FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Vilém Zouhar

Enabling Outbound Machine Translation

Institute of Formal and Applied Linguistics

Supervisor of the bachelor thesis: doc. RNDr. Ondřej Bojar, Ph.D.
Study programme: Computer Science
Study branch: General Computer Science

Prague 2020

I would like to express appreciation not only to my supervisor Ondřej Bojar, but also to other people who contributed to this thesis. Namely Michal Novák, Dominik Macháček and Tereza Vojtěchová (Charles University) and Frederic Blain (University of Sheffield). Special thanks to all the people from ÚFAL with whom I spent time during my stay in the office 423 at Malá Strana.

This results of this thesis were also used for the Bergamot project which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825303.

We used language resources developed and/or stored and/or distributed by the LINDAT-Clarin project of the Ministry of Education of the Czech Republic (project LM2010013).

Title: Enabling Outbound Machine Translation

Author: Vilém Zouhar

Institute: Institute of Formal and Applied Linguistics

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

Abstract: It is not uncommon for Internet users to have to produce text in a foreign language they have very little knowledge of and are unable to verify the translation quality. We call the task "outbound translation" and explore it by introducing an open-source modular system Ptakopět. Its main purpose is to inspect human interaction with machine translation systems enhanced by additional subsystems, such as backward translation and quality estimation. We follow up with an experiment on (Czech) human annotators tasked to produce questions in a language they do not speak (German), with the help of Ptakopět. We focus on three real-world use cases (communication with IT support, describing administrative issues and asking encyclopedic questions) from which we gain insight into different strategies users take when faced with outbound translation tasks. Round trip translation is known to be unreliable for evaluating MT systems but our experimental evaluation documents that it works very well for users, at least on MT systems of mid-range quality.

Keywords: quality estimation, machine translation, outbound translation, natural language processing, web application

Contents

Introduction

It is common for especially Internet users to have to produce text in a foreign language where they are unable to verify the quality of the proposed machine translation. The issue at hand is referred to as "outbound translation," which is a complement to gisting, also known as inbound translation.

Both in gisting and outbound translation, a message is transferred between an author and a recipient. The user has sufficient knowledge of only one of the languages. In outbound translation, the user is responsible for creating correct messages, while in gisting, their responsibility is to interpret them correctly. An example of outbound translation may be filling out forms in a foreign language, describing an issue to technical support, or ordering food in a foreign restaurant. An example of gisting is to understand Internet articles, presentations correctly, or to read restaurant menus, all in a foreign language.

When translating to foreign languages, users cooperate with machine translation tools to produce the best result. Many users also use machine translation to verify their own translation, or at least to affirm, that the machine translation is valid via backward translation. Machine translation systems advanced considerably in the last decade due to breakthroughs in statistical and later neural machine translation models but they still make often mistakes. Thankfully, they are of a vastly different kind than the mistakes people make when translating texts and so are detectable.

Outbound translation is a use case, which requires some confirmation of the translation quality. Users translating to languages which they do not master enough to validate the translation could use some additional system to verify that the machine translation output is valid and assure them. This system may be the missing connection necessary for establishing trust of the user to the machine translator. We hope that MT systems will achieve perfection, but we do not expect this to happen in the foreseeable future and especially for all possible language pairs.

The issue of outbound translation has not yet been fully explored, yet there exists much research on related problems, such as quality estimation and bitext alignment, which has been productized by translation companies for minimizing post-editing costs and by other NLP companies for robust information retrieval.

The proposed tool, Ptakopět, aims to showcase how such a product intended to help with outbound translation may function. It provides the coveted user experience by several quality estimation pipelines and backward translation, which we hope is served in an unobtrusive, yet informative way.

We follow up with an experiment whose objective is to examine different phenomena and strategies users use with the help of Ptakopět when they are faced with a task involving outbound translation. Such strategies are then further examined based on their performance (relevance and quality of produced foreign texts). This experiment also gives us valuable feedback from both the user's and experiment designer's perspective. This is very beneficial to us, as we can improve the system for future experiments with Ptakopět.

Thesis overview

This thesis is divided into five chapters. The first one introduces terminology and issues related to machine translation, quality estimation, bitext alignment and source complexity, as well as relevant areas of research.

The second chapter covers the development and deployment of two previous versions of Ptakopět (old-1 and old-2) as well as what we aimed to improve and what we learned from them. We then mention different approaches to outbound translation and what we think are their positives and negatives. We also briefly discuss the industry adoption of quality estimation systems.

The next chapter shows the behaviour of Ptakopět from the user perspective, together with screenshots and use case descriptions.

Data preparation, experiment setup, frontend, and backend implementation, quality estimation, and bitext alignment model and deployment and usage of the current version of Ptakopět is the focus of the fourth chapter.

In the fifth chapter, we propose, prepare, realize, and evaluate an experiment on the strategies users take when faced with outbound translation with the help provided by Ptakopět.

At the end of this thesis, we conclude and summarize both the Ptakopět implementation and the experiment interpretation with notes on future work in the area of outbound translation.

1. Background

This chapter aims to provide a brief overview of the available tools related to outbound translation, namely Alignment, Machine translation, Quality estimation and Source complexity.

1.1 Alignment

The alignment of parts of texts in parallel corpora is a common task in NLP and especially in tasks related to statistical machine translation. One of their use cases is creating resources used by MT systems for training. Usually, paragraph, sentence, phrase, and word alignment levels are considered. The second one and the last one are the most important for quality estimation.

A comprehensive source of alignment data is OPUS (Tiedemann, 2012), which contains documents automatically aligned both on the sentence-level and word-level.

1.1.1 Sentence alignment

The task of sentence alignment is to match two groups of sentences in parallel data (usually in two languages), for example in two paragraphs or documents, if and only if they are each other's translations. A classical approach is unsupervised. In this case, the problem can be approached with a dynamic programming algorithm described by Gale and Church (1993). Another approach is to view it as a machine learning problem, as summarized in Yu et al. (2012). Most of the approaches rely in some way on comparing the number of either, characters, words, or both in a sentence. The main idea is that a sentence's length correlates with the length of its translated counterpart.

Advances in deep neural models led to novel approaches to sentence alignment. As reported in Zweigenbaum et al. (2017), they achieve the state of the art results. However, such models are not yet readily available for deployment.

Given a sentence alignment and a gold standard, the recall, precision and F-measure are the most common metrics.

Bleualign

Bleualign[1] is a sentence alignment tool, which apart from the source and the target corpora requires machine translation of at least one of the texts. The alignment is then performed based on the similarity between two texts (in the target language) in the same sentence with a modified BLEU score. This approach was proposed in Sennrich and Volk (2011).

[1]github.com/rsennrich/bleualign

Yalign

Yalign[2] is an open-source product of the Machinalis company. It takes a dynamic programming approach to sentence alignment, using a sentence similarity metric, which in turn utilizes SVM to tell whether a pair of sentences are each other's translations.

Hunalign

Hunalign[3] uses the classical algorithm based on sentence-length information described in Gale and Church (1993). We found it to be more robust and usable compared to other publicly available sentence alignment tools.

1.1.2 Word alignment

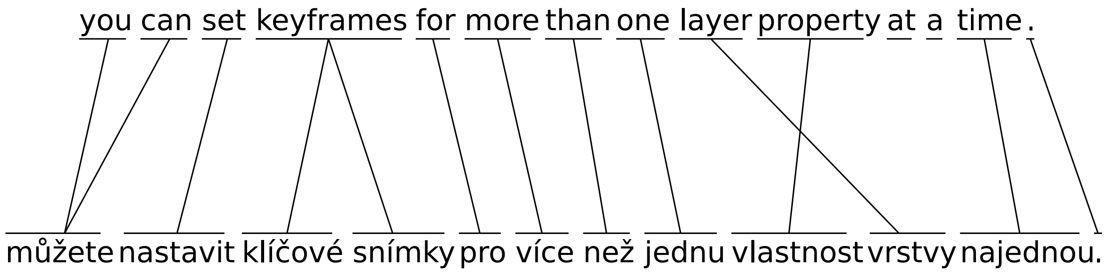

you can set keyframes for more than one layer property at a time .
můžete nastavit klíčové snímky pro více než jednu vlastnost vrstvy najednou.

Figure 1.1: Depiction of word alignment of the third sentence in WMT18 QE development data (English to German)"

Word alignment is the task of matching two groups of words in a bitext sentence if and only if they are each other's translations. An example[4] of word alignment between an English sentence and translated German sentence can be seen in Figure 1.1. Word alignment usually follows after sentence alignment.

	Klicken	Sie	mit	der	Maustaste	.
Click	■	■	■			
the				■		
mouse					■	
button					■	
.						■

Figure 1.2: Bitext matrix of English to German word alignment

Sometimes the word alignment is depicted as in Figure 1.2, which is known as a bitext matrix. There is a point on $A_{i,j}$ if and only if the $i-$th source word

[2]github.com/machinalis/yalign

[3]github.com/danielvarga/hunalign

[4]For the purposes of displaying word-alignment in the style of Figure 1.1 and Figure 1.3, we wrote a small web-based tool SlowAlign. It is publicly available at vilda.net/s/slowalign. The source code is both hosted at github.com/zouharvi/SlowAlign and attached to this thesis.

is aligned to the $j-$th target word. Diagonal bitext alignment would correspond to a bitext matrix with points on its diagonal (a geometrical diagonal for non-square matrices). Sometimes it is impossible to do word alignment. Example of this are idiomatic phrases, whose translations can be idiomatic phrases in the target language and then there is no explicit mapping between individual words. We show such an example of only a partial matching in Figure 1.3. In this case using phrase-level alignment may be more suitable, as explored in Bojar and Prokopová (2006). However, for most cases, we are able to construct meaningful word-level alignment.

