MASTER THESIS

Dušan Variš

Automatic Error Correction of Machine Translation Output

Institute of Formal and Applied Linguistics

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

Prague 2016

Title: Automatic Error Correction of Machine Translation Output

Author: Dušan Variš

Institute: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Ondřej Bojar, Ph.D., Institute of Formal and Applied Linguistics

Abstract: We present MLFix, an automatic statistical post-editing system, which is a spiritual successor of the rule-based system, Depfix. The aim of this thesis was to investigate the possible approaches to automatic identification of the most common morphological errors produced by the state-of-the-art machine translation systems and to train sufficient statistical models built on the acquired knowledge. We performed both automatic and manual evaluation of the system and compared the results with Depfix. The system was mainly developed on the English-to-Czech machine translation output, however, the aim was to generalize the post-editing process so it can be applied to other language pairs. We modified the original pipeline to post-edit English-German machine translation output and performed additional evaluation of this modification.

Keywords: automatic post-editing, machine translation, supervised machine learning, natural language processing, Treex

I would like to thank my supervisor, RNDr. Ondřej Bojar, Ph.D., for valuable advice and support during writing this thesis. I would also like to thank Matěj Trojan for helping me with the evaluation of Czech MLFix output. Finally, I would like to thank Ladislav Valkovič, Ph.D., Mgr. Ondřej Dušek and Radek Síleš for helping me with evaluating German MLFix output and providing me information about the language.

This thesis is dedicated to them.

Contents

1. Introduction

In this thesis, we present a statistical post-editing tool MLFix based on its rule-based predecessor Depfix (Rosa, 2014). We aim to use statistical machine learning methods to generalize a subset of Depfix rules and create a fairly language-independent automatic post-editing (APE) tool. Our main goal is to find a reasonable trade-off between the amount of linguistic knowledge needed in the training and input data and the dependence on language-specific third-party analysis tools. We also want to specify the machine learning (ML) task which our APE component should accomplish and train a sufficient ML model for correcting machine translation (MT) output. The system was developed using English-Czech language pair, however, we also present a preliminary results we gathered from English-German language pair experiments.

1.1 Task Motivation

Even though the state-of-the-art MT systems have been gradually improving in the past few decades, they are still not perfect. Currently, the most popular statistical machine translation (SMT) systems can be fairly effective even with little or no linguistic knowledge about the concerned language pair given they have access to sufficient amount of parallel data. However, when translating into morphologically rich languages, data sparsity increases rapidly and such systems quickly begin to introduce grammatical errors into the translated sentences and worsen the overall fluency of the translation.

Still, these systems can be of help to the human translators since it is usually less costly to start with a partially incorrect translations provided by the SMT system and have a human translator correct them than to translate the source text manually. Human post-editors can however still be quite expensive so naturally, there have been attempts to automate this process with automatic post-editing tools.

1.2 Related Work

Up to this date, there have been various approaches to the task of automatic error correction of machine translation output, each with a different level of success.

The very first attempts in the field of automatic post-editing (Simard et al., 2007) focused on applying phrase-based statistical machine translation (PB-SMT) system on the output of rule-base machine translation. The system was trained on monolingual bitextual data containing the MT output as the source sentence and the reference translation as the target sentence. This phrase-based automatic post-editor (PB-APE) helped to significantly improve the performance of the rule-based MT system in question. There were also attempts to apply the post-editing component on a phrase-based translation system but the combined system performed slightly worse in comparison with the standalone SMT system.

Further experiments with PB-SMT + PB-APE (Béchara et al., 2011) were done on English-French and French-English translation, where significant improvements were reported for the latter translation direction. They improved the design suggested by Simard et al. by creating a purely statistical pipeline. The PB-APE was then expanded by adding further context information about the source sentence to the SMT-generated output (used as an input for the PB-APE) reporting further improvement in performance.

These previous attempts can be considered purely statistical since only a little amount of linguistic knowledge about the concerned languages was used during development. To gain further insight into the task of automatic post-editing, a more thorough analysis of the most frequent errors made by the current SMT systems was performed by Bechara (2013) and later by Rosa (2013), the former for English-French and the latter for English-Czech.

The error analysis was later used during development of the rule-based APE Depfix (Rosa, 2014), which was designed to correct errors made by the English-Czech SMT systems. The system uses a set of finely hand-crafted rules that aim at identifying and correcting morphological errors such as incorrect agreement or valency, which are often encountered during English-Czech machine translation. It does not focus on a lexical errors although some minor corrections, e.g. insertion of missing reflexive particles, are made. The system succeeded to improve output of various MT systems and was deployed as a stable part of the Chimera (Bojar et al., 2013a) MT system.

The idea behind Depfix system seems promising, however, due to its rule-based nature, it is difficult to apply the APE on a different language pair since it would require costly modifications to the existing set of rules. One of the goals of this thesis is to try to replace the rule-based blocks by statistical ones so they can be applied to MT output in a new target language more easily, simply by training an appropriate statistical model.

1.3 Thesis Structure

In Chapter 2, we describe MLFix data processing pipeline and introduce all the tools we use. In Chapter 3, we take a look at all datasets which we used during our system development and analyze their usefulness for our system. In Chapter 4, we define the post-editing task, go through various modifications of the task we considered and explain our approach with the support of our analysis of the available data. In Chapter 5, we describe the process of the development of the statistical models for the MLFix post-editing component and present results of the evaluation of the trained models. In Chapter 6, we evaluate the performance of the whole MLFix system and analyze the level of contribution to the resulting performance of each individual statistical component. In Chapter 7, we briefly describe our modifications to English-Czech pipeline when used on English-German language pair, then summarize the differences in the training data and model training between the two language pairs. The chapter concludes with an evaluation of the English-German pipeline. We conclude the thesis in Chapter 8.

We include a summary of the contents of the attached CD in Attachment A and a full English-Czech and English-German Treex scenario in Attachment B.

2. System Description

In this chapter, we describe the main components of MLFix system and take a closer look at the suggested processing pipeline. We focus on English-Czech pipeline, however, we also describe modifications needed to apply MLFix for other language pairs.

Full examples of currently deployed Treex scenarios (English-Czech and English-German) are included in Attachment B.

2.1 Processing Pipeline

MLFix is, similarly to its predecessor, almost entirely implemented in the Treex (Popel and Žabokrtský, 2010)[1] framework (formerly known as TectoMT). The framework was originally created as a basis for English-Czech hybrid translation system, combining rule-based modules with statistical models and using deep semantic language representation for sentence translation. However, due to its modularity, it is now used for various tasks of natural language processing (NLP) across different languages. The framework was built to support the methodology of the theory of Functional Generative Description (Sgall, 1967) and was adapted to support sentence representation in Prague Dependency Treebank (Hajič et al., 2006). Mainly, it supports the representation of sentences on different layers of abstraction defined in FGD: morphological layer, analytical layer and a tectogrammatical layer, usually referred to as m-layer, a-layer and t-layer where prefixes m-, a- and t- are also used to refer to the objects at the corresponding layer of abstraction. Because the tools for tectogrammatical layer analysis are currently available for a limited set of languages (mainly Czech and English, with others in progress), we decided to use only the a-layer (surface syntax) for data representation.

2.1.1 M-Layer Analysis

MLFix analysis pipeline is derived from an existing Depfix pipeline with a several modifications to make it easier to apply to different target languages. MLFix takes a pairs of source sentences and their MT outputs, which are aligned on a sentence level, as the input. Additionally, sentence-level aligned reference translations are expected during the system training. The input data are first read in parallel and stored into Treex internal representation. Both source side and MT side are tokenized by a rule-based tokenizer, each token is then represented by a separate m-node.

Next, lemmatization and part-of-speech (POS) tagging is performed. For both English and Czech, we use MorphoDiTa (Straková et al., 2014)[2] tool for morphological analysis and tagging. MorphoDiTa is also used for Czech lemmatization. For English lemmatization, we use a rule-based block implemented in Treex. It is important to provide a POS tagger for the target language that supports relatively fine-grained morphological tags because our goal is to correct morphological

[1]`http://ufal.mff.cuni.cz/treex`

[2]`http://ufal.mff.cuni.cz/morphodita`

errors represented mainly through these tags. It was reported by Rosa (2013, p. 33) that the tagger produces significantly more errors in morphological analysis of Czech SMT outputs in comparison with normal text. In Depfix, this is covered by a rule-based block that identifies these errors and changes the incorrect morphological tag without changing the surface form. We decided to omit this block and leave the issue to our statistical component.

The last step we do in the scope of the m-layer analysis is a transformation of the morphological tags into a more general representation. Optionally, we can apply a named-entity recognition tool if one is available but it is not mandatory. For English, we use the Stanford Named Entity Recognizer (NER) (Finkel et al., 2005)[3], for Czech, we use a simple rule-based NER.

2.1.2 Interset

Since there are usually different tagsets used across individual languages, often engineered for purposes of that specific language and with no standardized tag representation, we would be forced to modify our existing pipeline to some extent every time a new language would be introduced. Therefore, we have decided to use Interset (Zeman, 2008)[4], an interlingua-based representation of morphological tags from various tagsets. To be able to use this representation to represent tags from a given tagset, a decoding/encoding module is required. However, the support for various tagsets spanning through different languages started growing lately mainly due to Universal Dependencies (Nivre et al., 2016)[5] project.

After the transformation, in following steps of the analysis, a choice of a specific tagset becomes transparent because MLFix blocks only have to deal with one well-defined set of features.

2.1.3 Word Alignment

In the next step, we create a word-level alignment for each sentence pair using GIZA++ (Och and Ney, 2000). We create one-to-one word alignment between the source language and the MT output where possible using the intersection symmetrization.[6] This step helps us later with feature extraction (e.g. extracting the information about the aligned source node) and with further processing of the target sentence.

In the process of training data extraction, we also create a simple alignment between the MT sentence and the reference sentence exploiting forms, lemmas and tags of the m-nodes. This is done by a language-independent rule based block which was already implemented in Treex, `Align::A::MonolingualGreedy`. Then, we create alignment between source sentences and reference sentences by combining the *SRC-MT* and *MT-REF* alignments created earlier.

[3]`http://nlp.stanford.edu/software/CRF-NER.shtml`

[4]`https://ufal.mff.cuni.cz/interset`

[5]`universaldependencies.org`

[6]When post-editing SMT output, we might expect to be provided the word alignment by the SMT system we are correcting. However, this would put additional constraint on the input data format so we do not make this assumption.

	Preprocessing		
	Source (English)	MT (Czech)	Reference (Czech)
Training data	MDiTa + MST	MDiTa + src_proj	MDiTa + MST
Test data	MDiTa + MST	MDiTa + src_proj	–

	Alignment		
	SRC-MT	MT-REF	SRC-REF
Training	GIZA++	Rule-based	SRC-MT + MT-REF
Test data	GIZA++	–	–

Table 2.1: Summary of the tools used to process source (English), MT (Czech) and reference (Czech) sentences. For preprocessing *tagger+parser* pairs are presented, MorphoDiTa (*MDiTa*) for tagging and *MST* parser or projection of the source trees (*src_proj*). Additionally, alignment type (GIZA++, Rule-based, or combination of alignments) is listed for each sentence pair.

2.1.4 A-Layer Analysis

After the m-layer analysis and the word alignment, we perform dependency parsing. For English, we use Maximum spanning tree (MST) parser (McDonald et al., 2005)[7] implemented in Treex framework. For SMT output, even though there might be an existing dependency parser available for the target language, it is usually trained on data that do not contain errors. Therefore, it has usually significantly lower performance when applied on the SMT output. research on Depfix has shown that the knowledge of the dependency structure of the SMT output can provide additional valuable information for identifying grammatical errors[8], thus improving the Depfix performance.

For the time being, we decided to build the dependency structures of the SMT output simply by projecting the dependency structure of the source sentence onto the target side using the word alignment we extracted in the preceding step. The resulting structure will most likely contain errors, but should be at least consistent throughout our data. To compensate for the lower accuracy of the extracted dependency structure we perform dependency parsing of the reference sentences during training, if a proper parser is available. For this purposes, we use the MST Parser for the Czech language as well.

Note that for Czech SMT output, an implementation of the MST parser adapted for the SMT output is already available (Rosa et al., 2012a), however, so far we have not done any experiments to determine whether the dependency structures provided by the adapted parser (which should be more accurate than our projected trees) influence the final performance of our system.

We summarize the methods used for preprocessing source, MT and reference sentences and methods for creating the corresponding word alignments in Table 2.1.

[7] `http://sourceforge.net/projects/mstparser`

[8] Actually, in the case of Depfix , the information about the dependency structure is crucial for most of the fixing components because e.g. the parent-child relationship is examined almost every time.

2.2 Statistical Component

After data preprocessing, we extract all available features and apply a trained statistical model. The features are extracted separately for each node and passed to a model. The model needs to accomplish these two goals:

1. identify candidate words with incorrect morphology,
2. fix incorrect morphological features.

Of course, these two steps can be split between multiple separate components (and models). We describe the post-editing process in more detail later in chapter 5.

We decided to use Scikit-Learn (Pedregosa et al., 2011)[9] toolkit to train and execute our models since it has an easy-to-use and quite uniform interface, which allows us to try out different machine learning (ML) methods simply by switching the model class. In Treex, we use a simple wrapper to load and execute the trained Scikit-Learn model. If a support for a different ML implementation is needed (e.g. VowpalWabbit[10,11]), this wrapper can be easily modified to suit the requirements.

2.3 Wordform Generation

After we have identified morphologically incorrect words and assigned them a new morphological categories, we need to generate new surface forms reflecting the changes we have made. This can be done either by rule-based component or with the help of another statistical model.

For Czech, we have used a morphological generator build upon the morphology of Hajič (2004) that is already part of the Treex framework. For other languages, when there was not another option available, we used Flect (Dušek and Jurčíček, 2013)[12]. Flect is a language independent morphological generation tool also using Scikit-Learn models. The tool learns morphological inflection patterns form an annotated corpus and should be able to inflect even previously unseen words using lemma suffixes as features and predicting the difference between the lemma and the necessary surface form given some morphological specifications.

2.4 Language Independence

From the previous description, we can notice that MLFix still depends on several language specific tools. It is quite dependent on the capability of a source language analysis (in our case English), because we do not only need a POS tagger but also a dependency parser. However, we assume that source sentences are usually grammatically correct so it is much easier to provide required tools than their modified versions targeted at the MT output.

[9]http://scikit-learn.org/stable/

[10]https://github.com/JohnLangford/vowpal_wabbit/wiki

[11]At the time of writing this thesis, there is already a limited support for VowpalWabbit in Treex.

[12]https://ufal.mff.cuni.cz/flect

We also require a specific decoder of the source and target POS tags into Interset feature structures but Interset already covers a large variety of the most widespread tagsets available and its support is still growing.

Finally, aside from a language specific post-editing models (which we do not expect to be reusable across different languages) we require a module that regenerates the corrected wordform (either from the "*lemma+tag*" or "*lemma+Interset features combination*" specification). A state-of-the-art tool might not be explicitly available for every language but if no other option is provided, we can use a statistical form generator, in our case Flect.

3. Available Data

In this chapter we take a closer look at available sources of data and describe how they contributed to our research.

We came across various sources of training data with various level of usefulness. The data was usually available only in a smaller volume. Some of the sources are:

- Khan Academy[1] human post-edits of manually translated (EN→CS) subtitles,
- Autodesk triparallel data[2],
- log files of human post-editing done by Lingea for the Health in my Language[3] (HimL) project test dataset,
- results from the previous workshops on machine translation (mainly WMT10 (Callison-Burch et al., 2010)[4], and WTM16 (Bojar et al., 2016)[5] datasets).

In the following sections, we describe each in more detail.

3.1 Khan Academy

The data provided by Khan Academy consist of English-Czech subtitles, where the Czech part (usually manually translated from English) was manually edited. During the analysis of the dataset, we noticed that most of the time, the corrections were made mostly on the lexical level which is to be expected since the Czech sentences were created by a human translator. Therefore, we concluded that this dataset has little to no value for the task of training a model for correcting errors in morphology created by MT systems.

