView chooses a sentence to be annotated. This view is implemented as a subclass of generic *DetailView* class which renders a single object. The only implemented method here is *get_object* which returns a *Sentence* object to be rendered. This method ensures that annotators are sometimes shown already annotated sentences (for the computation of agreements). In most of the cases, it simply choose a random unannotated sentence.

The last view is *SubmitView*. This view processes annotated sentences submitted by annotators back to server. The ranks and other data are extracted from the POST data and *Annotation* instances are created for each annotated segment. After that, the user is redirected to *AnnotateView* to get a new sentence to annotate.

There are also views which handle user registrations and logins, however these views are part of the *django-registration* library.

C.3 Templates

All templates are located in directory `segranks/templates`. The `base.html` template contains all the elements which are common for all templates (HTML head with links to client scripts and cascade styles, common navigation bar) and other templates derive from it. Other templates contain HTML code specific to views which they render.

The templates use *Twitter Bootstrap* framework to easily build a simple and responsive user interface. We also use *jQuery* for easy element selection and manipulation and for event handling. The drag-and-drop feature is implemented using *jQuery UI Sortable* widget.