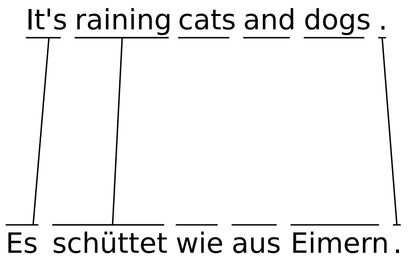


Figure 1.3: A partial alignment of an English idiom and its corresponding German translation.

For a long time, increasingly complex IBM Alignment Models 1 to 6 dominated the area of word-level alignment.[5] These models are a coproduct of statistical machine translation models. Summary of the development of IBM word alignment models can be found in the second chapter of Cuřín (2006).

Similarly to sentence alignment, research in the area of neural networks allowed for state-of-the-art results. An example of this can be found in Legrand et al. (2016), where the authors use the dot product of windows from two convolutional layers (source and target) as the alignment score. As far as we know, none of the neural models has been made into a deployable word alignment tool. There are however hints of progress, as word alignment can be extracted from neural attention based machine translation systems, shown in Nizza[6] and Neural Machine Translation.[7]

The task of word alignment is formalized in Mihalcea and Pedersen (2003) together with four standard metrics: precision, recall, F-measure and quality of word alignment.

[5]statmt.org/survey/Topic/IBMModels

[6]github.com/fstahlberg/nizza

[7]github.com/tilde-nlp/neural-machine-translation-tools

GIZA, GIZA++, MGIZA++, PGIZA++

The GIZA word alignment collection is a standard used for this task. The first version, GIZA, was part of the EGYPT SMT toolkit and was later extended to GIZA++ and later became part of the Moses SMT toolkit.[8] It incorporates IBM Model 4, IBM Model 5 and Hidden Markov Models. The support for multithreading and clusters was later added, resulting in MGIZA++[9] and PGIZA++[10], respectively.

In addition to word alignment, all of the ++ versions can perform some rudimentary form of sentence alignment.

Fast_align

Fast align (Dyer et al., 2013) is a word aligner based on IBM Alignment Model 2, with improvements on this model's over-parametrization issues. It outperforms the commonly used IBM Alignment Model 4, is easy to build and use and has low resource requirements (both memory and computational time). It is often used as a baseline solution.

Eflomal

Eflomal[11] is an improved version of Efmaral presented in Östling and Tiedemann (2016) together with comparison to the other two-word alignment systems. It is based on Bayesian alignment models with Markov Chain Monte Carlo inference. It is currently the state-of-the-art in word alignment.

1.2 Machine translation

The task of machine translation is to translate text from one language to another. In this context, it is strictly automatic and should not be confused with machine-aided human translation. This is an extensive topic and beyond the scope of this thesis. We thus provide only a very brief overview of the evolution and main approaches.

1.2.1 Rule based (RBMT)

First attempts for machine translation were made with rule-based systems, with either direct, syntactic or semantic transfer, which corresponds to steps described by the Vauquois triangle.[12] Models based on the manipulation of morphological, syntactic and semantic structures are known as transfer based. They require linguistic knowledge of both source and target languages to be produced. We list examples of RBMT systems with short descriptions. Most of them became obsolete with the advent of statistical machine translation systems.

[8]statmt.org/moses/
[9]github.com/moses-smt/mgiza
[10]www.cs.cmu.edu/q̃ing/giza/
[11]github.com/robertostling/eflomal
[12]mttalks.ufal.ms.mff.cuni.cz/index.php?title=Intro#Types_of_MT_systems

- Ruslan - Created for Czech-Russian translation (Hajič, 1987).
- APAČ - Created for Czech-English translation (Kirschner and Rosen, 1989).
- METEO - Used for 20 years for English-French Canada weather reports translation (Lawson, 1982).
- SYSTRAN - Developed by one of the oldest companies providing MT. It supports many language pairs and transitioned to SMT and NMT later (Toma, 1977).

1.2.2 Statistical (SMT)

Statistical machine translation models rely on a great amount of parallel data. They are usually modelled with two components: translation model and language model. This is shown in Equation 1.1 which shows the common application of the Bayes theorem for best translation t given a source sentence s. The model is split into the translation model $p(s|t)$, evaluating translation relevance, and language model $p(t)$, which describes how well the proposed target sentence fits the language. The motivation is that monolingual data, used by the language model, are much easier to obtain than bilingual ones.

$$\text{argmax}_t\ p(t|s) = \text{argmax}_t\ \frac{p(t) \cdot p(s|t)}{p(s)} = \text{argmax}_t\ p(s|t) \cdot p(t) \tag{1.1}$$

The language model is usually modelled with n-grams. Its use made the translations more fluent as opposed to previous approaches.

SMT was first word-based, meaning words were manipulated and strung together to form a sentence. An example of this are the IBM models (Brown et al., 1993). More modern systems were phrase-based (documented in (Koehn et al., 2003)), meaning whole sequences of words were manipulated and composed to form a sentence. This approach was widely adopted in the industry. We list examples of MT systems which are or were SMT based.

- Google Translate - Is one of the most known publicly available MT system.[13] It started transitioning to NMT in 2016.[14]
- Bing Translator - Developed by Microsoft and is another widely used publicly available MT. It started transitioning to NMT in 2016.[15]
- SYSTRAN - Was SMT, but partially transitioned to NMT in 2016.[16]

1.2.3 Neural (NMT)

In the past decade, NMT systems became the state of the art and thus widely used. There are two main cornerstones of NMT: encoder-decoder and (self-) attention. They both rely on recent advances in representation learning.

[13]blog.google/products/translate/ten-years-of-google-translate/

[14]blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate/

[15]microsoft.com/en-us/translator/business/machine-translation/

[16]kv-emptypages.blogspot.com/2016/12/systrans-continuing-nmt-evolution.html

Embedding

One possibility to represent words in memory is using one hot encoding. Given a dictionary D, the i-th word of the dictionary would be represented by a vector of size $|D|$ of zeros except for one 1 in the i-th place. This is far from the optimal approach, and there are many associated issues, such as data sparsity. In this encoding, each word form would be equally distant from each other. So the difference between the embeddings of `birds` and `bird` would be the same as between the embeddings of `birds` and `refrigerated`.

An alternative to this is to make a neural network learn a word representation. In this context, the representation is named *word embedding.* Word2Vec (Mikolov et al., 2013) are types of networks, which can create such word embeddings out of monolingual data in an unsupervised way. The authors propose two architectures. The first one is Continuous Bag-Of-Words, which tries to predict the current word based on the context, and the other is Skip-Gram, which tries to predict the context based on the current word.

Word embeddings are able to capture relationships between embedded words. So word embeddings for `birds` and `bird` would be closer together than the word embeddings for `birds` and `refrigerated`.

Another improvement was the introduction of subword units and relevant training algorithms, such as Byte Pair Encoding (Sennrich et al., 2016). Subword units are used to deal with unseen and rare words. If a word is too rare, it will not get its own word embedding but will be split into multiple subword units. This is intuitive, as humans are able to understand new words by analyzing their morphemes.

Encoder-Decoder

The encoder-decoder architecture is the standard approach for sequence to sequence tasks. In this case, it is a sequence of word embeddings, which is passed to multi-level bidirectional RNN, composed of either Long Short Term Memory (LSTM) or Gated Recurrent Unit (GRU). The result is called the hidden state, and it is a fixed-size vector, which is passed between the cells. The hidden state represents the whole input sentence. To get the translated output, the hidden state is unrolled with the same mechanism, but this time the probabilities of all possible output subword units are produced. A search algorithm, possibly a greedy one, selects one of them and the decoder moves to the next step.

The actual recurrent neural networks used for machine translation are made more complex, such as with the use of multiple layers of encoders.

Attention

One of the deficiencies of the purely encoder-decoder approach is that both of the RNN components only have direct access to the immediately preceding word. This is a problem in case the translation of the current word is dependent on a word, which was either encountered (source) or generated (target) several steps back. Such distant word is embedded in the hidden state (fixed size vector), but it may be too difficult to extract its features.

A novel approach was taken by Bahdanau et al. (2014). They proposed a system by which a current (to be produced) word can reference the words the translation is dependent on. Such referencing is called the attention mechanism and there are two types usually distinguished: global attention and local attention. This mechanism is employed only in the decoder part and the attention mechanism adds a context vector to the word translation computation.

Global attention computes the context from all of the source words using a hidden alignment vector, which supplies weights for each word. Local attention focuses on a window of a fixed size around the aligned source word.

Popular MT models such as Transformer (Vaswani et al., 2017) or BERT (Devlin et al., 2018) make use of attention mechanisms and add other improvements, such as multi-head self-attention. As is apparent from the transition in the industry standard in the listing in Section 1.2.2, NMT together with attention mechanisms outperforms SMT and it is now the state of the art and widely used.