3.2 Autodesk

Autodesk data consist of English sentences which were machine translated into a set of target languages (cs, de, pl etc.) complemented with human post-editing of the MT output. However, these datasets are domain specific (mostly user documentation), so they might not be very attractive to use with more general texts. We were not able to find any information about the MT system that was used to create the translated output. The biggest advantage of these data is their larger volume when compared with other post-editing datasets so we used them mainly for model development and benchmarking of the used machine learning methods.

[1]`https://khanovaskola.cz/`
[2]`https://autodesk.app.box.com/Autodesk-PostEditing`
[3]`http://www.himl.eu/`
[4]`http://www.statmt.org/wmt10/`
[5]`http://www.statmt.org/wmt16/`

3.3 HimL-Lingea Logs

The data provided by Lingea[6] were collected when official test sets for HimL project were created. The data consist mainly of texts related to public health. The original English sentences were first machine-translated to languages at which the project was aimed (Czech, German, Polish and Romanian) and then post-edited by professional translators using Lingea's post-editing tool. The datasets are probably the most detailed one since they consist of complete logfiles describing elementary actions taken by human post-editors (such as selecting phrases in a translated sentence, looking up alternative translations in a dictionary etc.).

When we examined the data more closely, we noticed that it is rather difficult to determine which actions are useful for our machine learning process. Also, we were a little disappointed when we found out that most of the time, the post-editors preferred to rewrite the whole sentence "from scratch"[7] instead of doing a few atomic modifications to the provided MT output.

In the end, we decided to simply extract triparallel data from these logs (the source sentence + SMT output + result of the human post-editing). In the future, we might consider to use other logged actions for model training.

3.4 WMT Datasets

For the last decade, the workshop on machine translation (WMT) has aimed to provide working grounds for many researchers in the field of machine translation. It has been great source of parallel data between English on one side and various other languages on the other. Each year, the scope of the workshop expands, including various new subtask related to machine translation, such as several evaluation tasks and, more recently, automatic post-editing task.

The data available for the post-editing task usually contains a set of:

- source English sentences,
- output of various MT systems, usually the ones that participate in the main translation tasks,
- either a reference sentences or human post-editing of the MT output from the participating MT systems.

These datasets give us the opportunity to compare the performance of our post-editing models when applied to the different systems. Even though the data available for the WMT subtasks are often from various domains (news, IT, biomedical), the domain of the post-editing data is more limited, mainly to the news articles.

[6] `http://www.lingea.cz/`

[7] By that, we mean that the human post-editor usually preferred to rewrite the whole corrected sentence, even though only a several changes (either lexical and morphological) were made, and delete the original. This might have been also motivated by the need to reorder the MT output.

3.5 Other Sources

The sources listed above (each one to a different degree) can be considered a knowledge base for examining the behavior of a human post-editors as well as training data for our system. We think that they provided us with some interesting insight into the post-editor's thought process. On the other hand, we have also considered using other sources since the data mentioned above are quite limited in size.

One possible way to tackle the shortage of training data is to use available parallel corpora. These corpora (containing only *source sentences + reference translations*) can be expanded by translating the source sentences and thus creating a set of sentences which contain MT generated errors and should be fixed to resemble the reference translation. These data can be then used to train post-editing models for that specific SMT system[8] or possibly for other SMT systems. This method can surely help to overcome the aforementioned data acquisition bottleneck since there is generally much more parallel data then post-edited sentences. For English-Czech language pair, the natural choice of the parallel corpus would be CzEng 1.0 (Bojar et al., 2012b)[9].

Of course, this aproach introduces some additional noise related to the post-editing task. For example, we can get fluent and correct MT output which is very different from the reference translation because sentences can easily have hundreds of thousands of correct translations (Bojar et al., 2013b). Therefore, it can be hard to distinguish which of the differences between MT and the reference translation are genuine MT errors.

The basic summary of the available data is shown in the table Table 3.1.

3.6 Monolingual Data

We have also considered using bitext *monolingual* data (either *MT output + post-edited sentences* or *MT output + reference translations*), however due to the nature of our processing pipeline we would be much more limited when analyzing the training data. Also this way, we would lose the additional information that can be extracted from the source sentences, which proved to be valuable in the practice (Rosa et al., 2012b).

[8]Obviously, the post-editing model training data have to be different from the parallel data used for the training of the SMT system, so some jackknife sampling should be used with limited training data.

[9]`http://ufal.mff.cuni.cz/czeng`

	# Sentences	# Tokens English	# Tokens Czech		
			MT	PE	REF
Khan Academy	∼14k	∼93k	∼93k*	∼93k	–
Autodesk	46,916	490,005	456,697	441,645	–
HimL-Lingea	3892	60,142	51,428	56,485	–
WMT10	2,489	54,021	44,578	–	45,422
WMT16	2,999	57,418	48,037	–	48,915
CzEng 1.0	15M	206M	–	–	150M

* Translations in Khan Academy data come from humans.

Table 3.1: Summary of the available post-editing data. Only English-Czech data is listed, however, for datasets where data for other language pairs are available, their volume is approximately the same. We provide only rough estimates for the Khan Academy data. There is no information about the number of tokens in the MT part of CzEng because we decided to abandon the idea of creating a triparallel corpus for the time being.

4. Task Definition

In this chapter we present results of the closer inspection of the available training data and describe our process of developing the MLFix post-editing component.

When we closely inspected the morphological errors present in our data we decided to approach the post-editing problem as a classification task. Basically, the main idea is to identify words with incorrect surface form, assign new morphological categories to such words (e.g. via new morphological tags, Interset categories) and generate a new surface form. This approach seems reasonable, since it gives us fair amount of freedom in generalizing the morphological post-editor, by customizing the scope of the trained classifier or classifiers (e.g. by specifying the set of categories being predicted by the classifier or the set of instances, which can be modified by the predictor). On the other hand, it raises several issues that have to be resolved, mainly:

- What instances should we extract from our training data?
- Which of the extracted instances should we mark as incorrect, as opposed to instances that represent words with correct surface from?
- Is it necessary to extract error-free training instances?
- What features should we extract for each instance?
- How should we apply the trained classifier? Should we apply it on each word in a sentence or should we identify the erroneous words first?
- What morphological categories should be predicted by our classifier?

Another important question is how should we measure the performance of our models. Obviously, we cannot base the quality of our system on the performance of the classifier itself since even a really well performing classifier can have only a small or negative impact on the edited sentences (e.g. if we choose to predict morphological categories that have very small impact on the final surface form, or we incorrectly identify erroneous words in our training data). Still, the standard metrics used for classifier evaluation, such as accuracy, precision and recall, can be helpful during some stages of development (e.g. the choice of machine learning method, hyperparameter tuning).

However, in the end, our main goal is not producing a well performing classifier, but creating more fluent, grammatically correct sentences. Naturally, for this purpose, human evaluation of the post-edited sentences is the best choice as far as reliable judgement goes, but it is also very costly and we usually need a more efficient method during system development. Therefore, we rely on the widely used BLEU scoring and probably even more suitable translation error rate (TER) metric. Relying on these methods alone has however a few drawbacks which we will address later in the in this chapter.

4.1 Specifying the Task

As mentioned above, our goal is assigning new surface forms to the morphologically incorrect words in the MT translated text via newly predicted morphological tags. For morphologically rich languages, this can be quite difficult due to large tagsets. Also, most of the time, the incorrect surface form of the word is only a result of an incorrect value in a small subset of morphological characteristics of the word (e.g. wrong case, number, gender). These reasons are why we decided to use the Interset categories instead[1] and allow for predicting only a few categories at a time.

Eventually, when we were deciding on how our post-editing system should work, we have settled on three possible scenarios:

1. Have one completely general classifier that is applied on each word in the sentence (some words might be omited by hand-crafted rule, e.g. ignore classes that do not flect).
2. Identify the erroneous words first by a separate classifier. Apply a second classifier on the marked words and predict new morphological categories.
3. Identify the erroneous words same way as in the previous scenario however, this time also choose a classifier that should be applied in the second step. The choice can be simply deterministic (e.g. via general POS), or stochastic.

The first scenario can be appealing because it is very easy to implement and requires only one model. The problem is that requirements for such model might be way too big and difficult to satisfy. Another problem might be an unbalanced training dataset since most of the training examples will represent “correct” (as far as fluency of the translated text is concerned) instances, where predicting a new morphological categories is not desired. Therefore producing a well performing model can become difficult in the end.

We thus find the second scenario much more plausible and it was our main focus during this research. It requires a simple binary classifier for the first step and one multiclass (or possibly multitask[2]) classifier. When we train the binary classifier (which identifies the words that need to be corrected) we still have to face the issue of an unbalanced training dataset, on the other hand, the classifier assigning new morphological categories can be simply trained only on the incorrect instances.

The third scenario seemed reasonable because we usually want to modify different morphological categories for different POSes. However due to the results of the data analysis (presented in more detail in Section 4.2), we found it unnecessary to implement for the time being.

[1]It is not necessary for Czech, since the Czech positional tagset allows us to modify only necessary parts of the morphological tag. However, other languages do not provide similar tagset, therefore Interset might be more feasible representation.

[2]We define a multitask classifier as a classifier combining two classification tasks together (e.g. predicting new number and new case). The tasks are separate, however the classifier takes the possible relation of the two tasks into account.

4.1.1 Oracle Classifier

Now that we have outlined the post-editing subtasks and assigned hypothetical classifiers to solve each one of them, there are following two issues left to resolve: the choice of training instances and the choice of classifier targets. Since we only focus on correcting morphological errors generated by the MT system and we ignore lexical errors completely, we have to be careful during the extraction of training instances for our classifier. We have found it very helpful to use a "fake" classifier for this task, that we simply call *Oracle*.

The basic idea behind the Oracle classifier is that it has access not only to the source sentences and the MT output like our production classifier but also to the "correct" answers contained in the *reference translations/post-edited sentences*. The most important task for the Oracle is to help us to observe whether the suggested definition of *training instances* and moreover the definition *incorrect instances* (the instances, which require post-editing) has a potential of improving the MT output (if we had a perfect classifier).

We decided to extract one *training instance* for each word in our data that meets all of the following criteria:

- the *MT-node lemma* IS_EQUAL to the *REF-node lemma*,
- the *MT-parent_node* IS_DEFINED AND
  the *MT-parent_node* IS_NOT_ROOT,
- the *REF-parent_node* IS_DEFINED AND
  the *REF-parent_node* IS_NOT_ROOT,
- the *SRC-node* IS_DEFINED,
- the *SRC-parent_node* IS_DEFINED AND
  the *SRC-parent_node* IS_NOT_ROOT,

We have settled for these criteria for the following reasons: we want to extract only instances that have true predictions available (thus the check for the presence of the aligned REF-node), we want to ignore misleading instances related to possible lexical errors or different lexical choice (equality of the SRC-node and REF-node lemmas) and finally, we want to make sure that enough relevant tree-based context information will be extracted. We have not modified the definition of the training instance much during the research. On the other hand, we experimented with several definitions of the *incorrect instance*.

We tried three different heuristics to identify *incorrect instances*, each with a different set of conditions:

1. the *MT-node form* IS_NOT_EQUAL to the *REF-node form*,
2. the *MT-node form* IS_NOT_EQUAL to the *REF-node form* AND the *MT-parent form* IS_EQUAL to the *REF-node form*,
3. the *MT-node form* IS_NOT_EQUAL to the *REF-node form* AND (the *MT-parent form* IS_EQUAL to the *REF-node form* OR the *MT-parent form* is marked as INCORRECT)

All definitions work in the context of the *training instance*. For later reference, we refer to these heuristics as *WrongForm1*, *WrongForm2* and *WrongForm3* respectively. In any case, if the conditions are not met, the training instance is marked as correct and should be left unchanged by the classifier. These definitions cover both instances that should be changed (*correct* vs. *incorrect* instance) and how the incorrect instances should be modified (values from the REF-node). The surface forms generated with the use of these training instances should be identical to those of the reference, if all morphological categories, that are different from the reference, are properly set before generating the new form. This fact has one major drawback related to the evaluation: the quality of the Oracle (and therefore the "best" possible result) or a production classifier cannot be reliably measured by automatic n-gram based metrics, because the post-edited sentences will always have same or better score than the MT output. Therefore, manual evaluation is required to some extent.

The first method (comparing only the surface forms) marked about one-tenth of the training instances as incorrect. When used in combination with the Oracle classifier, the post-edited sentences have shown moderate improvement in the automatic metrics. However in a closer observation of the sentences, we noticed many incorrect modifications such as Example 4.1. Clearly, even though the wordform of "*místo*" has been changed to match the reference, the governing verb "*mít*" requires its dependent to be in the dative case. The change has thus actually introduced a new grammatical error into the sentence and thus worsened the fluency of the translated sentence. Additionally, by changing the case of the word "*místo*", the already correct agreement with the subordinate adjectives "*poslední*" and "*volná*" got also broken, making the result even worse. On the other hand, the wordform in the reference sentence is correct because the governing word of "*míst*" is a noun ("*pár*") instead of a verb resulting in a genitive case of the word.

Source:	**We have the last few vacancies for New Year's Eve and Christmas.**
SMT output:	*Máme poslední volná* **místa** na Silvestra a Vánoce.
Gloss:	We *have* the *last*$_{dative}$ *few*$_{dative}$ **vacancies**$_{\mathbf{dative}}$ for New Year's Eve and Christmas.
Oracle output:	*Máme poslední volná* **míst** na Silvestra a Vánoce.
Gloss:	We *have* the *last*$_{dative}$ *few*$_{dative}$ **vacancies**$_{\mathbf{genitive}}$ for New Year's Eve and Christmas.
Reference:	Na Silvestra i na Vánoce *máme posledních pár míst.*

Example 4.1

The previous example has shown that for identifying incorrect instances, some additional information about the surrounding members of the sentence is required. We have seen that by slightly altering governing nodes, e.g. by only choosing a different lexical translation of the source node or by choosing a completely different expression, the equality between the surface form of the MT word and its reference counterpart cannot be enforced without harming the quality of the MT sentence. Therefore, we tried to introduce additional constraints to correctly identify candidates for post-editing. To mark and instance as incorrect, not only the ref-node form has to be different from its aligned ref-node form, but we must make sure that their governing nodes have identical surface form. We considered checking only for the lemmas of the governing nodes at first, but this constraint was too soft and was not able eliminate some of the previous errors. The main motivation behind this constraint is to identify at least some agreement and possibly valency errors without too much of language-specific insight. We assume that if the surface form of the mt-node and ref-node differ while their parent node is identical, there is a high chance that the mt-node's surface form is incorrect while the reference node has the right correction[3].

This additional constraint helped us to remove a large number of false positives and produced training examples such as Example 4.2. In this example, we can see that the noun "*život*" was correctly changed to "*života*" because of the valency frame of the verb "*sdílet*". However, we can see, that this constraint might be too strict because the adjective "*akademického*" was left unmodified even though by changing its governing noun "*život*", the agreement present in the MT output and should have been preserved after the post-editing was left unnoticed. The post-editing of the word "*akademického*" was omitted, because the surface forms of the governing nodes in the MT sentence and the reference sentence ("*život*" vs. "*života*") did not match. We observed this kind of false negative quite often which led us to introducing one additional constraint.

Source:	**. . . where he acknowledged the "wonderful people" he shared his academic life with.**
SMT output:	. . . kde potvrdil, že je "skvělí lidé" *sdílel* jeho akademického **života**.
Gloss:	. . . where he acknowledged, that is "wonderful people" he *shared* his academic **life$_{\textbf{genitive}}$**.
Oracle output:	. . . kde potvrdil, že je "skvělí lidé" *sdílel* jeho akademického **život**.
Gloss:	. . . where he acknowledged, that is "wonderful people" he *shared* his academic **life$_{\textbf{dative}}$**.
Reference:	. . . do poděkování "skvělým lidem", s nimiž *sdílel* akademický *život.*

Example 4.2

To soften the previous constraint in favor of not breaking the agreement of the dependent nodes, we decided that if the previous constraint is not satisfied (possibly due to a difference between the mt-parent surface form and the ref-parent surface form), the node in question should still be marked as incorrect if

[3]This, of course, depends on the quality of the dependency tree produced by the parser. However, since we use different parsing methods for mt-side (projection of the src-tree) and the reference (dependency parser) we expect that the number of false positives in our training data will be lowered.

and only if the governing node was marked as incorrect and the node's surface form is different form that of the reference node. This is basically similar to post-editing the MT output dependency tree recursively from its root to its leaf nodes.

Applying this new constraint we were able to produce training examples similar to Example 4.3. As we can see in the MT output, there is correct agreement between the words "*Potrefená*" and "*husa*",[4] however, they have broken agreement with the governing preposition "*naproti*". This is corrected in the Oracle output. Even more interesting is the phrase "*narazit na moravské náměstí*" and its counterpart, "*narazit na moravském náměstí*", both being grammatically correct while having different meaning. The first one means literally "*to come across moravské náměstí*" ("*náměstí*" meaning "*square*" in Czech), while the other one can be translated as "*to come across (someone/something) on moravské náměstí*". In this case the phrase created by the Oracle classifier (even though it does not work on the phrase-level) is closer to the original meaning. We might notice, the some named entities were not correctly identified ("*Moravské*", "*Husa*") but correcting these is not a goal of the original task.

Source:	**The only place we've managed to come across is on Moravské náměstí, opposite the Potrefená Husa.**
SMT output:	Jediné místo, kde se nám podařilo *narazit na* **moravské** *náměstí naproti* **Potrefená husa**.
Gloss:	The only place, where we've managed to *come across* **Moravské**$_{\mathbf{dative}}$ *náměstí*$_{dative}$ *opposite the* **Potrefená**$_{\mathbf{nominative}}$ **Husa**$_{\mathbf{nominative}}$.
Oracle output:	Jediné místo, kde se nám podařilo *narazit na* **moravském** *náměstí naproti* **Potrefené huse**.
Gloss:	The only place, where we've managed to *come across* **Moravské**$_{\mathbf{locative}}$ *náměstí*$_{locative}$ *opposite the* **Potrefená**$_{\mathbf{genitive}}$ **Husa**$_{\mathbf{genitive}}$.
Reference:	Jediné místo, na které jsem zatím natrefil, je na *Moravském náměstí naproti Potrefené Huse.*

Example 4.3

It should be noted, that these constraints do not detect all possible morphological errors. For example, since we check mostly only the relationship with the governing node, we effectively omit subject-verb agreement errors. This can be possibly fixed in the future by additional constraints, however, in the scope of this thesis, our main objective was to find and evaluate a method for identifying and correcting morphological errors that is as general as possible.

We also must keep in mind that the final constraint, while producing reasonable training examples, still managed to produce instances containing false positives. Even though it might be caused by incorrect assumptions during the

[4] Even though "*Potrefená Husa*" is a named entity, in Czech, these are still flected with regard to the rest of the sentence structure.

designing of our heuristics, we should point out that there are instances such as Example 4.4, where it is difficult to decide whether the post-editing helped to improve or harmed the fluency of the MT output, possibly due to being only one of the several steps that needs to be taken during the post-editing. In this example, we can see that the adjective "*osvětleného*" was changed to "*osvětleném*" because the governing node "*baru*", being a part of the apposition with the "*Julep Room*", should be in the preposition-noun agreement with the preposition "*v*". However due to mistranslation of the named entity ("*Julep Room*") and wrong indication of the apposition relationship (missing comma), the reader might consider the correction made by the Oracle less fluent or unnatural. On the other hand, the meaning of the original MT output is quite different from the meaning in the source sentence, therefore, some post-editing is definitely required.

As we have seen, the task of extracting meaningful and correctly annotated training instances for our system gets increasingly difficult with growing difference between the MT output and the reference sentences. If provided output only from a poor MT systems, the task might be almost impossible. For this reason, we have tried another, somewhat limited, method for creating and extracting training instances.

Source:	**. . . entertainment coordinator at The Julep Room, a dimly lit bar near Gautier, said. . .**
SMT output:	. . . programový koordinátor *v* Julep *místnosti* **osvětleného** *baru* u Gautiera, prý. . .
Gloss:	. . . entertainment coordinator at The Julep *room* of a $\textbf{lit}_{\textbf{genitive}}$ $\textit{bar}_{\textit{genitive}}$ near Gautier, said. . .
Oracle output:	. . . programový koordinátor *v* Julep místnosti **osvětleném** *baru* u Gautiera, prý. . .
Gloss:	. . . entertainment coordinator $\textit{at}_{\textit{locative}}$ The Julep room a $\textbf{lit}_{\textbf{locative}}$ $\textit{bar}_{\textit{locative}}$ near Gautier, said. . .
Reference:	. . . koordinátor zábavy *v The Julep Room*, spoře *osvětleném baru* poblíž Gautier, řekl. . .

Example 4.4

4.1.2 Depfix Reference

From the previous observations, we assume that the constraint-based method achieves best results when combined with data, where reference sentences (or human post-edited sentences) are as close to the MT output as possible. We can also take another look at the issue: the less is the amount of unnecessary changes , or in our case changes which cannot be identified by our system (e.g. lexical error corrections), more precise should be the extraction of the training instances. For this reason, we have decided to examine training data, where we created "syntetic" post-editating of the MT output. These synthetic sentences are neither produced by a human translator or created by a human post-editing process. A suitable tool for this task might quite naturally be Depfix.

Reference	WrongForm1	WrongForm2	WrongForm3
Original	0.159	0.021	0.026
Depfix	0.009	0.003	0.003

Table 4.1: Overview of the portions of instances from WMT10 dataset marked as incorrect using different heuristic rules. We compare original dataset containing reference translations and dataset where reference translations were replaced by Depfix output.

The main focus of the Depfix post-editing tool is correcting morphological errors (aside from some frequent lexical errors, e.g. missing reflexive particles and too eagerly translated named entities) which usually results in post-edited sentences that are not that different from the MT output. Aside from that, the morphological changes (even though they are a result of an applied set of rules) made by Depfix are very similar to the post-editing changes we are trying to teach our statistical component.

Considering these observations, we have created a new datasets by applying Depfix on the available bilingual data and using Depfix output in place of reference sentences. We have then extracted the training instances in a similar way as with the genuine post-editing data. We have also run our Oracle classifier on the data with the resulting sentences matching the Depfix output most of the time.[5]

This method seems quite viable for the task of identifying incorrect instances and learning the right correction method because it should reflect at least the thought process behind creating the corrections rules. It should be also aplicable on various MT systems because, as Rosa (2014) has previously reported, Depfix was able to improve the quality of various systems to some degree. The main downside to this method is currently being limited only to the Czech language since the Depfix was created with the aim to post-edit English-Czech machine translation. Furthermore, it is questionable whether the system trained on these data can surpass the performance of the original post-editing system. This approach might become more interesting if a viable method of adapting the trained models to another language pairs is devised in the future.

4.1.3 Oracle Evaluation

In this section, we present a brief evaluation of the Oracle classifier to show what is the possible upper limit, that can be reached by our statistical components. This is important, because the resulting model cannot perform well with regard to our demands, if the extracted instances that are provided during training are already incorrectly classified.

We present a brief statistic about the percentage of the instances that were marked by each heuristic presented in Table 4.1. The statistic was computed over several different datasets presented in the previous chapter (Autodesk, HimL, WMT10, WMT16). We also provide similar statistics for the data created via Depfix. We can see that the portion of incorrect instances, i.e. the instances which will serve as our training data, is not very high. This is due to many sentences

[5]The only cases when the Oracle classifier did not correspond to the modification made by Depfix were when Depfix made one of its lexical corrections.

Reference	Evaluated	Changed	+	−	0	Precision	Impact
Post-edits	800	95	61	16	18	79.2%	7.6%
Depfix	800	75	47	3	25	94.0%	5.5%

Table 4.2: Results of the manual evaluation of the ideal system based on the WrongForm3 heuristic. Sentences were taken from HimL dataset. We compare the results on both original dataset containing human post-edited sentences and the dataset with Depfix reference sentences.

being considered correct by our heuristics (possibly because they contained either too many errors, or the errors were of a different nature than the ones we detect).

To get a brief idea, if our heuristic has a potential of providing valuable information for the classifier training, we decided to manually annotate a portion of the data from the HimL testset post-edited by our Oracle classifier. Because this is not a final evaluation of our system, the evaluation was made only by the author of this thesis. The evaluation has been done on the pairs of MT output, Oracle post-edited output, randomly shuffled to lessen the bias of the annotator. We also provided the source sentence to help the annotator decide whenever both outputs seemed grammatically correct but had a different meaning. We did not provide the annotator with reference sentences due to the nature of Oracle post-editor. For comparison, we also extracted a similar number of sentences from the same dataset, where the reference sentences were replaced by the Depfix output. This evaluation was done mainly for the purpose of development so only one annotator was involved.

In Table 4.2, we present results of the manual evaluation. We present number of sentences in the evaluated dataset, the number of sentences, that were actually modified by Oracle and the manually assigned label given to each modified sentence: better (+), worse (−) or indecisive (0). Indecisive annotations were usually a result of both translations being too incomprehensive or due to multiple corrections in the sentence, both improving and worsening. In addition, we have computed the precision (4.1) and impact (4.2) of Oracle in a same manner as Bojar et al. (2013a) did during Depfix evaluation in the Chimera MT system:

$$precision = \frac{better}{better + worse} \tag{4.1}$$

$$impact = \frac{better}{evaluated} \tag{4.2}$$

We can see from the results, that the final heuristic presented in this chapter, if we ignore the indecisive corrections, made mostly positive changes to the MT output (about four-fifths of the time) However, the impact is quite low, only slightly more than 7% of the sentences were modified by Oracle classifier. When compared to the results achieved on the data with Depfix reference sentences we can see that the precision grows even more and the overall impact drops a little lower. Therefore, we can assume that the presented heuristic can provide us with quite reliable training data. Sadly, due to the low impact, much larger volumes of post-edited data are needed to extract of reasonably big training dataset.

4.2 Feature Extraction

We have chosen the following strategy for designing the initial set of features: we extract as many distinct features based on the available node information and later we reduce the features by one or several methods of features selection, either manual or stochastic.

Our feature set has a hierarchical structure. For each training instance, we extract information about the node, its parent, the aligned source node and aligned source node's parent. For training purposes, we also extract information from the aligned reference node. Note, that we ignore information about the parent of the reference node, we use this information only for distinguishing the incorrect instances from the correct ones according to the previously described approach.

From each of these "*main*" nodes inside the instance, we extract information specific to them and to their close neighborhood (e.g. their parent, grandparent, preceding child, following child, preceding sibling, following sibling). We also extract information about the number of preceding and following children, direction of the edge coming from the node's parent and finally the Interset representation of the morphological POS tag. As a default, if a value is not defined we use an empty string instead. We are allowed to do that because we treat every feature as categorical, i.e. taking one of discrete values.

We have also tried extracting the lemmas but it resulted in a large growth of an already quite big feature space, so we initially abandoned the idea. In the future, it might be interesting to include at least a limited number of the most frequent lemmas in our feature set and observe how they can affect the performance of the classifiers.

This way, we extracted around 1500 initial features. Of course, this was a general feature set that needed to be further processed within every classification task. Also, the method by which we chose the initial features resulted in many of the features having zero variance. Those features we removed, giving us around 1000 features.

5. Model Training

In this chapter, we describe the process of developing the statistical post-editing models. We take a closer look at the task of model selection and parameter tuning, the feature selection methods we have experimented with and several methods of evaluation used during the tuning. We only present experiment results for the separate statistical models, evaluation of the whole MLFix system is presented in the next chapter.

In this chapter, we cover the following two classification tasks: the identification of incorrect instances (words from the MT output with incorrect surface form), addressed as *error detection* for short, and the prediction of the new morphological categories for the incorrect instances, referred to as *morphological prediction*.

Our development process can be separated into three stages: in the first stage, we have focused on basic comparison of different ML methods and choice of the most suitable candidate for each task. In the second stage we have tried to further increase the performance of the model based on the chosen ML method by experimenting with various methods for feature selection and in the third stage, for each dataset, we have searched for the best hyperparameters of the both the selected ML method and the feature filtering method.

5.1 Model Evaluation Methodology

We have decided to do word-level evaluation during the model development. More precisely, we have evaluated performance of models using the instances extracted by the process we have described in the previous chapter.

For both error detection and morphological prediction task, we have defined a baseline model for comparison. The baseline model basically represents a predictor which does not detect any errors (all instances are marked as correct) or keeps the original morphological categories for each instance.

Aside from a standard accuracy metric, we have also measured precision and recall of the models we trained to gain additional information about the model performance. We use standard definition of precision (5.1) and recall (5.2) computed using the following equations:

$$precision = \frac{TP}{TP + FP} \tag{5.1}$$

$$recall = \frac{TP}{TP + FN} \tag{5.2}$$

However, we have slightly altered the definitions of *true positives* (TP), *false positives* (FP) and *false negatives* (FN) with regard to the baseline model.

For error detection, each instance that was assigned the same value as the *true prediction* and a different value than the *baseline* is marked as TP. Eash instance that was assigned a value different from both the *true prediction* and the *baseline* is marked as FP and each instance with the *predicted value* equal to the *baseline* but different from the *true prediction* is marked as FN.

For morphological prediction, we use the same definition of TP, however, the definitions of FP and FN are altered in the following way (the *true prediction* does not match the *predicted value*):

- if the *baseline* value matches the *predicted value*, an instance is marked as FN,
- if the *baseline* value matches the *true prediction*, an instance is marked as FP,
- if the *baseline* does not match either one, an instance is marked as *wrong positive* (WP).

The WP is a special case which reflects the situation where predictor tries to predict new value (different from the original one) but fails and returns just another incorrect value.

We must also take into account that the error detection classifier is a general model, which is not designed to distinguish the types of morphological errors and the morphological predictor might be specialized only on a limited set of the morphological categories. Therefore, sometimes we want the morphological predictor to just leave the "incorrect" instances unchanged because it is simply incapable of predicting new values of the incorrect categories.

In the end, we have found the basic accuracy metric to be more informative for evaluating the morphological predictor, not only because in takes the WP instances into account but also due to the nature of our training data. We use only instances that are marked as incorrect by our heuristic which make precision/recall less relevant compared to accuracy. On the other hand, if we decided to expand our training data by "correct" instances, these metrics would become more useful.

5.2 Automatic Error Detection

As we have already pointed out, the task of identifying morphologically incorrect words in the text is more difficult than the task of assigning new morphological categories. It is not surprising that we have faced several issues during the development which we describe in more detail in the following sections.

5.2.1 Unbalanced Data Problem

In this task, we face the problem of binary classification, where we assign the value 0 to instances that we consider correct (they are not going to change) and the value 1 to instances that need to be corrected. The process of assigning these values to the extracted training instances was described in the previous chapter.

The *baseline classifier* for error detection has already achieved accuracy larger than 95% simply by marking all the instances as correct (class 0). However, we are more interested in maximizing the precision and recall of the class 1 predictions. As we have already pointed out, this is caused by a fact that only a small portion of our training instances is being marked as incorrect by our heuristic. This became a severe issue since most of the machine learning methods rely more or less on accuracy during the process of searching for the best hypothesis, however, it is the minority class, that is our target during the error classification.

There are several methods that can be used to solve this problem: e.g. creating synthetic training data by over-sampling instances with our minority class or under-sampling instances belonging to the dominating class (Batista et al., 2004), weighting of the training instances or modification of the cost function (Domingos, 1999). The manipulation with the training data (over and under-sampling) is the easiest method, however, by modifying the distribution of the classed in training data the classifier performance can drop when applied to data with real-world distribution. Still, the idea of modifying the distribution of our training data is worth considering.