1.2.4 Evaluation

Automatic machine translation evaluation assigns scores to the produced machine translations, given also the source text and reference translation. The goal is to find such a metric, that would correlate the most with human annotations. This is not an easy task to automate, as the machine translation can be vastly different from the reference translation, but still be correct. Alternatively, it can be very close to the reference translation, but be very ungrammatical or can distort the meaning seriously.

Many other metrics have been proposed and this area of research is still active. The metrics shared task is part of WMT (Ma et al., 2019).

BLEU

Bilingual Evaluation Understudy (Papineni et al., 2002) is the most common metric for evaluation of MT systems. It is based on searching for n-grams in the translated sentence, which are also in the reference translation. The geometric mean of unigram, bigram, trigram and quadrigram precision is calculated and the score computed as in the formula in Equation (1.2).

The first part of the product is called the *brevity penalty*. It is there to make sure that the MT system does not game the metric by outputting for example only one quadrigram, which it is very sure about. In this case all of the precisions would be close to 1, but human annotators would not consider this to be a good translation.

$$\text{BLEU} = exp\Big(\min(1, 1 - \frac{\text{output length}}{\text{reference length}})\Big) \cdot \Big(\prod_{i=1}^{4} \text{prec}_i\Big)^{\frac{1}{4}} \tag{1.2}$$

METEOR

Metric for Evaluation of Translation with Explicit ORdering aims to fix some disadvantages of BLEU. In contrast to BLEU, it considers only unigrams and not bigrams, trigrams, nor quadrigrams. It uses stemming and synonyms and tries to create mappings between the translated and reference sentence. It has been shown (Banerjee and Lavie, 2005) that it correlates better with the human judgement than BLEU.

Round trip translation

Also called backward translation, round trip translation is an idea of evaluating an MT system without reference. It is based on translating the source sentence to a foreign language and then back and measuring BLEU score between source and backward translated text. In (Somers, 2005) it was demonstrated, that RTT does not correlate with the quality of MT systems in any statistically meaningful way.

1.3 Quality estimation

As opposed to machine translation evaluation, which uses source text and target and reference translations to evaluate the output, the objective of quality estimation (also known as confidence estimation) is to predict the translation quality without reference translations.

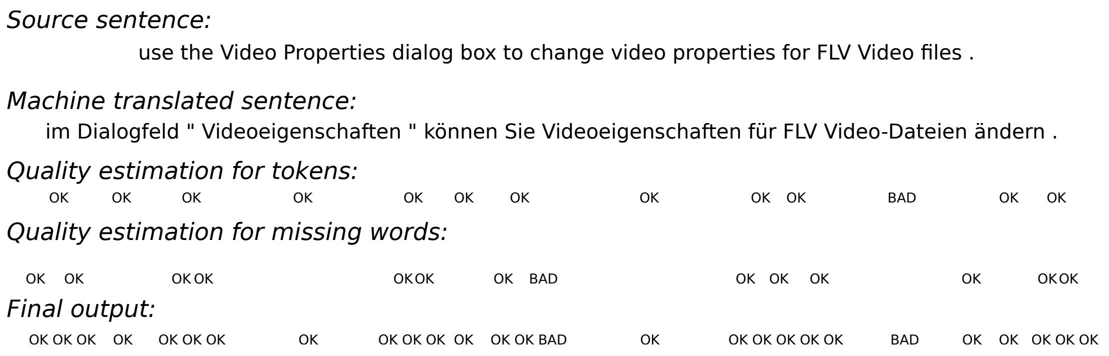
Machine translation quality estimation aims to provide hints as to whether a produced machine translation is good or bad. Quality estimation shared task was part of WMT since 2012[17] and was forked from automatic machine translation evaluation shared task, which was part of WMT since 2008[18] and in 2012 renamed to metrics task. The results of the latest findings of the WMT quality estimation shared task 2019 are discussed in (Fonseca et al., 2019).

The input of metrics task, as defined by WMT, is the translation together with human reference translations. It is performed usually on sentence-level and system-level. In contrast, the input for quality estimation system is the source text and the translated text. Quality estimation is then performed on word-level, phrase-level and sentence-level.

For document-level quality estimation, the document is either to be assigned a score, or the system should report spans, which contain translation errors. The quality of sentence-level quality estimation is measured in HTER (edit-distance metrics).

The output of word-level quality estimation is usually a list of probabilities for each token in the target sentence. In WMT, the format is only binary values for each token. For word-level quality estimation models, to encapsulate missing words, tokens for gap were added. Thus each space adjoining a target token can also be classified as `OK` or `BAD`, denoting a missing word in the latter case. For N target tokens, $2 \cdot N + 1$ binary values should be generated. This is illustrated in Figure 1.4.

[17]statmt.org/wmt12/quality-estimation-task.html
[18]statmt.org/wmt08/shared-evaluation-task.html


Source sentence:

use the Video Properties dialog box to change video properties for FLV Video files .

Machine translated sentence:

im Dialogfeld " Videoeigenschaften " können Sie Videoeigenschaften für FLV Video-Dateien ändern .

Quality estimation for tokens:

OK OK OK OK OK OK OK OK OK OK BAD OK OK

Quality estimation for missing words:

OK OK OKOK OKOK OK BAD OK OK OK OK OKOK

Final output:

OK OK OK OK OK OK OK OK OK OK OK OK OK OK OK BAD OK OK OK OK OK OK BAD OK OK OK OK OK

Figure 1.4: Quality estimation tags for tokens and gaps on German sentence translated from English (from WMT19 quality estimation shared task)

1.3.1 QE from machine translation models

Some machine learning systems, especially classifiers, are able to report also the confidence in their output. To our knowledge, quality estimation from the machine translation models themselves is used only in combination with another QE system which use this self-reported confidence as one of the features.

1.3.2 QuEst++

The main pipeline of QuEst++ by Specia et al. (2015) consists of feature extraction and machine learning prediction. This system first extracts features from the input data and then runs a machine learning algorithm for example with scipy's LarsCV (Cross-validated Lasso, using the LARS algorithm) or Conditional Random Fields suite by Okazaki (2007).

A large part of their work is devoted to feature exploration and fine-tuning. There are four groups of features for word-level quality estimation:

- word alignment
- POS of source and target words
- n-gram frequencies in source and target texts
- syntactic, semantic and pseudo-reference binary flags

They are an extension of features for word-level quality estimation used by WMT12-13-14-17-18 for the baseline model.[19] For sentence-level, the features are focused on the relations of tokens in the source and target sentences, e.g. the ratio of such tokens and distribution of parts of speech. Document features incorporate aggregation of sentence-level features as well as features on the discourse level by Scarton and Specia (2016).

Since the system was not designed to provide online results, the consequence is, that especially the feature extraction part is not optimized and is quite slow. It can handle only ten sentences at a time, so larger inputs have to be split into multiple batches. At the time of deployment there were two bugs, for which we opened pull requests.[20]

[19]quest.dcs.shef.ac.uk/quest_files/features_blackbox_baseline_17

[20]github.com/ghpaetzold/questplusplus/pull/45 and github.com/ghpaetzold/questplusplus/pull/46

1.3.3 DeepQuest

DeepQuest (Ive et al., 2018) takes a neural approach to quality estimation and is capable of performing on any language pair. The toolkit offers two architectures.

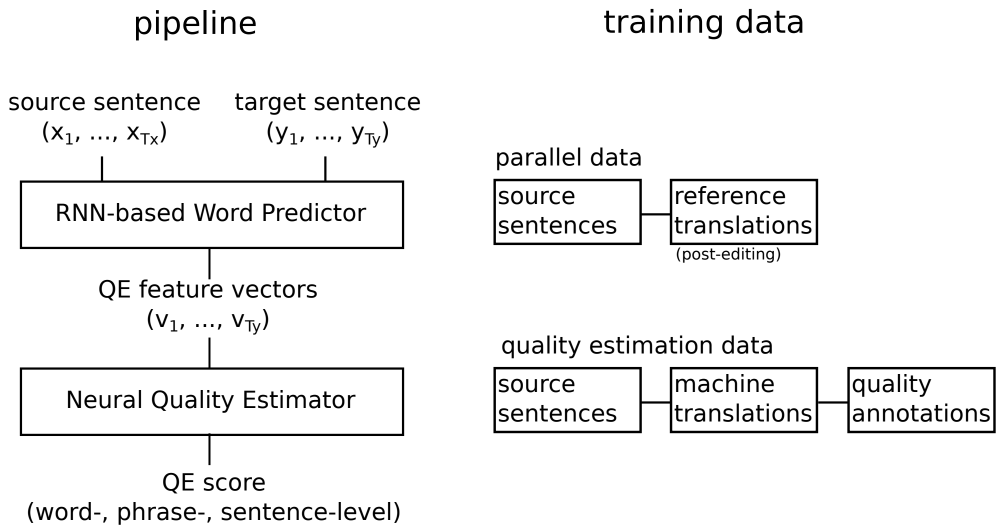


Figure 1.5: Predictor-Estimator QE model pipeline and type of training data, adapted from Fig. 1 from (Kim et al., 2017)

Predictor-Estimator architecture (Kim et al., 2017) consists of two stages of training. In the first one, a predictor (encoder-decoder RNN) is trained on parallel data to predict words based on their context representations. Feature vectors from this network are then passed to the estimator, which is trained on quality estimation data. This delegates the issue of feature engineering onto machine learning itself. The architecture is illustrated in Figure 1.5.

The other architecture implemented by DeepQuest is biRNN. A sequence is passed into RNN both in the original ordering as well as in reverse. The results are then concatenated and propagate further into the rest of the network. The network can then make use of not only the preceding item in the sequence but also the subsequent one.

Models of both architectures can be executed on document-level, sentence-level, phrase-level and word-level.

Ive et al. (2018) reported better results in most experiments on WMT17 with the Predictor-Estimator than the baseline (QuEst++). Performance of the biRNN architecture is still better than the baseline and does not significantly lack behind the Predictor-Estimator. They also stress the difference of performance of implemented systems on SMT and NMT data, with the later resulting in worse results, presumably because the error is less predictable in the latter case.

Despite the better results, we found the implementation difficult to work with because of several bugs. Even though some of the bugs we reported were fixed, the latency of this tool was still too high for online purposes.

1.3.4 OpenKiwi

OpenKiwi (Kepler et al., 2019) implements three quality estimation models: QUality Estimation from ScraTCH (Kreutzer et al., 2015), NeUral Quality Estimation (Martins et al., 2016) used for WMT19[21] baseline and Predictor-Estimator (Kim et al., 2017). In addition OpenKiwi implements stacked ensemble as proposed in Martins et al. (2017).

QUETCH is a linear combination of baseline features and custom neural network. The architecture of the later is shown in Figure 1.6. For each target token t_i an aligned source token $s_{a(i)}$ is considered. Then windows of size three are concatenated: $(t_{i-1}, t_i, t_{i+1}, s_{a(i)-1}, s_{a(i)}, s_{a(i)+1})$ and passed through a lookup-table (pretrained word2vec). The resulting vectors are then passed through a densely connected layer and finally through an output layer, which outputs `OK` or `BAD`.

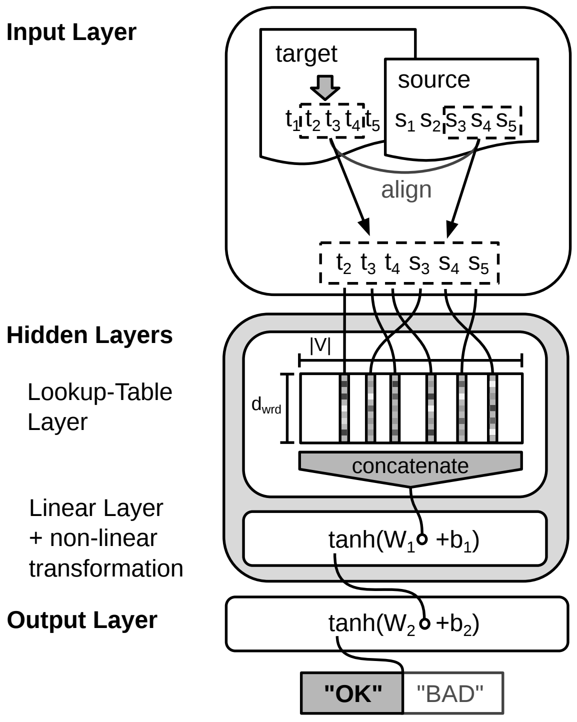


Figure 1.6: QUETCH model architecture, from Fig. 1 in (Kreutzer et al., 2015)

NuQE is very similar to QUETCH in most respects. In addition, POS tags are concatenated and passed through a POS embedding table and finally, two layers bidirectional gated recurrent units are appended (with two extra dense layers in between).

We opted for the Predictor-Estimator architecture because even though it requires pretraining, it does not consume many resources compared to the stacked ensemble. Except for the ensemble, it also provides the best results as shown in Kepler et al. (2019). OpenKiwi, in general, proved to be faster, more robust and easier to use[22] than DeepQuest. Because of this, the experiment was conducted with this quality estimation backend.

[21]statmt.org/wmt19/qe-task.html

[22]OpenKiwi is distributed via pip pypi.org/project/openkiwi/ as a Python module.

1.3.5 Evaluation

Since the output of sentence-level and phrase-level quality estimation is usually HTER, there are several metrics for quality estimation systems. They are measured over the whole dataset.

- Root Mean Squared Error
- Mean Average Error
- Pearson's correlation
- Spearman's rank correlation

For word-level quality estimation, the output is a sequence of probabilities, which can be, for a given threshold, transformed into a sequence of two classes `OK` and `BAD`. Since the distribution of `OK` and `BAD` for most of the sentences is unbalanced, accuracy could be cheated by predicting always `OK`. To balance precision and recall, F measure with respect to `OK` and `BAD` is used. To incorporate both, $F_{MULTI} = F_{OK} \cdot F_{BAD}$ is usually used as a metrics for WMT quality estimation task.

1.4 Source complexity

The task of estimating source complexity is not rigidly defined and has not been thoroughly explored. In the context of machine translation, we can think of it as finding patterns in a sentence which are hard to translate by MT systems. For our purposes, it is beneficial to know which parts of the source sentence are challenging to translate, so that users speaking only the source language can reformulate that specific segment.

Lexical choice

One of the most straightforward ideas would be to look at individual words in the source sentence and describe the probability of them being translated correctly. This can be done, for example, by searching for that word in the data the specific MT system used for training. Subword units can help with unseen words, but we can still hypothesize, that if a word has never been seen by an MT system, it will be difficult to translate.

Another approach to recognize problematic source words would be using word alignment and word-level quality estimation. A QE system gives a score to every translated word. These values can be mapped back to the source sentence by word alignment. This approach was chosen for Ptakopět because it could be done with already existing tools. The implementation is discussed in Section 4.1.

Syntactic structures

For most MT systems it is not a single unknown word which worsens the translation quality but syntactic structures. This has been described extensively in Choshen and Abend (2019). Their claim is that long distance dependencies are still a massive problem for translation quality.

The work of Niehues and Pham (2019) focuses not only on target tokens QE but also on confidence annotation of source tokens. Their methods are based on complex similarity measures between the given source sentence and training data.

2. Previous work

This thesis is focused on the latest version of Ptakopět. The previous two versions (old-1 and old-2) were vastly different and created as part of two other classes taught at Charles University. Summaries of their respective goals, functionalities, and conclusions follow. Both projects are archived in a combined GitHub repository.[1]

We also comment on the adoption of quality estimation systems in the industry and publicly available services Section 2.3.

2.1 Ptakopět old-1

2.1.1 Introduction

The first version was developed as a semestral assignment for class Competing in Machine Translation led by Ondřej Bojar. The goal was to explore the issue of outbound translation for daily internet usage (e.g. filling out forms in websites of a foreign language). Because of this intended usage, it was designed as a browser extension compatible with major browsers (tested on Firefox, Google Chrome, and Opera).

It used to be available on Chrome Web Store and Add-ons for Firefox, but we removed it from these places as this version soon became deprecated.

2.1.2 Usage

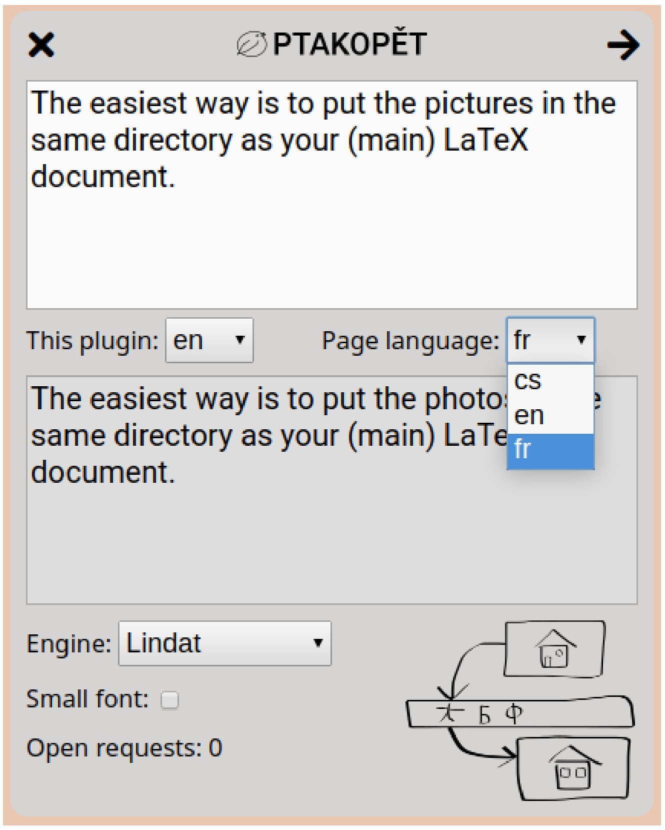
The core functionality was to display backward (round-trip) translation so that users could check, for example, whether the verb tense was changed or if the backward translation matches the original sentence in its meaning. This basic functionality, which remained in future version, gave this project the name of Ptakopět (**p**řeklad **t**am **a ko**ntrolně z**pět**). Ptakopět old-1 ran as a small plugin located, if active, in the top left or top right corner of the page. The plugin could be used either as a browser extension (users install this extension) or as a part of a web page (web admin inserts a loading script into their page).