We took the inspiration from the work of Jia et al. (2013) describing the classification of grammar errors in the texts written by a human. They also approach the task as a classification problem. During training, they filter their training corpus to only around 5% of sentences, because only those were ones that contained grammatical errors.

In our case, if we look at the results produced by Oracle classifier, we can see that only a small portion of the MT sentences has been actually modified. This means that a large number of the sentences were considered morphologically "correct".[1] These sentences are still part of our original training data, introducing a significant amount of training instances belonging to the majority class. Therefore, to balance our data, we have decided to use only training instances extracted only from the sentences where at least one word was marked as incorrect. This way, we were able to increase the portion of the minority class up to 10%. The training data is still unbalanced but to a much lower degree.

This less unbalanced dataset can be used in two ways: we can simply treat it like a downsampled data and use the trained models "globally" (on every MT sentence), or we can add another component for identifying these "incorrect" sentences and than apply our error detection model on them. In the scope of this thesis, we opt for the former approach.

5.2.2 Machine Learning Method Comparison

There are many machine learning methods that support binary classification, so we have decided to only compare a limited subset of the available methods that are implemented in the Scikit-Learn framework. For each classifier we tried several hyperparameter settings to observe the changes in the classifier behavior. However, we have made only a rough examination of hyperparameter configuration due to the high number of tested methods. At this stage, the goal was not to train the best possible classifier but eliminate those, that are not suitable for the task.

We have measured the performance of the classifiers on several datasets: WMT10 data translated by CU-Bojar system (Bojar et al., 2012a), dataset extracted from the lingea logfiles (HimL) translated by simple moses SMT system (Koehn et al., 2007) and WMT16 newstest dataset translated by the Chimera system. For each dataset, only instances from the "incorrect" sentences (sen-

[1] This is very likely not true but we can at least assume that they do not contain training instances marked as incorrect (as far as our heuristic goes).

Dataset	System	Origin	# Instances	# Instances (filt.)
WMT10	CU-Bojar	REF	23,470	6,033
HimL	Moses	PE	7,234	2,469
WMT16	CU-Chimera	REF	26,942	7,047
WMT10	CU-Bojar	Depfix	40,678	4,101
HimL	Moses	Depfix	10,491	1,114
WMT16	CU-Chimera	Depfix	42,021	1,897

Table 5.1: Summary of the size of the training data extracted from various datasets translated by different SMT systems. We present size before and after (filt.) removing the instances extracted from the "correct" sentences. *Origin* column indicates origin of the reference sentences: post-edited (PE), standard reference (REF) or created by Depfix.

tences containing at least one error) were used. Additionally, their counterpart was created by substituting the reference sentences with the Depfix output to additionally measure the performance on the "synthetic" data. The summary of the size of the used training data is in Table 5.1.

For the purposes of this coarse evaluation, we used each dataset separately for both training and testing of the classifiers, by performing one-against-the-rest 10-fold jack-knife sampling. Therefore, the results presented in this stage should be considered only as an in-domain performance for a specific MT system.

We have compared the following methods: logistic regression, ridge regression classifier, random forests, extremely randomized trees and support vector machines (SVM) classifier, all of them being implemented in the Scikit-Learn toolkit. In Figure 5.1, we can see the comparison of the performance of various classifiers based on the F1-measure metric. We can see that for the task of error identification, support vector machines with linear kernel might be the most suitable method, outperforming other methods in most of our datasets. The "baseline" performance is not spectacular, with score of less 0.3 for the normal data and a slightly better score (∼0.5) for the Depfix data. Therefore, we have chosen SVM based models for the following stages of model development.

5.2.3 Feature Selection

During model comparison, we have compared the performance on two initial feature sets: one that did not contain any information about the source sentence (∼680 initial features) and one with source sentence features (∼1360 initial features). Before training, the features with zero variance were removed, however, no other feature filtering has been performed. We have noticed that the additional information provided by the source sentence features significantly improves performance of the majority of the ML methods. Therefore, we have decided to use this feature set for feature selection method comparison.

Having selected SVM, we tried out several methods for feature selection and compared their influence on the classifier performance. During the comparison, we used a SVM model with fixed hyperparameters. We have compared the following methods for feature selection: KBest selection (with chi-squared scoring function), selection of the percentile of the features (based on the ANOVA F-test), selection based on lasso regularization and selection through models with

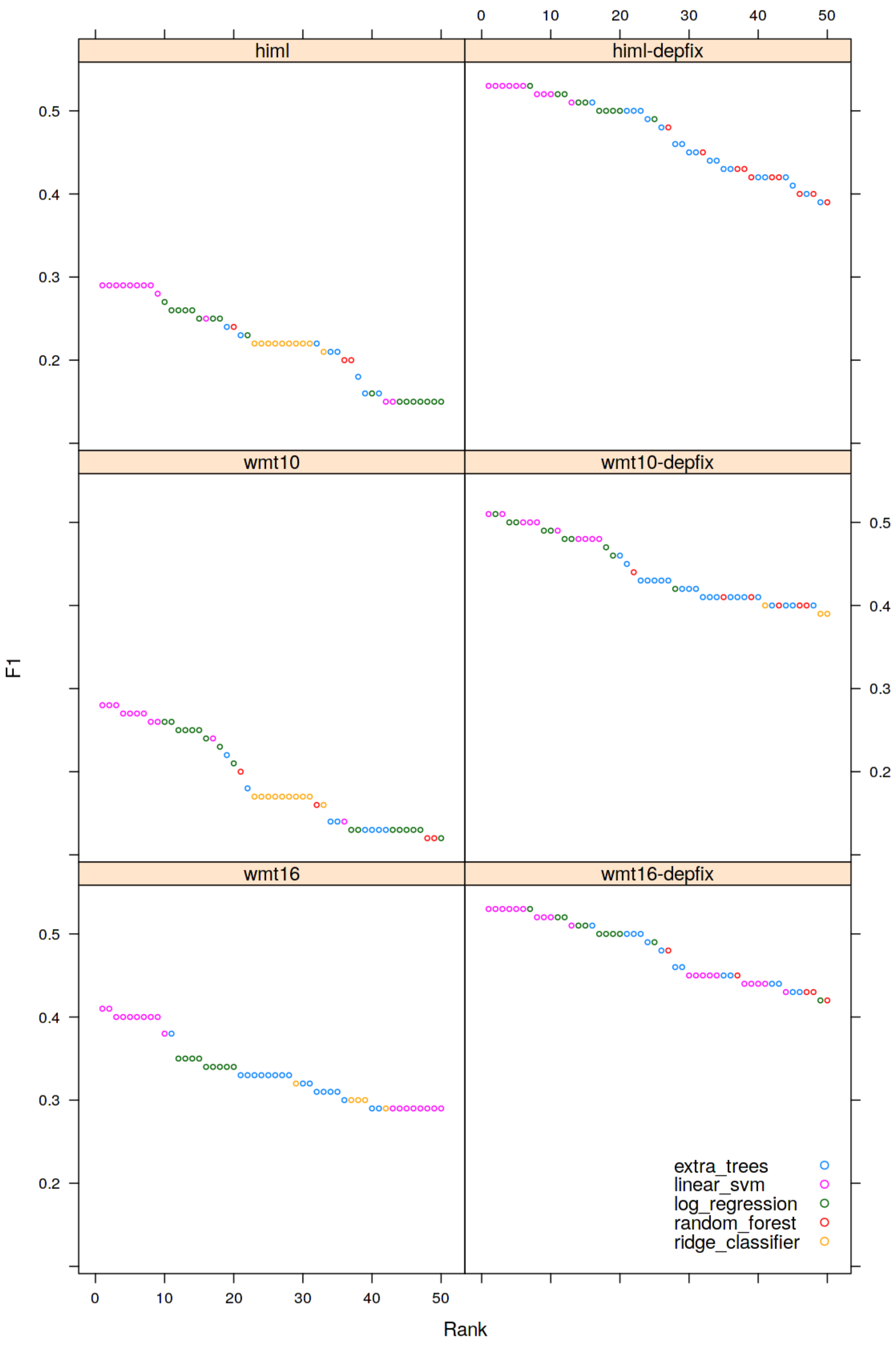

Figure 5.1: Overview of the classifier performance (error detection). We have tried several variations of the hyperparameters for each classifier. The classifiers are ordered from the best to the worst. Only top 50 results are shown for each dataset.

Dataset	System	Origin	Precision	Recall	F1
WMT10	CU-Bojar	REF	0.29	0.25	0.27
HimL	Moses	PE	0.28	0.32	0.30
WMT16	CU-Chimera	REF	0.40	0.44	0.42
WMT10	CU-Bojar	Depfix	0.51	0.55	0.53
HimL	Moses	Depfix	0.55	0.51	0.53
WMT16	CU-Chimera	Depfix	0.50	0.52	0.51

Table 5.2: Summary of the in-domain performance of the trained error detection models. The evaluation was performed by a jack-knife one-vs-rest classification on each dataset. *Origin* column indicates origin of the reference sentences: post-edited (PE), standard reference (REF) or created by Depfix.

feature importance scoring (svm, random forest). As far as importance scoring goes, we compared different model configurations and during features selection, only features with importance higher than the mean of the feature importance distribution were selected.

The results of feature selection method comparison are shown in Figure 5.2. We can see that most of the time the feature selection performed by either ANOVA percentile selector or SVM slightly improved the model performance. Therefore, we have decided to use these methods during the model tuning.

5.2.4 Model Summary

With the ML method and feature selection method chosen, we have proceeded to the development of the final model. We have trained several different models, one for each presented dataset, instead of combining the datasets and training one larger model. We have chosen this approach because it makes it easier to exclude specific models if needed (e.g. during final evaluation) and in the future, include additional models when more training data become available without the need to retrain the whole error detection component. The use of multiple models for error detection is described later in Chapter 6.

We present the summary of the trained models in Table 5.2. We can see, that the models trained on the Depfix data performed better than the ones trained on the original datasets. Unfortunately, we did not perform any cross-domain evaluation at this stage so we cannot say that the datasets with Depfix reference sentences provide better training instances for the error detection models. The better performance might be just a result of more consistent training data in the corresponding dataset.

The overall performance is still quite low, so there is still room for improvement. We think that introducing additional features (e.g. information from the t-layer) or increasing the size of the training data might help us improve these models in the future.

5.3 Automatic Morphology Prediction

The second classification task to predict correct morphological categories for the words that were marked as incorrect. Because we are using the Interset represen-

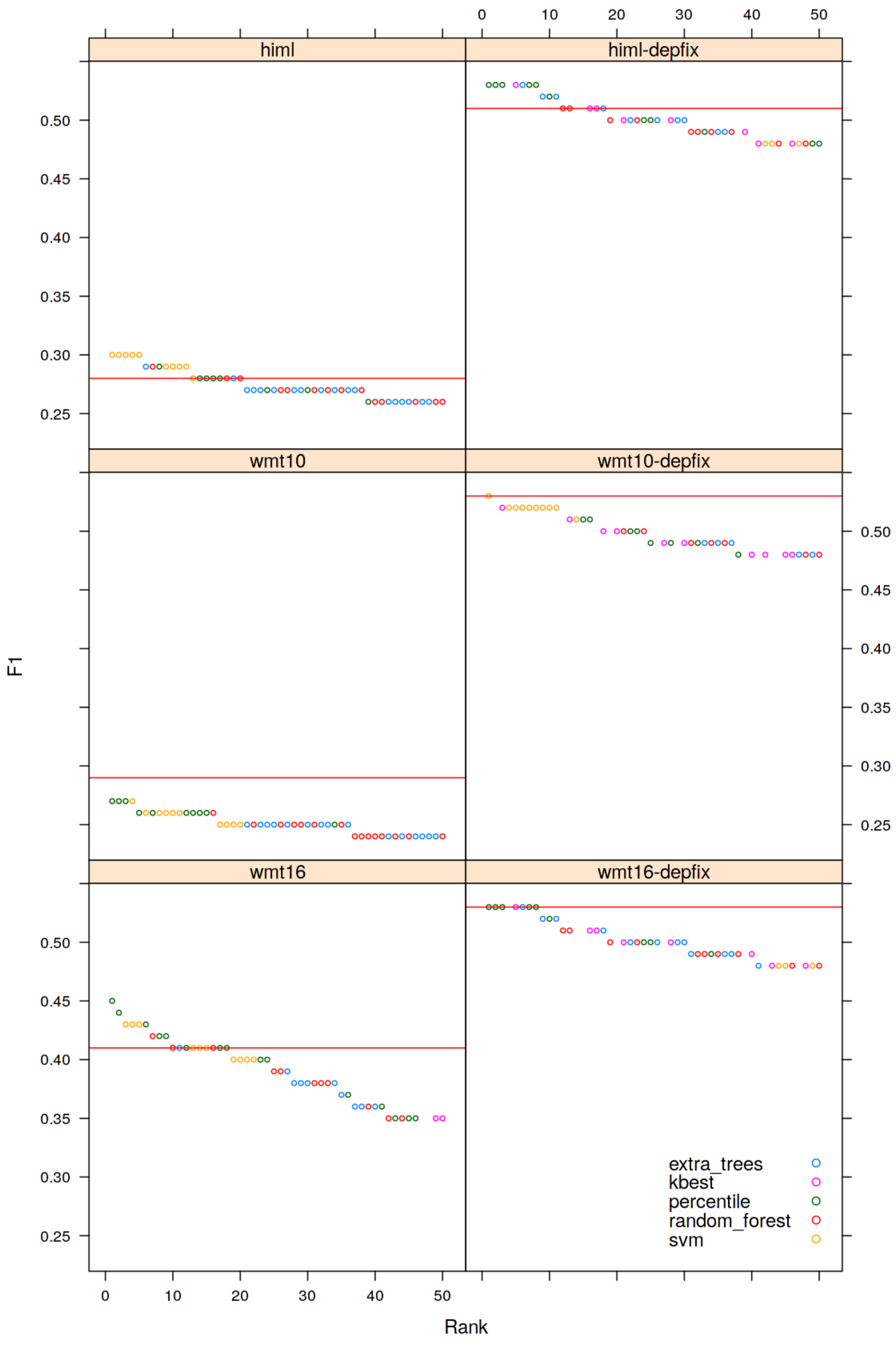

Figure 5.2: Overview of the performance of the SVM with linear kernel when combined with various feature selection methods. We have tried several variations of the hyperparameters for each method. The horizontal line marks the best performance without any feature selection method. The methods are ordered from the best to the worst. Only top 50 results are shown for each dataset.

POS	Frequency
noun	38%
adj	16%
adp	10%
verb	9%
adv	9%

Table 5.3: Change frequency of various POS classes in Czech.

tation of morphological features we have several possibilities how to handle this task such as:

1. predict each category separately,
2. concatenate the features and treat them as a single prediction target,
3. use the methods that support multitask classification,

With the first option, the biggest issue is determining the order in which the classifiers should be applied. Additionally, we have to decide if we also want to include the current node's morphological features into our model's feature set or use the newly predicted ones. The second approach eliminates this problem by predicting the values simultaneously. This, however considerably enlarges the set of predicted values increasing data sparsity. This can be a big problem as we have already shown that the amount of data available for the post-editing task (mainly the post-edited data) can be quite small. The third option combines the first two by training an estimator which handles multiple joint classification tasks, one for each morphological category. The Scikit-Learn toolkit provides several classifiers which support this option.

Before jumping straight into the developing of our classifier, it is important to examine what morphological changes are made in our training data, how frequently they are made and how can they affect the resulting surface form generation. For instance, we can predict new values of the "*punctype*" category, but it is very unlikely that it will affect the resulting wordform of any of the words classified as incorrect, because this category is related strictly to punctuation. There are of course less obvious examples and some categories, while being relevant in one target language can be pointless in another.