The plugin shown in Figure 2.1, contains two main textareas. The top one expected input in the language of the user. The bottom one eventually contained the backward translation. The translation was in the active input element (on the webpage). This process of three text elements (input, translation and backward translation) was hard to communicate to users, so the window also contained an explanatory diagram in the bottom right corner, as seen in Figure 2.1.

The intended workflow was to write text to the top textarea and validate against the backward translation in the bottom one. During this process, the translated text appeared in the selected area on the web page. The active target input element was changed every time a possible element (all textareas and text inputs) got focus.

The plugin also contained other miscellaneous control elements, such as translator backend selector, small font checkbox and indicator of open requests.

[1]github.com/zouharvi/ptakopet-old


PTAKOPĚT

The easiest way is to put the pictures in the same directory as your (main) LaTeX document.

This plugin: en

Page language: fr

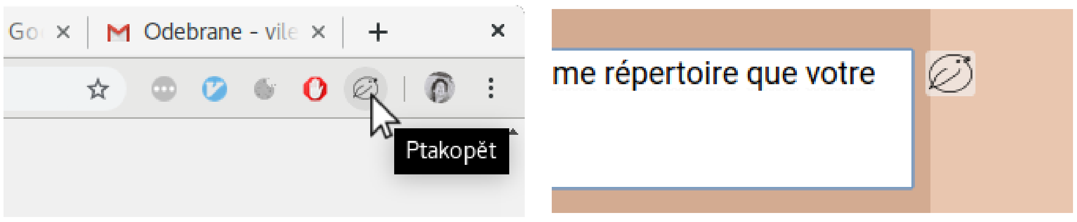
The easiest way is to put the photo[s in th]e same directory as your (main) LaTe[X] document.

Engine: Lindat

Small font:

Open requests: 0

Figure 2.1: User interface of the Ptakopět old-1 browser extension

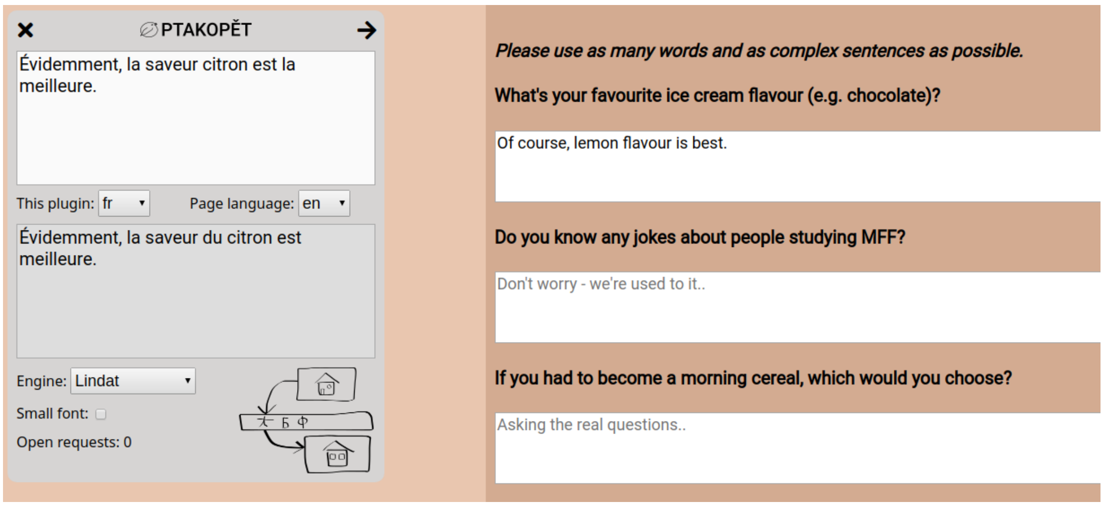


(a) Toolbar icon (b) Input element icon

Figure 2.2: Two ways of launching Ptakopět old-1

The Ptakopět window could be invoked in multiple ways: by clicking the toolbar button, as shown in Figure 2.2b, by launching it from the context menu (right mouse button) or by clicking an icon, that appeared next to all web page text input elements.

In Figure 2.3, the user selected the first web page input element, hence making it active to Ptakopět and wrote a text in their native language (French) to the first Ptakopět input. The backward translation removed a definite article for an uncountable noun; otherwise, the sentence matches the original. The English translation of the input sentence then appeared in the target textarea, as expected. Should the user now edit the English text, the translation would appear in the bottom Ptakopět textarea. Writing into the top Ptakopět input field would overwrite the manual changes, as hinted in the bottom right diagram in the Ptakopět window.


PTAKOPĚT

Évidemment, la saveur citron est la meilleure.

This plugin: fr
Page language: en

Évidemment, la saveur du citron est meilleure.

Engine: Lindat
Small font:
Open requests: 0

Please use as many words and as complex sentences as possible.

What's your favourite ice cream flavour (e.g. chocolate)?

Of course, lemon flavour is best.

Do you know any jokes about people studying MFF?

Don't worry - we're used to it..

If you had to become a morning cereal, which would you choose?

Asking the real questions..

Figure 2.3: Ptakopět helps a French-speaking user with filling in an English form

2.1.3 Technical details

Ptakopět old-1 was written entirely with basic HTML, CSS and JS stack with some usage of jQuery. There was no packaging or build system. Part of the codebase dealt with abstraction on top of different browser plugin APIs and the differences between content and background script execution. Most major browsers support the WebExtensions API of W3C, but there are some differences between browsers.

The rest of the code dealt with DOM manipulation and mostly with handling translation requests. Two translation backends were used: Khresmoi (Dušek et al., 2014) and LINDAT Translation (Popel, 2018), because of their ease of use and availability. At the time of deployment, LINDAT Translation supported translations between English, French, and Czech, while Khresmoi supported English, French, Czech, German, and Spanish.

Four months after the finished project was demoed, the API which Ptakopět used for communication with LINDAT Translation was deprecated and two months after that the Khresmoi translation service shut down. Instead of making necessary fixes, this project was abandoned for a newer version and the plugin was removed from the public listing.

2.1.4 Conclusion

This project was completed and met all criteria regarding quality and functionality, even though the extension was not usable on all websites, which stemmed from the extension architecture.

It was demoed during the Open Days at Charles University Faculty of Mathematics and Physics 2019, although only few visitors tried it. The demo page is not hosted publicly anymore, but source code is available.[2] A part of the demo page is visible in Figure 2.3. The demo included a simple form with random personal questions in a foreign language, which visitors were to fill. The source language in the demo was Czech and, unfortunately, the foreign language was

[2]github.com/zouharvi/ptakopet-old/tree/master/old-1/dod_ufal

English, so it was hard to demonstrate the issue of outbound translation fully, as most people know English to at least some extent.

A four-page report was submitted.[3] This first attempt for an outbound translation tool provided us with findings as what to avoid and what to focus on in the next version, Ptakopět old-2.

2.2 Ptakopět old-2

2.2.1 Introduction

The second version of Ptakopět aimed to extend the functionality of the first version, while improving on usability. One of the downsides of the Ptakopět old-1 was the disunified behaviour and lack of consistency across different websites. The user interface proved to be too complex to work with and hence we opted for a more traditional approach. The entire application would then be hosted on a separate web page with no interaction with other web pages. This form of serving public machine translation is similar to the one by Google, Microsoft, DeepL and other online services.

In addition to the major change of moving from browser plugin to a standalone web page, we decided to add visual quality estimation cues in the form of word highlighting (from word-level quality estimation models).

Ptakopět old-2 was created to meet the requirements of class Semestral Project and has a corresponding specification document.[4] It was again supervised by Ondřej Bojar.

2.2.2 Usage

There are generally two use cases for the second version: Outbound translation and user translation quality estimation.

Outbound translation

In outbound translation the user tries to translate text to and validate produced translation in a language their are not familiar with.

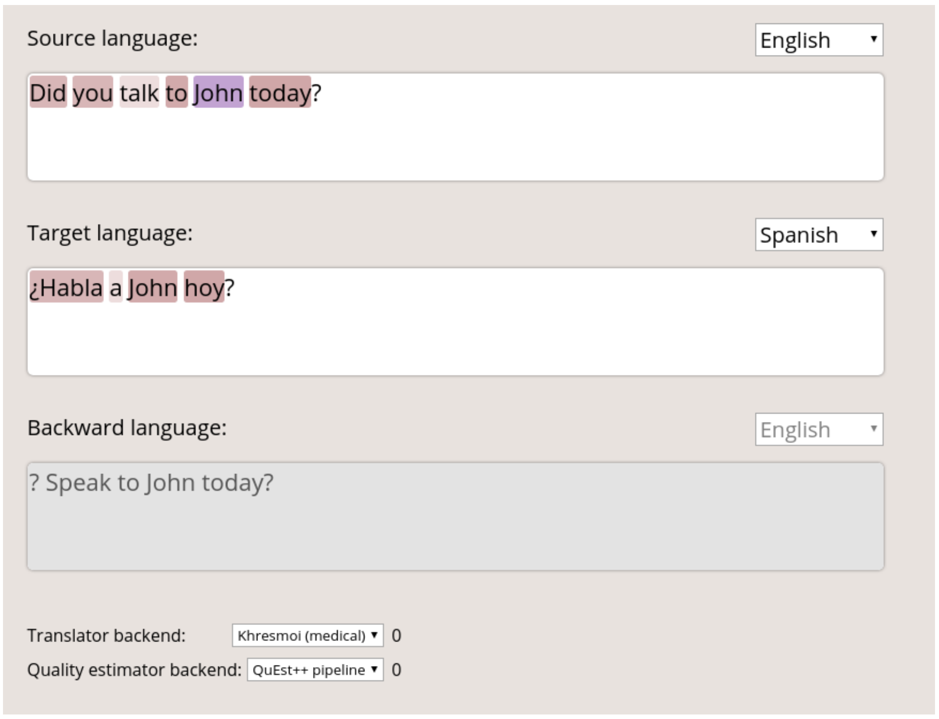
To perform outbound translation, the user selects the source and target languages from the select box, then writes text in the source language input window (first textarea in Figure 2.4). Problematic words in the source and target text can be seen (more intense colouring means worse quality; purple signifies, that the particular word was not translated) as well as backward translation of the already translated text (third textarea in Figure 2.4).

User translation quality estimation

The user could then follow up from the previous use case with a more agile workflow and dynamically edit the translated text. They would see the quality estimation of their own translation, as well as backward translation.

[3]github.com/zouharvi/ptakopet-old/blob/master/old-1/meta/report.pdf

[4]github.com/ zouharvi/ ptakopet-old/ blob/ master/ old-2/ meta/ Ptakopět v2 - specification.pdf


Source language: English

Did you talk to John today?

Target language: Spanish

¿Habla a John hoy?

Backward language: English

? Speak to John today?

Translator backend: Khresmoi (medical) 0

Quality estimator backend: QuEst++ pipeline 0

Figure 2.4: Example usage of Ptakopět old-2. The more red the word highlighting, the worse the estimated translation quality. Purple highlighting was used for words, which were not translated.

In this case the quality estimation is not exactly the same as the task defined by WMT, because the provided translation is not a product of an MT system, but of a human user. With current QE models we do not think that this can be used reliably.

2.2.3 Technical details

The second version was also written in plain HTML, CSS, JS + jQuery (frontend) and Python 3 (backend request handler), but contains many interactions (through system pipes) to other frameworks written in Python 2, Java, Perl and C++. The frontend tech stack is used contrary to modern approaches to web development, but the limited scale of this project and focus on other aspects allowed for it.

The backend ran at servers provided by ÚFAL and responded to requests for either quality estimation or word alignment. The same two translation backends were used: Khresmoi (Dušek et al., 2014) and LINDAT Translation (Popel, 2018).

Quality Estimation

There were two quality estimation backends: DeepQuest for English-German and QuEst++ for English-Spanish.

Highlighting parts of texts seems trivial at first, but taking into consideration, that the text must be editable and that the highlighting must work on different zoom levels, browsers and on mobile, it soon becomes complex. Finally, we opted for a jQuery plugin developed originally by Will Boyd,[5] which we forked,[6] as some changes to the internal workings of this plugin were necessary. We also wanted to highlight words, which were not translated. To do this, we highlighted all words, which were mapped via alignment to a word of the same form. Such an occurrence is displayed in Figure 2.4.

Alignment

Fast align[7] was used for the alignment backend, as it was recommended in the QuEst++ documentation. Word alignment was necessary for the highlighting itself as well as for QuEst++.

2.2.4 Issues

We later found that we used Fast align incorrectly, applying the unsupervised word-alignment method only to the single input sentence pair given. Since the model lacks any lexical knowledge, it thus essentially provided a simple diagonal alignment most of the time.

Because of missing Access-Control-Allow-Origin header on the Khresmoi translator backend, a proxy was added. This was unavoidable but a wrong decision since it is considered unsafe and software such as Avast would notify the users.

At the time of the deployment, the web hosting server had a valid SSL certificate but made requests to unsafe servers, so it had to be served over HTTP.

Another issue was the lack of a clear indication of supported language pairs. Ptakopět allowed users to use a quality estimation model even for language pairs that the model did not support.

Setting up the server was also a difficult task, as proper replicable setup scripts were not written.

2.2.5 Conclusion

While Ptakopět old-2 passed as a semestral project, there were many ideas on how to improve the experience and project quality. In Ptakopět old-2 only English-Spanish and English-German language pairs were supported by QuEst++ and DeepQuest respectively. Especially DeepQuest could be used with more language pairs if relevant data were provided. DeepQuest also reported only binary values `OK` or `BAD` at that time, but continuous confidence values from 0 to 1 existed inside. Extracting them would have provided more information to the user.

During development, the whole project accumulated significant technical debt. This was due to decisions such as writing in JavaScript without a framework, not having more structured backend and missing setup scripts.

Both technical and user documentation[8] was written on this version.

[5]github.com/lonekorean/highlight-within-textarea
[6]github.com/zouharvi/highlight-within-textarea
[7]github.com/clab/fast_align
[8]ptakopet.vilda.net/docs/old-2

2.3 Industry

As far as we observed, modern outbound translation workflow is condensed to roundtrip translation done manually by users (switching the language direction and copy-pasting the translated text).

Quality estimation is used in translation companies mostly to minimize post-editing costs. Despite that, QE cues are missing in most of the mainstream public translation services, such as Google Translate[9] (provides alternatives to words and shows their usage frequencies), Microsoft Bing Translator[10] or DeepL[11] (provides alternatives to phrases).

Memsource

The company Memsource, which specializes in cloud-based translation environment with tools like translation memory and terminology management, is also deploying quality estimation models, to minimize post-editing cost. For example, if a machine-translated sentence receives a full score, then there is no need to pay professional human translators to verify the translation. Even though they are developing their models in-house and closed-source, they disclosed some details in one of their blog posts.[12]

Notable is their way of presenting the quality estimation data. Instead of displaying the percentage score in any form (e.g. highlighting), they approach this as a classification problem with the classes: 100%: probably perfect translation, 95%: possibly requires minor post-editing, 85%: will require post-editing, no score: needs to be manually checked. They focus on the phrase-level and sentence-level quality estimation.

Unbabel

The company Unbabel, which delivers machine translation solutions, is also developing[13] a QE pipeline. As opposed to Memsource, most of their QE research is public, such as Martins et al. (2016) and Kepler et al. (2019). They also make use of both word-level and phrase-level quality estimation. One of their components, OpenKiwi, is also part of Ptakopět and is described in Section 1.3.4.

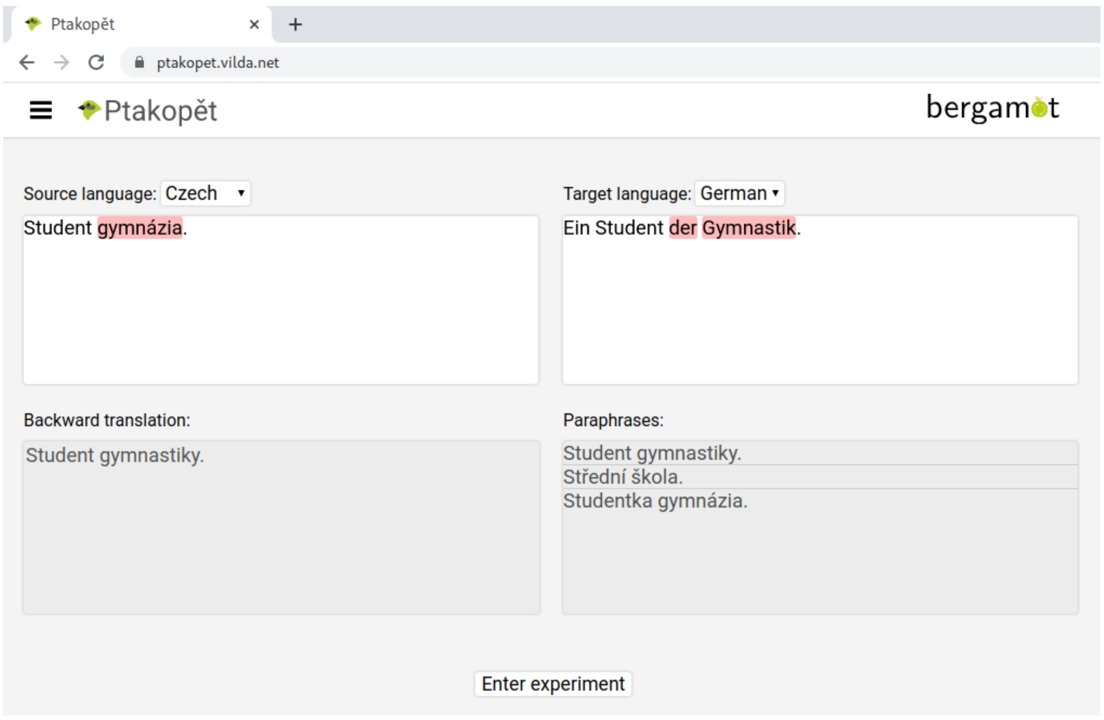
[9]translate.google.com
[10]bing.com/translator
[11]deepl.com/en/translator
[12]memsource.com/blog/2018/10/01/machine-translation-quality-estimation-memsources-latest-ai-powered-feature/
[13]unbabel.com/blog/unbabel-translation-quality-systems/

3. Final Ptakopět

The final version of Ptakopět is publicly accessible from ptakopet.vilda.net. The goals of this version were to improve the overall user experience, make the system more modular, scalable and robust and also to make it ready for experiments on human annotators.