For this reason, we have made a frequency analysis of the changes encountered in our data, shown in Figure 5.3. We can see that most of the time, only the grammatical case was modified (more than 50% of the instances for the Depfix-based datasets and more than 40% of the instances for the genuine post-editing datasets). Other changes were a lot less frequent (less than 10% of the modified instances). We have also checked, the amount of modifications made for individual general POS classes. Table 5.3 summarizes the modification frequencies for each POS class. In conclusion, we have decided to focus on predicting the following categories: grammatical case, number, gender and animateness. These categories are relevant to the majority of the changed words, therefore, changes made by a classifier predicting these categories should be noticeable.

Dataset	System	Origin	# training instances
WMT10	CU-Bojar	REF	645
HimL	Moses	PE	338
WMT16	CU-Chimera	REF	722
WMT10	CU-Bojar	Depfix	210
HimL	Moses	Depfix	72
WMT16	CU-Chimera	Depfix	99

Table 5.4: Summary of the size of the training data extracted from various datasets translated by different SMT systems. *Origin* column indicates origin of the reference sentences: post-edited (PE), standard reference (REF) or created by Depfix from the given MT output.

5.3.1 Machine Learning Method Comparison

We have decided to train four different types of model: one predicting case only (C), one predicting case and number (CN), one predicting case, number and gender (CNG) and one predicting case, number, gender and animateness (CNGA). We have used the same datasets as in the previous task, however, this time we have extracted only feature vectors of the instances that were marked as incorrect in our training data. Since our predictors classify only the incorrect instances, training them on the whole dataset would only create unnecessary bias. On the other hand, this has made the training sets quite small (containing only a few hundreds of examples at most). The summary of the training data is in Table 5.4. We have decided to train a separate classifier for each dataset instead of combining the data together, because we can simply combine the models instead (e.g. via majority vote, best prediction etc.). This allows us to evaluate the combined model on a test set of our choice. We can also leave out the model trained (using jack-knife) on that particular test set, to see the applicability across test sets or domains.

Again, we had to decide which ML method should we use for this task. We have examined similar set of classifiers with similar hyperparameters as in the error classification task to get a rough idea about their capabilities. We have considered using the F-measure again, however, due to the nature of the training data (all instances are classified), the model accuracy metric seemed more informative. A rough comparison was made with the case classifier only. Adding additional targets to the classifiers naturally lowers their overall accuracy, however, their performance have been similar with respect to each other The results are shown in Figure 5.4. We can see, that even without any sophisticated parameter tuning or feature filtering, the classifiers perform quite well. We can also notice that in most of the cases, the ensemble methods (mainly extremely randomized trees) performed slightly better than the rest of the examined ML methods. Therefore, we have decided to pick this method for further experiments.

5.3.2 Feature Selection

We have decided to perform additional feature selection with models trained on the standard HimL dataset and WMT10 dataset because there is still a reasonable room for an improvement. Again, we have tried two initial feature sets, one

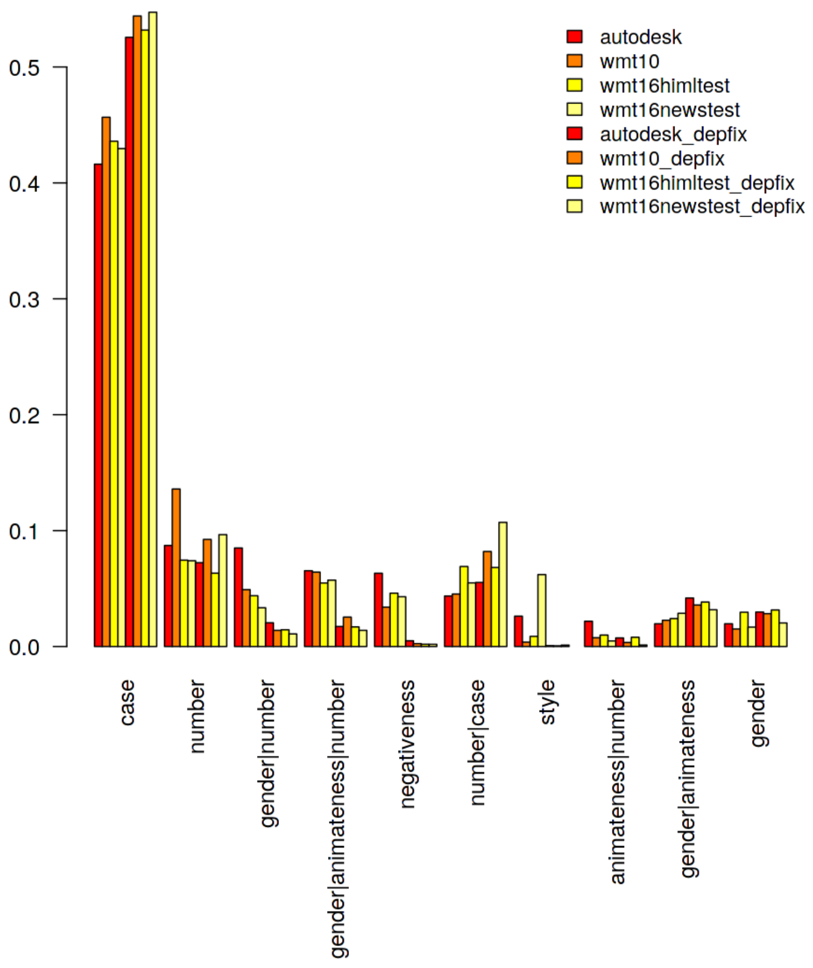

Figure 5.3: Frequency of the most changed Interset categories, grouped by a datasets. Categories containing "|" symbol (e.g. gender|number) represent changes made simultaneously.

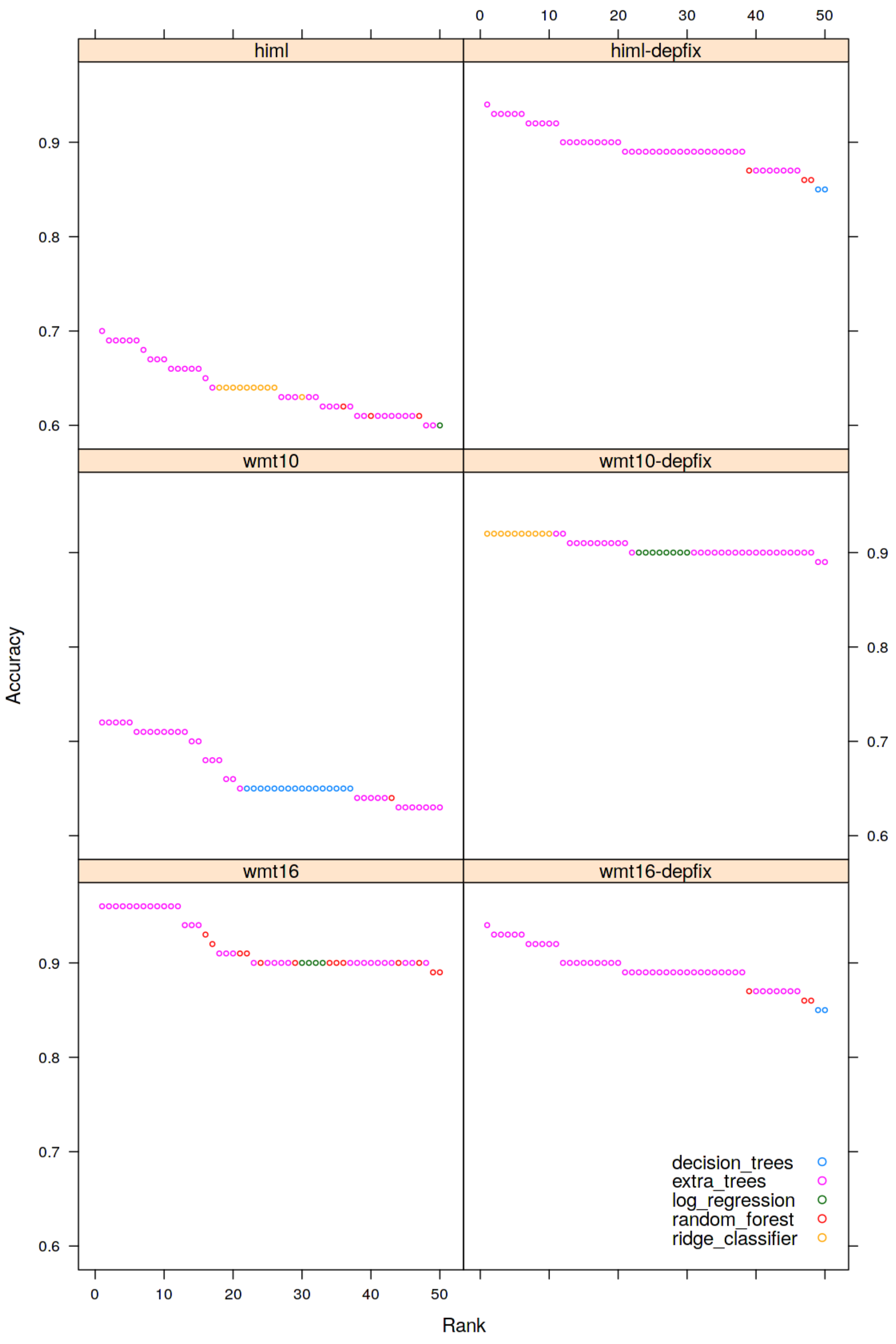

Figure 5.4: Overview of the classifier performance (category prediction). We have tried several variations of the hyperparameters for each classifier. The classifiers are ordered from the best to the worst. Only top 50 results are shown for each dataset.

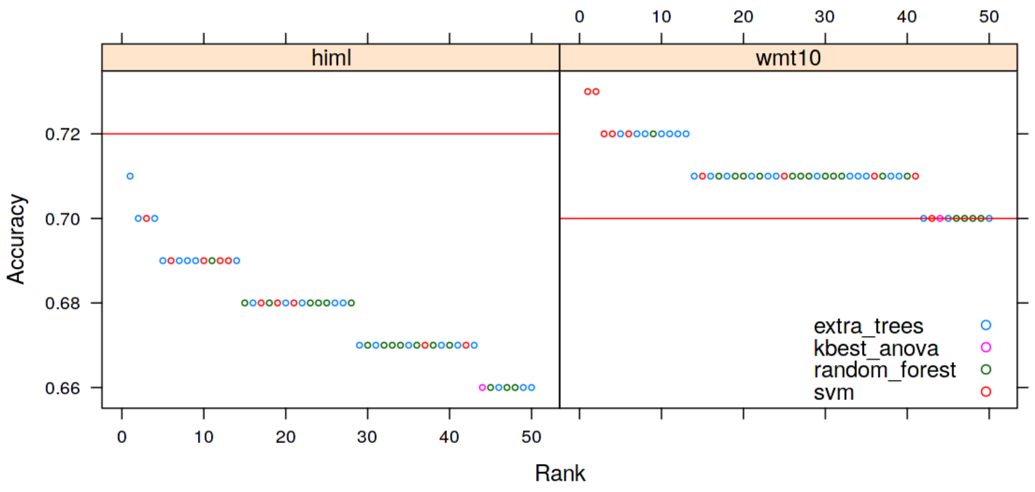

Figure 5.5: Overview of random forest performance when combined with various feature selection methods. The methods are ordered from the best to the worst. The horizontal line indicates the best performance without any feature selection method. Only top 50 results are shown for both examined datasets.

using the source side features and one without them. We have not noticed any significant difference in performance between the models trained on these two initial feature sets so we decided to use the larger one and leave the feature selection to the automatic feature selection method. Additionally, probably due to the nature of the classifier, we have noticed another slight improvement by adding the *source lemma* (both of the aligned node and its parent) features, so we have also included them to the initial feature set.[2]

We have compared several methods of feature selection: KBest selection (with chi-squared scoring function), selection based on lasso regularization and selection through models (svm, random forest). Again, we have done a rough comparison of these methods by trying out several hyperparameter configurations. We tested them on a model with fixed parameters. The result summary is in Figure 5.5. We can see that regarding HimL dataset, feature selection did not have any positive effect on the resulting performance. On the other hand, we have decided to use the SVM-based feature selection for the WMT10 dataset model training.

5.3.3 Model Summary

In the end, we have trained six different models, one for the each dataset presented in the rough comparison. Aside from the case classifier, we have also compared performance of the chosen multitask classifiers: the case-number (CN), the case-number-gender (CNG) and the case-number-gender-animateness (CNGA) classifier. These classifiers were trained using the same ML methods, each one was tuned separately. The summary of the final in-domain performance is in Table 5.5.

[2]We have not included the *MT lemma* feature, because we wanted to try using models trained on Czech for German post-editing and this feature is too language-specific for that purpose.

Dataset	System	Origin	Case (Base)	CN (Base)	CNG (Base)	CNGA (Base)
WMT10	CU-Bojar	REF	73% (35%)	53% (14%)	40% (7%)	37% (7%)
HimL	Moses	PE	72% (34%)	50% (11%)	41% (5%)	41% (4%)
WMT16	CU-Chimera	REF	95% (50%)	45% (21%)	34% (10%)	32% (10%)
WMT10	CU-Bojar	Depfix	92% (45%)	83% (22%)	69% (19%)	69% (19%)
HimL	Moses	Depfix	92% (47%)	78% (26%)	73% (23%)	71% (23%)
WMT16	CU-Chimera	Depfix	93% (54%)	64% (16%)	56% (11%)	53% (9%)

Table 5.5: Summary of the in-domain accuracy of the trained morphological prediction models. The evaluation was performed by jack-knife one-vs-rest classification of the each dataset. Performance of the baseline (Base) classifier is presented for comparison. *Origin* column indicates origin of the reference sentences: post-edited (PE), standard reference (REF) or created by Depfix.

We can see that by including additional target for the multitask classification the accuracy of the trained model becomes naturally lower. However, we must take into account that we have only classified the predicted values as correct or incorrect and we have not distinguished partially correct predictions. Brief overview of the categories predicted by the CNGA classifier has shown us that usually a predictor misclassifies only one or two of the target categories. Another thing to keep in mind is the fact, that even though the predicted categories might not be equal to the gold standard, the generated surface form might still match the reference form due to the morphological ambiguities of the language. For these reasons, we decided to use all these models during the system evaluation.

6. System Evaluation

In this chapter, we present the results of the final MLFix system evaluation. We describe additional datasets we used during the evaluation and different configurations of MLFix we compared. Furthermore, we present evaluation of the individual MLFix components. We also present a comparison with the Depfix system. We performed both automatic and manual evaluation.

6.1 Automatic Evaluation

During automatic evaluation, we used BLEU (Papineni et al., 2002) translation quality metric based on measuring the n-gram difference between the MT output and the reference translation. Even though it has its limitations, it is the most widely used evaluation metric at the moment and it is considered a standard metric for automatic evaluation. We relied mainly on the BLEU scoring metric, however, we also need to mention Translation Edit Rate (TER) (Snover et al., 2006) which we used during system development because it measures the quality of translation based on the amount of corrections needed to match the reference translation. Lowering the amount of work needed to post-edit the MT output is one of the aims of MLFix system. Furthermore, the metric has been proved to provide reasonable correlation with a human judgement.

We evaluted MLFix on the following datasets: Autodesk, WMT10 (Callison-Burch et al., 2010), WMT16 (Bojar et al., 2016), and HimL. These datasets were translated with various SMT systems.

Because the domains of the training datasets differ to a various degree, we decided to combine the final models instead of evaluating them separately. The methods for combining the output of multiple classifiers differ with each MLFix component and are described in more detail in the following sections. Naturally, for each dataset we exclude the models trained on the corresponding dataset from the evaluation.

6.1.1 Morphology Prediction Evaluation

We performed the evaluation of the morphological prediction module first to determine which combined model to use during the error detection evaluation. We compared several model combinations: *Case* models only, *CN* models only, *CNG* models only, *CNGA* models only and a combination of all available models (*Comb*). Each combined model runs all classifiers separately and stores the probabilities of their predictions. Using these predictions, candidate tags are generated and scored by these probabilities. If a tag is predicted by multiple classifiers, the scores are summed together. The tag with the highest score is then chosen as the result of the combined model.