3.1 Overview

Ptakopět helps users with outbound translation. The full layout is displayed in Figure 3.1. It is composed of four blocks (modules): input, translation, backward translation, paraphrases. Source and target languages can be selected in the drop-down menus at the top of the first two modules. After writing text in the first textarea, it is translated to the second textarea (also editable), then backward translation, quality estimation and paraphrases are generated.



Source language: Czech
Student gymnázia.

Target language: German
Ein Student der Gymnastik.

Backward translation:
Student gymnastiky.

Paraphrases:
Student gymnastiky.
Střední škola.
Studentka gymnázia.

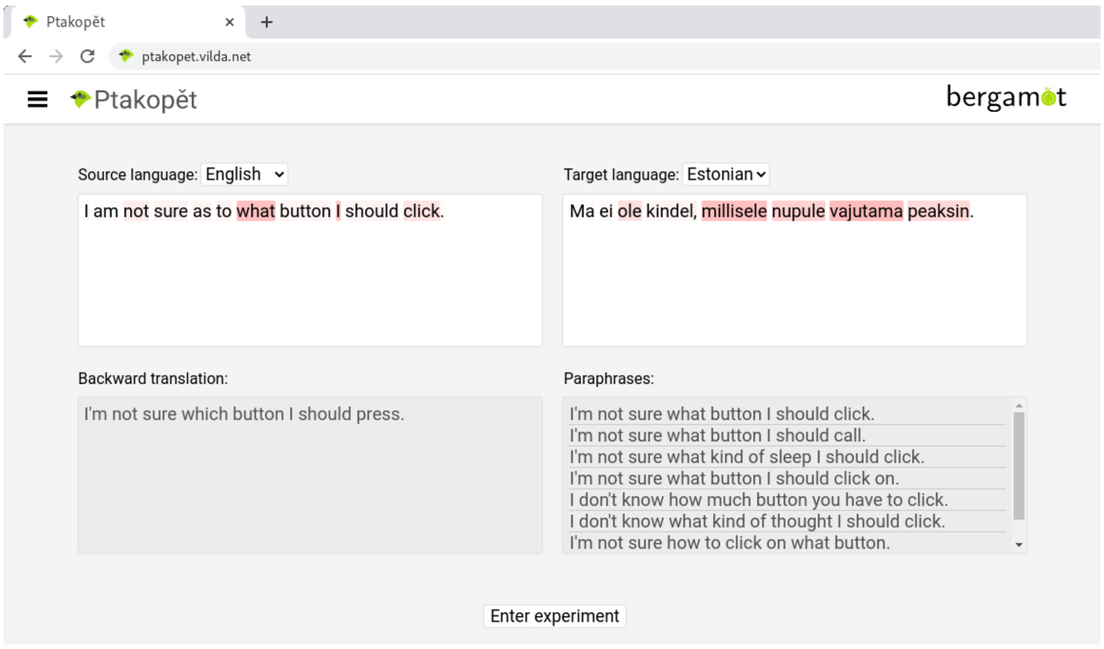
Figure 3.1: Ptakopět is used to translate a simple Czech noun phrase to German. QE highlights parts of both source and target, that were translated incorrectly.

In Figure 3.1 it is seen, that the source was mistranslated by the red highlighting (on both source and target), but also by the backward translation. Paraphrases (or source variations) are displayed in the second to last block. In the case of simple noun phrases, they are irrelevant, but they are useful for more complex inputs, as shown in Figure 3.2. In those cases, the user might want to reformulate (mostly simplify) specific parts of their input, so that the MT system can produce a better translation.

Unfortunately, we found the quality estimation not to be reliable enough. It usually works only with concise sentences or simple noun phrases. This can be seen when comparing the highlighting in Figure 3.1 and Figure 3.2.

The translation can be then also edited. A common error of MT systems is that they try to translate named entities. This is easy to recognize even in languages the user is not proficient in. They may then choose to fix this error in the translation textarea and check that all went well in the backward translation block. Typing anything in the source textarea would then rewrite these manual changes.

The backend for each service can be selected in the settings burger menu, hidden behind the burger icon in the top left corner in Figure 3.1. The expanded settings menu is shown in Figure 3.3 in the last block. Changing any language or backend selection causes a cascade so that relevant information is recomputed and showed. Pending requests to a specific backend are signalized by a loading indicator next to every module block.



I am not sure as to what button I should click.

Ma ei ole kindel, millisele nupule vajutama peaksin.

I'm not sure which button I should press.

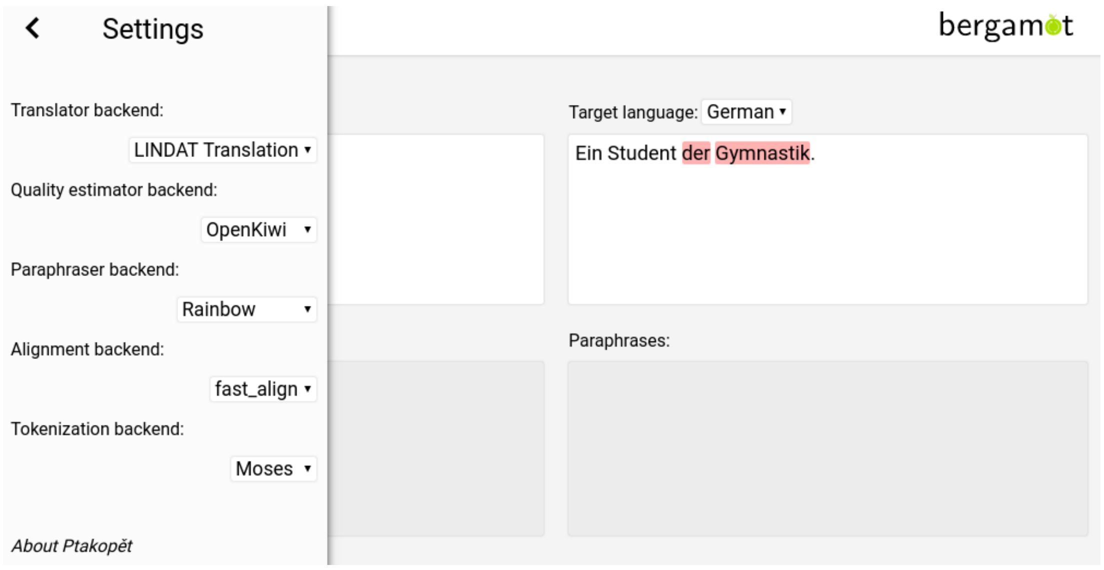
I'm not sure what button I should click.
I'm not sure what button I should call.
I'm not sure what kind of sleep I should click.
I'm not sure what button I should click on.
I don't know how much button you have to click.
I don't know what kind of thought I should click.
I'm not sure how to click on what button.

Figure 3.2: Ptakopět is used to translate a complex English sentence to Estonian. User may opt to reformulate the input according to the paraphrases suggestions.

3.2 Settings overview

Backend settings are freely accessible, even though we do not expect most users to interact with them. Some particular settings are not intended to be used for outbound translation but are included either as placeholders or for debugging or presentation purposes. In this section, we aim to give a brief overview of their respective roles and origins. Backends, which were not created or setup by the author of this thesis, are explicitly mentioned.

An exclamation mark is shown next to the specific module in case the selected backend is incompatible with a given language pair. This warning is seen in Figure 3.6.



Ein Student der Gymnastik.

Figure 3.3: Expanded settings burger menu in Ptakopět, which allows the users to change service backends.

Translator

The main translator backend is **LINDAT Translation** (Popel, 2018), which provides translations between Czech, English, French and German. **Strong EN-CS** is just a shadow copy of this backend. Due to a collaboration with Tartu on the Bergamot project, we also added English↔Estonian backend **Neurotõlge**[1] and **Avg EN-ET**. For future experiments, we also added **Weak EN-CS** and **Strong EN-CS** for English↔Czech translation at various quality. There are also two placeholder backends, which evaluate client-side. **Identity** only copies the input and **None/Manual** does nothing so that the user can input their own texts without being interrupted. All of the non-local backends are not part of the work on this thesis.

Quality Estimation

Our instances of **OpenKiwi** and **DeepQuest** backends support Czech→German and English→German quality estimation. The running instance of **QuEst++** supports English→Czech quality estimation but is of very low quality. Collaborators from the Bergamot project who specialize in quality estimation provided two backends **Sheffield EN-ET** and **Sheffield EN-CS**. For presentation purposes, the QE values can be inputted manually by selecting the **Manual** backend. It can also be set to **Random**, which assigns each target word a random value between 0 and 1. This fake backend is good for analyzing alignment because one can then easily see what the words map to. The highlighting can be turned off by selecting **None**.

[1]neurotolge.ee

Paraphraser

The first paraphraser is **LINDAT Mock**, which relies on round-trip translation with LINDAT Translation. It is extremely unoptimized and is there only for testing purposes. The second paraphraser backend, **Rainbow**, works on a similar principle but is more elaborate and faster. This backend is a transformer-based and is the work of Matúš Žilinec.[2] The paraphraser module can also be turned off by selecting **None** as the paraphraser backend.