We present the final results of the morphological prediction module evaluation in Table 6.1. We compare BLEU scores of each model combination and their improvement over the baseline scores. The values are multiplied by 100 for easier reading. We can see that the CNG and CNGA models had the largest impact most of the time. Since the difference between the CNG and CNGA performance

Dataset	System	Oracle	Base	Case	CN	CNG	CNGA	Comb
Autodesk	NA	49.20	47.82	47.89 (+0.07)	47.91 (+0.09)	**48.22 (+0.40)**	48.21 (+0.39)	47.90 (+0.08)
HimL	Moses	23.33	20.66	21.14 (+0.48)	20.97 (+0.31)	21.36 (+0.70)	**21.49 (+0.83)**	21.03 (+0.37)
WMT10	CU Bojar	16.70	15.66	15.84 (+0.18)	15.82 (+0.16)	**15.95 (+0.29)**	**15.95 (+0.29)**	15.82 (+0.16)
WMT16	UEDIN NMT	27.31	26.31	26.43 (+0.12)	26.48 (+0.17)	26.47 (+0.16)	**26.50 (+0.19)**	26.44 (+0.13)
	CU Chimera	23.13	21.72	21.94 (+0.22)	21.99 (+0.27)	22.02 (+0.30)	**22.05 (+0.33)**	22.02 (+0.30)
Autodek-D	NA	98.32	96.93	98.12 (+1.19)	98.09 (+1.16)	**98.17 (+1.24)**	98.16 (+1.23)	98.08 (+1.15)
HimL-D	Moses	96.05	94.23	95.23 (+1.00)	95.04 (+0.81)	95.41 (+1.18)	**95.42 (+1.19)**	94.98 (+0.75)
WMT10-D	CU Bojar	95.72	93.42	95.03 (+1.61)	94.99 (+1.57)	**95.10 (+1.68)**	95.07 (+1.65)	94.96 (+1.54)
WMT16-D	CU Chimera	97.51	96.56	97.22 (+0.66)	97.23 (+0.67)	97.26 (+0.70)	**97.27 (+0.71)**	97.20 (+0.64)

Table 6.1: Automatic evaluation of the morphological prediction module using BLEU and the relative improvement over the baseline MT output. Values are multiplied by 100 for easier reading. The performance of the Oracle classifier is also provided for comparison. Datasets with the *-D* suffix have Depfix output in place of reference sentences. The best model for each dataset is printed in bold.

is very small we chose CNG model for the final evaluation because its models performed better during the model training.

Surprisingly, the performance of the Comb models was lower than we anticipated. This might be caused by the fact that these models use basically four times more classifiers and no weighting is used when combining their results. Therefore, the final results of the combined models may be biased towards the simpler models working with a lower number of possible outcomes and thus assigning their predictions higher scores. Still, a more thorough investigation is needed in the future.

6.1.2 Error Detection Evaluation

We performed error detection evaluation in a similar way. We used a combination of binary classifiers in our prediction module. However, we chose a different strategy for combining the output of the classifiers. We decided to use voting scheme to determine the final prediction. In the end, we compared three basic methods: *Majority* vote, *AtLeastOne* method, and *Average* prediction method.[1]

The *Majority* vote basically chooses the class that was marked by the majority of classifiers. We think that this way can help us compensate for the low precision of some of the classifiers we trained. Because the classifiers were trained on different datasets, there should be some level of complementarity between them. Still, we did not perform any more thorough analysis to support this assumption. Potentially, this method can lead to a lower recall due to classifier combination. We assume that combining several low recall classifiers can further bias the module toward the majority class when using this method.

The *AtLeastOne* method classifies the word as incorrect if at least one classifier marked it as incorrect. This method should counter the problem of combining several low recall classifiers to some extent, however, we expect the precision to drop rapidly when adding more classifiers, especially if their own precision is low to begin with.

Because we are using classification models that support weighting of the predicted classes, we also considered the *Average* voting scheme. If the combined classifier provide us not only with a predicted class but also some sort of confi-

[1]Surely, there are other voting strategies available but they were not investigated in the scope of this thesis.

Dataset	System	Oracle	Base	Majority	AtLeastOne	Average
Autodesk	NA	49.20	47.82	47.89 (+0.06)	**48.47 (+0.65)**	47.89 (+0.06)
HimL	Moses	23.33	20.66	20.69 (+0.02)	**22.08 (+1.41)**	20.69 (+0.02)
WMT10	CU Bojar	16.70	15.66	15.84 (+0.18)	**16.30 (+0.64)**	15.77 (+011)
WMT16	UEDIN NMT	27.31	26.31	26.31 (0)	**26.49 (+0.18)**	26.31 (0)
	CU Chimera	23.13	21.72	21.79 (+0.07)	**22.01 (+0.28)**	21.79 (+0.07)
Autodek-D	NA	98.32	96.93	97.99 (+1.05)	**97.99 (+1.05)**	98.31 (+1.37)
HimL-D	Moses	96.05	94.23	94.62 (+0.39)	**96.04 (+1.81)**	94.62 (+0.39)
WMT10-D	CU Bojar	95.72	93.42	94.76 (+1.33)	**95.61 (+2.18)**	94.80 (+1.37)
WMT16-D	CU Chimera	97.51	96.56	97.04 (+0.48)	**97.47 (+0.9)**	97.05 (+0.49)

Table 6.2: Automatic evaluation of the error detection module using different voting methods to interpret output of multiple models using BLEU. Values are multiplied by 100 for easier reading. For comparison, the performance of Oracle classifier is also provided. Values in brackets indicate the difference between the method and the baseline (Base) MT output. Datasets with the *-D* suffix have Depfix output in place of reference sentences.

dence value (e.g. probability of the prediction correctness), we can average these values and mark instance as incorrect if the averaged value exceeds a certain threshold. This method is similar to the *Majority* scheme with the difference that it can choose the result predicted by the minority only when their overall confidence of the prediction is large enough.

We present the comparison of these three methods in Table 6.2. Morphological values of the "incorrect" words were predicted by the Oracle morphological predictor. We can notice that the AtLeastOne method achieved significantly better BLEU scores than the other two. This was expected and in this case it does not necessarily mean that the resulting sentences improved from the human evaluation viewpoint, because the situation is similar to the heuristic selection we presented in the task definition. Even though many wordforms were changed by this method to reflect the wordforms in the reference sentence, it might have lowered the overall fluency of the translated text. Still, we think that it is worth exploring this method further.

As for the other two methods, the Average voting method seems to perform slightly better than the Majority voting but the difference is not significant.

6.2 System-wide Evaluation

In this section, we present the final evaluation of the MLFix system. Based on the evaluation of the separate components, we chose the CNG model combination for morphological prediction and we decided to compare both AtLeastOne and Average voting method for error detection. We also made a comparison with the Depfix system. The results are summarized in Table 6.3. The number of changed sentences by each system is shown in Table 6.4.

The results confirm that AtLeastOne voting method, while having large impact, does not work quite well with Czech morphological prediction module. We cannot tell how many of the marked instances were actually correct, nevertheless, the morphology correction had probably hard time predicting correct morphological categories for these instances. On the other hand, we can see that MLFix

Dataset	System	Base	ALO-CNG	Avg-CNG	Depfix
Autodesk	NA	47.82	44.94 (-2.87)	**47.89 (+0.06)**	47.63 (-0.19)
HimL	Moses	20.66	18.91 (-1.75)	20.69 (+0.02)	**21.02 (+0.35)**
WMT10	CU Bojar	15.66	14.61 (-1.04)	15.76 (+0.10)	**15.91 (+0.25)**
WMT16	UEDIN NMT	26.31	25.75 (-0.55)	**26.49 (+0.18)**	26.15 (-0.15)
	CU Chimera	21.72	21.44 (0.27)	**21.79 (+0.07)**	21.75 (+0.02)

Table 6.3: Final evaluation of MLFix using BLEU. Values are multiplied by 100 for easier reading. AtLeastOne (ALO-CNG) and Average (Avg-CNG) voting methods were compared. The performance of Depfix is shown for comparison.

Dataset	System	ALO-CNG	Avg-CNG	Depfix	Sent.
Autodesk	NA	20,104	7,203	11,415	42,259
HimL	Moses	473	30	261	800
WMT10	CU Bojar	1,441	202	973	2,489
WMT16	UEDIN NMT	668	86	418	2,999
	CU Chimera	713	122	624	2,999

Table 6.4: Number of sentences changed by different systems. Total number of sentences in each dataset (*Sent.*) is listed for reference.

with the Average voting method was able to improve every tested MT output. However, the improvement was usually quite small, always smaller when compared to Depfix. Interestingly, Depfix was not quite able to improve the output of the neural network machine translation system (UEDIN NMT) in terms of BLEU, while MLFix achieved positive score improvement. It might be only a coincidence and it needs to be investigated more thoroughly in the future. Depfix results on the CU-Chimera dataset should be taken with a grain of salt since Depfix is already part of the Chimera SMT system. Depfix also had a problem with Autodesk dataset but we think this might be caused by the specific domain of the dataset (user documentation).

Even though the changes were positive according to the automatic evaluation, the impact of MLFix was very low. If we compare the number of sentences changed by MLFix and Depfix, we can see that Avg-CNG MLFix modified five to ten times fewer sentences depending on the dataset. We think that this might be mainly due to the poor performance of the error detection classifiers in our error detection module. Furthermore, Depfix covers wider range of corrections which also reflects in a larger overall impact.

6.3 Manual Evaluation

For manual evaluation, we used the HimL dataset translated by Moses and the WMT16 CU Chimera dataset. They were post-edited by the Avg-CNG MLFix configuration. We used two independent annotators, both were presented with pairs of MT output and MLFix output which were randomly shuffled. The source sentence and reference translation were also provided for each pair. In the end, a total of 126 changed sentences were evaluated. The results are shown in Table 6.5.

The results show that MLFix was able to improve around four-fifths of the modified sentences. This result is quite pleasant, but it still needs to be confirmed

	Evaluated	Changed	+	−	0	Precision	Impact
HimL-A	800	26	17	5	4	77.2%	2.1%
WMT16-A	2,999	100	73	21	6	77.6%	2.4%
HimL-B	800	26	5	5	16	50%	0.6%
WMT16-B	2,999	100	71	12	17	85.5%	2.3%
Total	7,598	252	166	43	43	79.4%	2.1%

Table 6.5: Results of manual evaluation of the best MLFix configuration (Avg-CNG). Annotators A and B are distinguished by a suffix for each dataset.

A/B	+	−	0
+	64	7	5
−	5	13	0
0	21	5	6

Table 6.6: Matrix containing inter-annotator agreement of MLFix manual evaluation.

in the future by a larger scale manual evaluation. As we expected, the overall impact of MLFix is quite low, most likely due to the beforementioned low recall of error detection classifiers in the error detection module. This module therefore requires further improvement in the future.

Both annotators evaluated the same set of 126 sentences, their inter-annotator agreement is shown Table 6.6. The inter-annotator agreement reached 62% (87%, if we disregard the indefinite changes), therefore, the results of manual evaluation cannot be completely relied on this time.

7. English-German

In this chapter, we describe changes we made to the English-Czech MLFix pipeline to be able to apply the system to the English-German SMT outputs. We summarize the data available for the model training and evaluate the system in a similar way we did with the English-Czech pipeline.

7.1 Processing Pipeline Modifications

As we already pointed out, we focused on making the processing pipeline as independent on the target language as possible. However, we still had to replace some of the tools used during the Czech analysis to be able to correctly process German sentences.

Again, we used the Treex framework as a backbone of the processing pipeline and necessary 3rd party tools were implemented into the framework via wrappers. After the sentences are read in parallel, they are processed separately, English sentences following the same scenario as in the English-Czech pipeline. German is tokenized, this time by a set of regex rules inspired by the Tiger corpus (Brants et al., 2004) with main focus on abbreviations, ordinal numbers and compounds connected by hyphens.

Next, lemmatization and morphological POS tagging is performed by a Mate tools[1] toolkit. The tagger is using CoNLL2009 (Hajič et al., 2009) tagset. We also considered using Stanford POS Tagger (Toutanova and Manning, 2000)[2], however, the tagset it uses contains only coarse tags with little morphological information. To convert the CoNLL2009 tags to Interset, we use a decoder which was already available at the time of our research.

For the word alignment, we use GIZA++ again. Similar to English-Czech, we produce one-to-one word alignment between the source sentences and the MT output via intersection symmetrization. The alignment model was trained on the European Parliament Parallel corpus (Koehn, 2005)[3] (Europarl) containing nearly two million sentences. During training, we create the MT-REF and SRC-REF alignments in a same fashion we did in the original pipeline.

The dependency structure of the MT output is created by projecting the English dependency structures on the MT sentences. When we process the reference sentences during training, we use a graph-based parser implementation (Bohnet, 2010) which is also a part of the Mate tools toolkit. We stop the analysis at the a-layer, but again, further analysis of the sentences at the t-layer to gain additional features for extraction might be helpful in the future.

We reused the statistical component used in the English-Czech pipeline, because it was designed to be language independent (with exception of the statistical models). For wordform generation we use Flect morphology generation tool mentioned earlier. We trained the generator on a small fraction (around one hundred

[1]`http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/matetools.en.html`

[2]`http://nlp.stanford.edu/software/tagger.shtml`

[3]http://www.statmt.org/europarl/

Reference	Evaluated	Changed	+	−	0	Precision	Impact
Post-edits	800	77	20	39	18	33.9%	2.5%

Table 7.1: Results of the manual evaluation of the ideal fixing module based on the same heuristic that was used in the English-Czech pipeline (WrongFrom3). Sentences were taken from HimL dataset. For wordform generation, a statistical component was used impacting the overall performance.

thousand sentences) of the Europarl corpus. The tool is trained on a set of features based on a combination of lemma+Interset producing the inflected word. The feature set was copied from the Dutch feature set so it might not contain all the useful features. The accuracy of the inflection model measured on a separate test set was around 94.5%. However, when we briefly examined the sentences produced via Oracle, we noticed that much larger amount of words was flected incorrectly.

It is not in the scope of this thesis but it is a future goal to replace the current inflection model with a better solution, either stochastic or rule-based.

7.2 Data Analysis

We were able to collect only a smaller variety of data for English-German compared to English-Czech mostly due to some datasets we mentioned earlier simply not being available for this language pair. Still, we were able to gather the following datasets: WMT16, HimL and Autodesk. Note that in case of Autodesk dataset, the size of English-German corpus is about three times bigger than the size of English-Czech (around 120k sentences). For this reason, we decided to include Autodesk dataset into our training data even though it covers a quite specific domain.

When extracting the training instances for the model training we followed same scenario as before using the same heuristic (WrongForm3) to identify “incorrect” wordforms. We also used the Oracle classifier to gather information about possible improvements this heuristic can bring when applied to German.

We also performed quick manual evaluation of the output produced by the Oracle classifier by a non-native German speaker. The evaluation was performed on the HimL dataset. Due to the limited resources we used only a single annotator for this evaluation task. The evaluator, not being familiar with the MLFix system, was presented with a set of instances containing the following: randomly shuffled MT output and Oracle output, the source English sentence and the German reference translation. The results of the evaluation are shown in Table 7.1. We decided to only correct morphological categories by Oracle leaving the surface form generation to Flect module because we wanted to see the best possible outcome that can be achieved by the statistical fixing components. In the future the Oracle evaluation with “Oracle” surface form generation might also be a valuable source of information. This choice resulted in worse performance of the Oracle classifier when compared to Czech Oracle. Therefore, we still chose to use the same heuristic for marking incorrect instances in the training data, because it probably was not because of the mark method that the Oracle performance dropped.

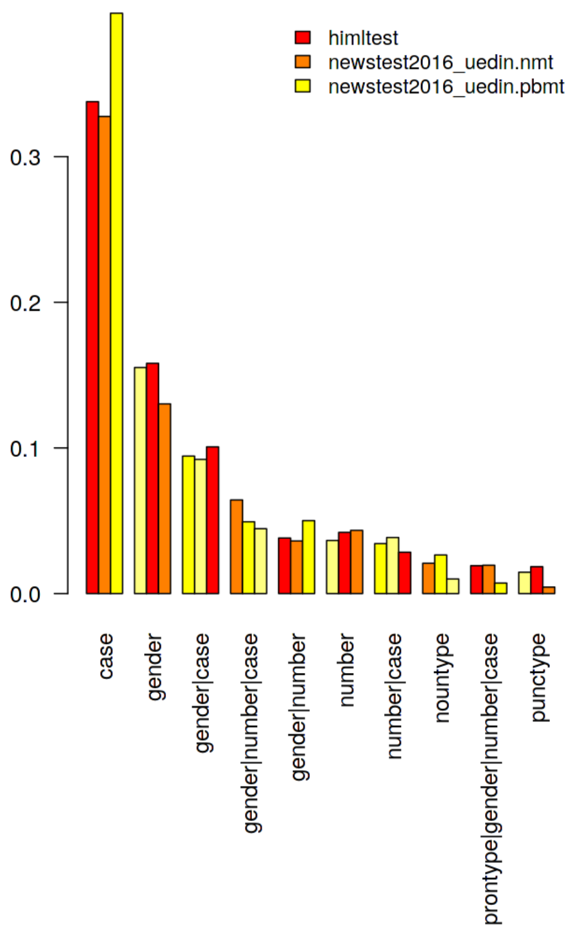

Figure 7.1: Frequency of the most changed Interset categories in German data, grouped by a datasets. Categories containing "|" symbol (e.g. gender|number) represent changes made simultaneously.

After the evaluation of the data extraction method we analyzed the extracted instances and compared them to the information we gathered during Czech data analysis. Figure 7.1 shows the frequency of the changed Interset categories. Again, case was the most changed category among our data followed by gender and number, either as a standalone change or as a part of a clustered change. This led us to training the morphological prediction models in a same manner as in the English-Czech pipeline dropping the animateness (*CNGA*) models this time. To make sure that we can make similar assumptions during the model development we also checked the distribution of changes made per individual POS classes. The summary is shown in Table 7.2. Even though the distribution is slightly different, nouns and adjectives are still the most changed words which further supports our decision for training Case, CN and CNG models.

POS	Frequency
noun	35%
adj	25%
punc	10%
adp	8%
conj	7%

Table 7.2: Change frequency of various POS classes in German.

Dataset	System	# Instances	# Instances (filt.)	# Incorrect
HimL	Moses	12,067	2,729	369
WMT16	UEDIN-NMT	22,353	3,340	344
WMT16	UEDIN-PBMT	21,394	3,727	412
Autodesk	–	1,263,750	124,698	17,268

Table 7.3: Summary of the size of the training data extracted from each dataset. We present size before and after (filt.) removing the instances extracted from the "correct" sentences. The size of datasets for category predictor training is presented in the *Incorrect* column.

7.3 Model Development

We skipped the process of choosing proper ML and feature selection method assuming that the ones used for training models for English-Czech pipeline will also be sufficient for German. We only searched for the best combination of hyperparameters during the model development process. We used both source and MT features with addition of the source language lemmas to create the initial feature set. We present a summary of the available training data for both error detection and morphological prediction in Table 7.3. Again, we can see, that the training data are quite small with exception of the Autodesk dataset. That is also the main motivation behind including it in our training data.

The summary of the final error detection models is in Table 7.4. Surprisingly, while having quite similar precision to the Czech models they achieved much better recall overall. At this moment, we cannot say if it is caused by a ML method choice or a suitable initial feature set, however, since there is still room for improvement, we consider investigating the issue further in the future.

In Table 7.5, we present performance of the final German mophology prediction models. The results are not very different from the Czech models with HimL model being slightly better in this case, possibly due to a lesser amount of possible values of the case category. Surprisingly, the accuracy of a model trained on the Autodesk dataset drops only a little with increasing complexity suggesting that increasing the amount of training data can significantly improve the overall accuracy of the resulting model.

7.4 Evaluation

We followed the same procedure during German MLFix evaluation as for Czech. We compared several configurations of the morphological prediction module and error detection module separately first, using Oracle to substitute the other mod-

Dataset	System	Precision	Recall	F1
HimL	Moses	0.39	0.56	0.46
WMT16	UEDIN-NMT	0.28	0.43	0.34
WMT16	UEDIN-NMT	0.29	0.50	0.37
Autodesk	–	0.41	0.77	0.53

Table 7.4: Summary of the in-domain performance of the final German error detection models.

Dataset	System	Case(Base)	CN(Base)	CNG(Base)
HimL	Moses	84%(29%)	79%(13%)	70%(5%)
WMT16	UEDIN-NMT	57%(46%)	48%(27%)	34%(22%)
WMT16	UEDIN-PBMT	57%(38%)	47%(21%)	35%(18%)
Autodesk	–	96%(28%)	95%(17%)	93%(8%)

Table 7.5: Summary of the in-domain performance of the final German category prediction models and its comparison with the baseline predictor.

ule, then we evaluated the whole MLFix system. We compared similar configurations we used in Czech MLFix: *Case*, *CN*, *CNG* and *Combined* model configuration for morphological prediction and *Majority*, *AtLeastOne* and *Average* voting scheme for error detection. For the whole system evaluation, we picked the two most promising configurations and compared their performance. We also measured a performance of the best Czech configuration (using Czech models) when applied to German. We used BLEU scoring metric during automatic evaluation. The evaluation was performed on the following datasets: Autodesk, HimL Lingea logs and WMT16. For each evaluated dataset, the models trained on the corresponding dataset were excluded.

We present the results of morphological prediction module evaluation in Table 7.6. Once again, CNG configuration proved to be the reliable choice, having the best score on each dataset. However, the overall improvement in BLEU score is much lower when compared with Czech version of MLFix. This is little surprising because the individual performance of the morphological prediction models measured during training was fairly equal and in some cases even better than the one of the Czech models. It is possible that this might be either a result of lower diversity in our data (most of the data belongs to Autodesk dataset) or a consequence of the poor performance of the inflection module.

In Table 7.7, we present the results of the evaluation of the error detection module. We can see that the module was performing quite poorly, not bringing any improvement to any dataset at all. Due to the nature of the error detection module evaluation (new morphological categories are taken from the reference sentences and the wordform is regenerated by the inflection module), we suspect that the main reason behind the poor performance is truly the inflection module. Aside from that, we can see that this time it was the Majority scheme which performed much better than the rest. However, the results might only point to the fact that the Majority scheme marked the smallest number of instances as incorrect thus worsening the MT output much less than the other two.

Nevertheless, we decided to pick *Majority-CNG* (*Major-CNG*) and *Average-CNG* (*Avg-CNG*) configurations for the final evaluation. Their performance is summarized in Table 7.8. Again, both systems performed poorly and it might

Dataset	System	Oracle	Base	Case	CN	CNG	Comb
Autodesk	–	46.23	45.90	45.96 (+0.06)	45.95 (+0.05)	**46.02 (+0.12)**	45.98 (+0.08)
HimL	Moses	31.94	30.95	31.37 (+0.41)	31.29 (+0.34)	**31.59 (+0.63)**	31.46 (+0.50)
WMT16	UEDIN NMT	35.05	34.82	34.82 (0)	34.82 (0)	34.82 (0)	34.82 (0)
	UEDIN PBMT	29.38	29.11	29.11 (0)	29.11 (0)	29.11 (0)	29.11 (0)

Table 7.6: Automatic evaluation of German morphological prediction module using BLEU metric and the relative improvement over the baseline MT output. Values are multiplied by 100 for easier reading. Performance of Oracle classifier is provided for comparison. The best model for each dataset is printed in bold.

Dataset	System	Oracle	Base	Majority	AtLeastOne	Average
Autodesk	–	46.23	45.90	**45.80 (-0.10)**	45.52 (-0.38)	45.79 (-0.11)
HimL	Moses	31.94	30.95	**30.89 (-0.05)**	30.17 (-0.77)	30.58 (-0.36)
WMT16	UEDIN NMT	35.05	34.82	**33.25 (-1.56)**	30.15 (-4.67)	30.78 (-4.03)
	UEDIN PBMT	29.38	29.11	**27.96 (-1.15)**	25.41 (-3.7)	25.97 (-3.14)

Table 7.7: Automatic evaluation of the error detection module using different voting methods to interpret output of multiple models using BLEU metric. Values are multiplied by 100 for easier reading, and the relative improvement over the baseline MT output. Performance of the Oracle classifier is provided for comparison. The best model for each dataset is printed in bold.

look like the Czech MLFix provided the best results. Therefore, we also provide a summary of the number of sentences that were changed by each system in Table 7.9. We can see that our suspicion that the negative score correlates with the recall of each configuration was not completely wrong. When we compare results we gathered during the Oracle evaluation and model development for each language, we do not think that the poor performance during final evaluation was caused mainly by the fixing components. We suspect that the current performance bottleneck lies within the inflection module. A more thorough investigation of the inflection module is required in the future, with a possible replacement with an alternative.

We also performed manual evaluation of the Avg-CNG configuration[4]. Two independent non-native German speakers jointly evaluated 444 changed sentences. The results of manual evaluation are in Table 7.10. They both evaluated a subset of 141 to measure their inter-annotator agreement, shown in Table 7.11. We can see that the impact of the German pipeline was similar to the Czech pipeline. However, the low precision (∼13%) confirms the results measured by the automatic metric. This is further supported by a reasonably high inter-annotator agreement of 83%.

[4]At the moment, we cannot surely tell what is the best possible configuration for German, so we simply followed the approach from English-Czech pipeline.

Dataset	System	Base	Major-CNG	Avg-CNG	CS-Best
Autodesk	–	45.90	45.64 (-0.26)	45.54 (-0.36)	45.82 (-0.08)
HimL	Moses	30.95	34.81 (-0.56)	28.35 (-2.60)	30.95 (0)
WMT16	UEDIN NMT	34.82	30.46 (-4.36)	19.95 (-14.87)	34.81 (0)
	UEDIN PBMT	29.11	25.70 (-3.41)	17.02 (-12.09)	29.11 (0)

Table 7.8: Final evaluation of the Englsh-German configuration of MLFix using BLUE metric. Values are multiplied by 100 for easier reading. Majority-CNG and Avg-CNG methods were compared with the best English-Czech configuration.

Dataset	System	Major-CNG	Avg-CNG	CS-Best	Sent.
Autodesk	–	6,626	8,794	2,535	124,498
HimL	Moses	145	421	0	800
WMT16	UEDIN NMT	2,078	2,949	3	2,999
	UEDIN PBMT	2,014	2,921	5	2,999

Table 7.9: Number of sentences changed by different systems. Total number of sentences in each dataset (*Sent.*) is listed for reference.

	Evaluated	Changed	+	–	0	Precision	Impact
A	640	313	36	263	14	12.0%	5.6%
B	320	141	18	118	5	13%	5.6%
Total	960	444	54	381	19	12.4%	5.6%

Table 7.10: Results of the manual evaluation of chosen German MLFix configuration (Avg-CNG) on a subset of HimL dataset.

A/B	+	–	0
+	7	10	1
–	6	109	3
0	0	2	3

Table 7.11: Matrix containing inter-annotator agreement of German MLFix manual evaluation.

8. Conclusion

In this thesis, we presented MLFix, an automatic post-editing tool focusing on statistical post-editing of incorrect morphology in machine translation output. The system was developed as a successor of a rule-based system, Depfix, with the aim to generalize some of its rules to a stochastic model which can be applied across languages.

During the development, we had to find a compromise between the level of language independence and overall usefulness of the system. In the end, we have chosen a unique approach to the problem of correcting the morphology by solving a two-step classification task: error detection and morphological prediction. We have faced a problem of automatic identification of correct/incorrect training instances which we have solved with a fairly effective heuristic. Still, further refinement of the training data extraction method is desired in the future.

Out of the two classification tasks, the morphological prediction proved to be much easier. The resulting models, while being very good at predicting simple categories (e.g. morphological case) manifested much lower individual performance with increasing task complexity. However when combined together, they performed quite well. Also, since we have used only really small datasets for model training, we think that these models can be improved in the future by increasing the training data or providing additional features. This claim is supported by the model trained on a much larger Autodesk dataset, which achieved really good in-domain performance even when it was trained to classify multitask problem (prediction of case-gender-number) Furthermore, we think that these models have a potential use even in different fields of application, e.g. as a part of automatic correction suggestion in a human post-editing framework.

The task detecting targets for our morphological prediction tool became main a hurdle during the development of MLFix. Aside from the correct identification of the training instances in our data, there was also an issue with highly unbalanced training set which we partially resolved by upsampling the minority class and filtering out instances from the “correct” sentences. Even though the resulting models' performance seemed unsatisfactory at first, they performed resonably well during final evaluation as far as precision of the resulting system was concerned. The weaker side was the fairly low overall impact on the MT output. As far as future improvement goes, the results achieved during model training on the large Autodesk dataset suggest that the performance can still be improved simply by increasing the amount of our training data.

As we are mentioning using larger training datasets in the future, in the scope of this thesis we have focused mainly on investing human post-edited data that are, at the moment, available in much smaller volumes than data containing reference translation. However, we have tried training few models on smaller datasets with reference sentences instead of human post-editing. The resulting models still performed fairly well if the reference sentences were reasonably similar to the MT output. Lastly, we have also examined data created by replacing the human post-editing with Depfix output resulting in reliable source of training data. These data tend to be much more sparse (as a result of Depfix impact on the MT output) and the method is currently restricted to English-Czech language pair only. However, if we can achieve adapting Czech models for other languages in the future, this method might become viable.

During the final evaluation, the system performed well when measured with the BLEU scoring metric. These results were confirmed to some extent by manual evaluation, however, we are still not completely confident in the results and suggest evaluating MLFix on much larger scale in the future. Surprisingly, MLFix was able to surpass Depfix when it was applied to the output of NMT system. This result caught our attention and will be investigated closely in the near future. If confirmed, the application to the increasingly popular approach to the machine translation might become valuable.

We have also evaluated performance of a modification aimed at correction German SMT output. We were satisfied with the results during model development, achieving results similar to Czech pipeline, which confirms that the classification tasks themselves (as they were defined in the thesis) are not language dependent. No special attention to manually modifying the feature set or choosing different approach was required.

The results of the final evaluation for German were however poor. Aside from the morphology module applied separately, German MLFix always worsened the MT output as the automatic metric have shown. This was further confirmed by a manual evaluation. We pointed out several indications that this might be caused by a poor performance of the German inflection module we use to generate new wordforms. We still have to investigate the matter further to confirm this hypothesis. If confirmed, replacing (or improving) the module in the future might be the first and fastest way to improvement. Another goal is to investigate the German MT errors more thoroughly and adapt the German pipeline to the findings.

Even though we mainly focused on morphology correction in this thesis, MLFix can be further improved in the future by introducing other statistical modules addressing additional MT errors. As an example we mention a possible word reordering model because there were instances where words with new surface forms predicted by MLFix still needed rearrangement to fully utilize the modification. Due to the modularity of Treex framework, introducing new improvements to MLFix is easy.

Bibliography

Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004. ISSN 1931-0145. doi: 10.1145/1007730.1007735. URL `http://doi.acm.org/10.1145/1007730.1007735`.

Hanna Bechara. Statistical Post-editing and Quality Estimation for Machine Translation Systems. Master's thesis, Dublin city University, School of Computing, 2013.

Hanna Béchara, Yanjun Ma, and Josef van Genabith. Statistical Post-Editing for a Statistical MT System. In *Proceedings of the 13th Machine Translation Summit*, pages 308–315, 2011.

Bernd Bohnet. Very High Accuracy and Fast Dependency Parsing is Not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 89–97, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1873781.1873792`.

Ondřej Bojar, Bushra Jawaid, and Amir Kamran. Probes in a Taxonomy of Factored Phrase-Based Models. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 253–260, Montréal, Canada, 2012a. Association for Computational Linguistics. ISBN 978-1-937284-20-6.

Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. The Joy of Parallelism with CzEng 1.0. In *Proceedings of the Eighth International Language Resources and Evaluation Conference (LREC'12)*, pages 3921–3928, Istanbul, Turkey, Květen 2012b. ELRA, European Language Resources Association. ISBN 978-2-9517408-7-7.

Ondřej Bojar, Rudolf Rosa, and Aleš Tamchyna. Chimera – Three Heads for English-to-Czech Translation. In *Proc. of the WMT*, pages 92–98, Sofia, Bulgaria, 2013a. ACL. URL `http://www.aclweb.org/anthology/W13-2208`.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 Conference on Machine Translation (WMT16). In *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W16/W16-2301`.

Ondřej Bojar, Matouš Macháček, Aleš Tamchyna, and Daniel Zeman. Scratching the Surface of Possible Translations. In *Text, Speech and Dialogue: 16th*

International Conference, TSD 2013. Proceedings, pages 465–474, Berlin / Heidelberg, 2013b. Springer Verlag. ISBN 978-3-642-40584-6.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. TIGER: Linguistic Interpretation of a German Corpus. *Research on Language and Computation*, 2(4):597–620, 2004. ISSN 1572-8706. doi: 10.1007/s11168-004-7431-3. URL `http://dx.doi.org/10.1007/s11168-004-7431-3`.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W10-1703`. Revised August 2010.

Pedro Domingos. MetaCost: A General Method for Making Classifiers Cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 155–164, New York, NY, USA, 1999. ACM. ISBN 1-58113-143-7. doi: 10.1145/312129.312220. URL `http://doi.acm.org/10.1145/312129.312220`.

Ondřej Dušek and Filip Jurčíček. Robust multilingual statistical morphological generation models. In *51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, Proceedings of the Student Research Workshop, 4-9 August 2013, Sofia, Bulgaria*, pages 158–164. The Association for Computer Linguistics, 2013. URL `http://aclweb.org/anthology/P/P13/P13-3023.pdf`.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219885. URL `http://dx.doi.org/10.3115/1219840.1219885`.

Jan Hajič. *Disambiguation of Rich Inflection - Computational Morphology of Czech*, volume I. Karolinum, Charles Univeristy Press, Prague, Czech Republic, 2004.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková Razímová. Prague Dependency Treebank 2.0. LDC2006T01, ISBN: 1-58563-370-4, 2006.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and

Yi Zhang. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, Boulder, Colorado, USA, 2009.

Zhongye Jia, Peilu Wang, and Hai Zhao. Grammatical Error Correction as Multiclass Classification with Single Model. In *Proc. of 7th CoNLL: Shared Task*, Sofia, Bulgaria, 2013.

Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT. URL `http://mt-archive.info/MTS-2005-Koehn.pdf`.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1557769.1557821`.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proc. of HLT/EMNLP*, 2005.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Paris, France, 2016. European Language Resources Association. ISBN 978-2-9517408-9-1.

Franz Josef Och and Hermann Ney. A Comparison of Alignment Models for Statistical Machine Translation. In *Proc. of COLING*, pages 1086–1090. ACL, 2000. ISBN 1-555-55555-1.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL*, pages 311–318, Philadelphia, Pennsylvania, 2002.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Martin Popel and Zdeněk Žabokrtský. TectoMT: Modular NLP Framework. In *Proceedings of the 7th International Conference on Advances in Natural*

Language Processing, IceTAL'10, pages 293–304, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-14769-0, 978-3-642-14769-2. URL `http://dl.acm.org/citation.cfm?id=1884371.1884406`.

Rudolf Rosa. Automatic post-editing of phrase-based machine translation outputs. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Praha, Czechia, 2013.

Rudolf Rosa. Depfix, a Tool for Automatic Rule-based Post-editing of SMT. *The Prague Bulletin of Mathematical Linguistics*, 102:47–56, 2014. ISSN 0032-6585.

Rudolf Rosa, Ondřej Dušek, David Mareček, and Martin Popel. Using Parallel Features in Parsing of Machine-Translated Sentences for Correction of Grammatical Errors. In *Proceedings of Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-6), ACL*, pages 39–48, Jeju, Korea, 2012a. Association for Computational Linguistics. ISBN 978-1-937284-38-1.

Rudolf Rosa, David Mareček, and Ondřej Dušek. DEPFIX: A System for Automatic Correction of Czech MT Outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 362–368, Montréal, Canada, 2012b. Association for Computational Linguistics. ISBN 978-1-937284-20-6.

Petr Sgall. *Generativní popis jazyka a česká deklinace.* Prague: Academia, 1967.

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. Rule-Based Translation with Statistical Phrase-Based Post-Editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W07/W07-0228`.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231, 2006.

Jana Straková, Milan Straka, and Jan Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P14/P14-5003.pdf`.

Kristina Toutanova and Christopher D. Manning. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-speech Tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 63–70, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/1117794.1117802. URL `http://dx.doi.org/10.3115/1117794.1117802`.

Daniel Zeman. Reusable Tagset Conversion Using Tagset Drivers. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, pages 213–218, Marrakech, Morocco, 2008. European Language Resources Association. ISBN 2-9517408-4-0.

List of Figures

List of Tables

A. Contents of the CD

The attached CD contains following items:

- *config* - configuration files for the statistical components
- *data* - sample input data
- *INSTALL* - a file containing a manual for MLFix installation
- *Makefile* - a Makefile containing MLFix commands
- *models* - models used by MLFix components that are currently outside of Treex-shared directory
- *README* - manual for using MLFix
- *scenarios* - scenarios that are not implemented by Treex::Scen:: blocks
- *scripts* - scripts used for minor tasks (e.g. model training, data pre-, post-processing)
- *settings_cs.mak*, *settings_de.mak* - settings files for EN-CS and EN-DE ML-Fix pipeline
- *thesis.pdf* - a PDF file containing this thesis

B. MLFix Scenarios

In this attachement we list both English-Czech and English-German scenarios we have used for MLFix processing pipeline. The scenarios can be generated by `Treex::Scen::MLFix::` blocks. At the beginning of each scenario we have included a commented Treex command which dumps the scenario, on the command line. The scenarios are listed in the order they are applied to the input. Note that Czech and German blocks are called only in their respective scenario. Blocks for processing English are identical in both pipelines, however, we list them in both scenarios for easier comprehension.

B.1 English-Czech

B.1.1 Analysis on M-layer

```
# Source (English)
# treex -d Scen::MLFix::Analysis_1 language=en
   iset_driver="en::penn"
Util::SetGlobal language=en selector=
Util::Eval zone='$zone->remove_tree("a") if $zone->
   has_tree("a");'
W2A::EN::Tokenize
W2A::EN::NormalizeForms
W2A::EN::FixTokenization
W2A::EN::TagMorphoDiTa lemmatize=0
W2A::EN::FixTags
W2A::EN::Lemmatize
A2A::ConvertTags input_driver=en::penn

# treex -d Scen::MLFix::NER language=en model=ner-eng-
   ie.crf-3-all2008.ser.gz
Util::SetGlobal language=en
A2N::EN::StanfordNamedEntities model=ner-eng-ie.crf-3-
   all2008.ser.gz
A2N::EN::DistinguishPersonalNames


# Target (Czech)
# treex -d Scen::MLFix::Analysis_1 language=cs tagger=
   morphodita iset_driver="cs::pdt"
Util::SetGlobal language=cs selector=
Util::Eval zone='$zone->remove_tree("a") if $zone->
   has_tree("a");'
W2A::CS::Tokenize
W2A::CS::TagMorphoDiTa lemmatize=1
W2A::CS::FixMorphoErrors
```

```
W2A::CS::FixGuessedLemmas
A2A::ConvertTags input_driver=cs::pdt
A2N::CS::SimpleRuleNER

# Reference (Czech)
# treex -d Scen::MLFix::Analysis_1 language=cs selector
   =ref tagger=morphodita iset_driver="cs::pdt"
Util::SetGlobal language=cs selector=ref
Util::Eval zone='$zone->remove_tree("a") if $zone->
   has_tree("a");'
W2A::CS::Tokenize
W2A::CS::TagMorphoDiTa lemmatize=1
W2A::CS::FixMorphoErrors
W2A::CS::FixGuessedLemmas
A2A::ConvertTags input_driver=cs::pdt
A2N::CS::SimpleRuleNER
```

B.1.2 Alignment

```
# English-Czech
# treex -d Scen::MLFix::RunMGiza from_language=cs
   to_language=en model=cs-en
Align::A::AlignMGiza dir_or_sym=intersection selector=
   from_language=cs to_language=en model_from_share=cs-
   en tmp_dir=/mnt/h/tmp cpu_cores=1
Align::AddMissingLinks layer=a selector= language=cs
   target_language=en alignment_type=intersection
Align::ReverseAlignment language=cs selector=

# Reference
Align::A::MonolingualGreedy selector=T language=cs
   to_selector=ref
```

B.1.3 Analysis on A-layer

```
# Source (English)
# treex -d Scen::MLFix::Analysis_2 language=en parser=
   mst
Util::SetGlobal language=en selector=
W2A::EN::ParseMST
W2A::EN::SetIsMemberFromDeprel
W2A::EN::RehangConllToPdtStyle
W2A::EN::FixNominalGroups
W2A::EN::FixIsMember
W2A::EN::FixAtree
W2A::EN::FixMultiwordPrepAndConj
W2A::EN::FixDicendiVerbs
```

```
W2A::EN::SetAfunAuxCPCoord
W2A::EN::SetAfun


# Target (Czech)
# treex -d Scen::MLFix::Analysis_2 language=cs
   src_language=en parser=
Util::SetGlobal language=cs selector=
A2A::ProjectTreeThroughAlignment language=en
   to_language=cs to_selector=

# Reference (Czech)
# treex -d Scen::MLFix::Analysis_2 language=cs selector
   =ref
Util::SetGlobal language=cs selector=ref
W2A::CS::ParseMSTAdapted
W2A::CS::FixAtreeAfterMcD
W2A::CS::FixIsMember
W2A::CS::FixPrepositionalCase
W2A::CS::FixReflexiveTantum
W2A::CS::FixReflexivePronouns
```

B.1.4 Fixing

```
# Preparation
# treex -d Scen::MLFix::FixPrepare src_language=en
   tgt_language=cs
Util::Eval language=cs selector=T zone='$zone->
   remove_tree("a") if $zone->has_tree("a");'
Util::Eval language=cs selector=FIXLOG zone='$zone->
   set_sentence("");'
A2A::CopyAtree source_language=cs language=cs selector=
   T align=1
Align::AlignForward language=cs selector=T overwrite=0
   preserve_type=0

# Fixing
# treex -d Scen::MLFix::Fix "mark_method=scikit-learn "
   fix_method=scikit-learn" "language=cs" "selector=" "
   mark_config_file=XXX" "fix_config_file=XXX" "
   iset_driver=cs::pdt"
MLFix::MarkByScikitLearn language=cs selector=
   config_file=XXX
MLFix::CS::ScikitLearn language=cs selector=
   config_file=XXX iset_driver=cs::pdt
```

B.1.5 Detokenization

```
# treex -d Scen::MLFix::WriteSentences language=cs
Util::SetGlobal language=cs selector=
A2W::Detokenize
A2W::CS::DetokenizeUsingRules
A2W::CS::DetokenizeDashes
Util::Eval zone='print $zone->sentence . "\n";'
```

B.2 English-German

B.2.1 Analysis on M-layer

```
# Source (English)
# treex -d Scen::MLFix::Analysis_1 language=en
   iset_driver="en::penn"
Util::SetGlobal language=en selector=
Util::Eval zone='$zone->remove_tree("a") if $zone->
   has_tree("a");'
W2A::EN::Tokenize
W2A::EN::NormalizeForms
W2A::EN::FixTokenization
W2A::EN::TagMorphoDiTa lemmatize=0
W2A::EN::FixTags
W2A::EN::Lemmatize
A2A::ConvertTags input_driver=en::penn

# treex -d Scen::MLFix::NER language=en model=ner-eng-
   ie.crf-3-all2008.ser.gz
Util::SetGlobal language=en
A2N::EN::StanfordNamedEntities model=ner-eng-ie.crf-3-
   all2008.ser.gz
A2N::EN::DistinguishPersonalNames


# Target (Czech)
# treex -d Scen::MLFix::Analysis_1 language=de tagger=
   mate iset_driver="de::conll2009"
Util::SetGlobal language=de selector=
Util::Eval zone='$zone->remove_tree("a") if $zone->
   has_tree("a");'
W2A::DE::Tokenize
W2A::DE::LemmatizeMate
W2A::DE::ParseMate lemmatize=0
A2A::DE::CoNLL2Iset

# Reference (Czech)
# treex -d Scen::MLFix::Analysis_1 language=de selector
   =ref tagger=mate iset_driver="de::conll2009"
Util::SetGlobal language=de selector=ref
Util::Eval zone='$zone->remove_tree("a") if $zone->
   has_tree("a");'
W2A::DE::Tokenize
W2A::DE::LemmatizeMate
W2A::DE::ParseMate lemmatize=0
A2A::DE::CoNLL2Iset
```

B.2.2 Alignment

```
# English-German
# treex -d Scen::MLFix::RunMGiza from_language=de
   to_language=en model=de-en
Align::A::AlignMGiza dir_or_sym=intersection selector=
   from_language=de to_language=en model_from_share=de-
   en tmp_dir=/mnt/h/tmp cpu_cores=1
Align::AddMissingLinks layer=a selector= language=de
   target_language=en alignment_type=intersection
Align::ReverseAlignment language=de selector=

# Reference
# Align::A::MonolingualGreedy selector= language=de
   to_selector=ref
```

B.2.3 Analysis on A-layer

```
# Source (English)
# treex -d Scen::MLFix::Analysis_2 language=en parser=
   mst
Util::SetGlobal language=en selector=
W2A::EN::ParseMST
W2A::EN::SetIsMemberFromDeprel
W2A::EN::RehangConllToPdtStyle
W2A::EN::FixNominalGroups
W2A::EN::FixIsMember
W2A::EN::FixAtree
W2A::EN::FixMultiwordPrepAndConj
W2A::EN::FixDicendiVerbs
W2A::EN::SetAfunAuxCPCoord
W2A::EN::SetAfun


# Target (German)
# treex -d Scen::MLFix::Analysis_2 language=de
   src_language=en parser=
Util::SetGlobal language=de selector=
A2A::ProjectTreeThroughAlignment language=en
   to_language=de to_selector=

# Reference (German)
# treex -d Scen::MLFix::Analysis_2 language=de selector
   =ref
Util::SetGlobal language=de selector=ref
W2A::DE::ParseMate
A2A::DE::CoNLL2Iset
```

B.2.4 Fixing

```
# Preparation
# treex -d Scen::MLFix::FixPrepare src_language=en
   tgt_language=de
Util::Eval language=de selector=T zone='$zone->
   remove_tree("a") if $zone->has_tree("a");'
Util::Eval language=de selector=FIXLOG zone=$zone->
   set_sentence("");
A2A::CopyAtree source_language=de language=de selector=
   T align=1
Align::AlignForward language=de selector=T overwrite=0
   preserve_type=0

# Fixing
# treex -d Scen::MLFix::Fix mark_method=scikit-learn
   fix_method=scikit-learn language=de selector=
   mark_config_file=XXX fix_config_file=XXX iset_driver
   =de::conll2009
MLFix::MarkByScikitLearn language=de selector=
   config_file=XXX
MLFix::DE::ScikitLearn language=de selector=
   config_file=XXX iset_driver=de::conll2009
```

B.2.5 Detokenization

```
# treex -d Scen::MLFix::WriteSentences language=de
Util::SetGlobal language=de selector=
A2W::Detokenize
Util::Eval zone='print $zone->sentence . "\n";'
```