Alignment

Alignment requests for most languages are by default relayed to **fast_align Ubuntu** on server-side. There also exists a special backend **fast_align Michal** which was setup by a colleague Michal Novák for future experiments with Ptakopět on English-Estonian and English-Czech language pairs. The alignment can also be evaluated locally by **Diagonal** placeholder backends, which for sentences of M and N tokens generates alignment $\{(i, j) : 0 \leq i \leq M, 0 \leq j \leq N\}$. The **None** turns off the alignment altogether.

Tokenization

The **Moses** tokenizer is the main tokenization backend and is very robust. There are two alternatives, which evaluate client-side. The first one, **Spaces**, is just splitting by single spaces, while the second one, **Local**, uses a more complex tokenization scheme.

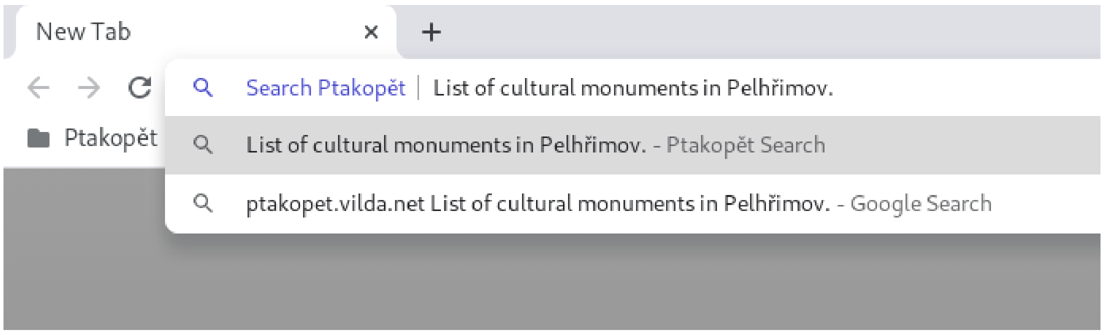
3.3 Miscellaneous

Omnibox OpenSearch

Google services and most notably Google Translate use omnibox OpenSearch to improve the user experience. This was also implemented in Ptakopět, so instead of having to go to the web page and then start typing the source text, it can be typed in the address bar after pressing the `TAB` key. The web page is then loaded and the input pasted in the input textarea.

This makes it more convenient for using Ptakopět to quickly translate pieces of texts. An example omnibox input and the result is shown in Figure 3.4.

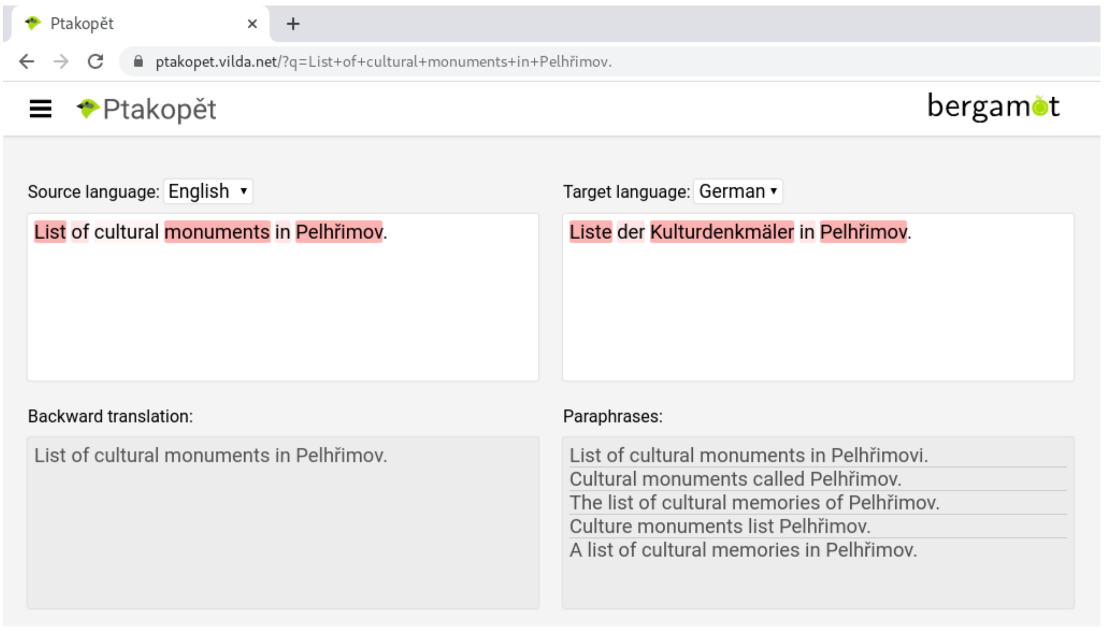
[2]github.com/mzilinec/paraphrase-server


New Tab

Search Ptakopět | List of cultural monuments in Pelhřimov.

List of cultural monuments in Pelhřimov. - Ptakopět Search

ptakopet.vilda.net List of cultural monuments in Pelhřimov. - Google Search


ptakopet.vilda.net/?q=List+of+cultural+monuments+in+Pelhřimov.

Source language: English

List of cultural monuments in Pelhřimov.

Target language: German

Liste der Kulturdenkmäler in Pelhřimov.

Backward translation:

List of cultural monuments in Pelhřimov.

Paraphrases:

List of cultural monuments in Pelhřimovi.
Cultural monuments called Pelhřimov.
The list of cultural memories of Pelhřimov.
Culture monuments list Pelhřimov.
A list of cultural memories in Pelhřimov.

Figure 3.4: Example usage of omnibox OpenSearch input. The text appears in the main input textarea once the user hits `ENTER`. The default language and backends settings are used.

URL parameters

Ptakopět also supports other URL GET parameters:

- `userID` logs the user to the experiment by the supplied userID value. The experiment is discussed in Section 4.2 and Chapter 5.
- `q` pastes the text query into the input box. This is to support the omnibox OpenSeach functionality.
- `p` sets a given settings profile to other than `default`, e.g. `pilot`, `edin`, `csen` or `sao`. This is helpful when one wishes to provide a quick link with all settings prepared. These exact settings profiles were used for different presentation purposes.
- `source` sends this extra value to the start log.
- `test` used for testing, described in Section 4.1.

The redirect URL for profile `pilot` with the user `testuser` and source information `online-ad` can then look like:

```
ptakopet.vilda.net/?p=pilot&userID=testuser&source=online-ad
```

Error masking in backtranslation

When using the same data for both forward and backward MT, an error can be introduced in forward translation but removed in the backward translation. After experimenting with Ptakopět we found several examples described in Figure 3.5. All can be tested in the live system using the **LINDAT Translation** backend.

svírá úhel	$\xrightarrow{\text{de}}$	Er schließt den Winkel.	$\xrightarrow{\text{cs}}$	Zavírá úhel.
svírá úhel	$\xrightarrow{\text{fr}}$	Sait l'angle	$\xrightarrow{\text{cs}}$	Zná úhel
svírá úhel	$\xrightarrow{\text{en}}$	grips the angle	$\xrightarrow{\text{cs}}$	svírá úhel

Figure 3.5: Example of error masking in backward translation in English MT compared to German and French MT in which the error is revealed.

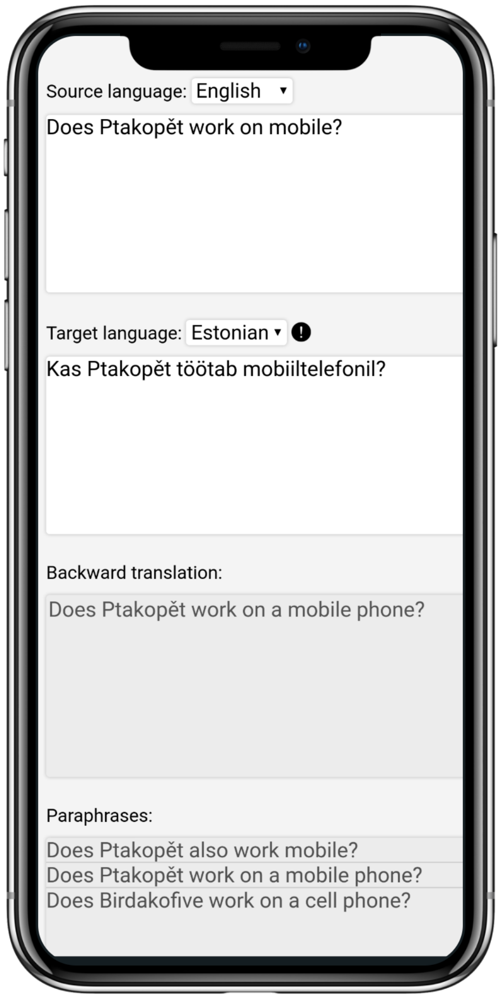
The German and French MT in Figure 3.5 introduced an error in forward translation, but the backward translation was accurate and thus, the user could recognize this and reformulate the input. However, in the case of the English machine translation, an error is still introduced in the forward translation but the backward translation removes this error. The backward translation could, in fact, be accurate and this masking could be a result of sense ambiguity of the word “svírat”. Nevertheless, the user could then get a false sense of a correct translation, which is undesirable.

Platforms

Ptakopět was tested to run properly on Edge 80, Chrome 83, Firefox 75 and Opera 66.

We wanted Ptakopět to be accessible from as many devices as possible and not just desktops. The frontend was designed to adapt to almost any screen size, most notably mobile. The layout is then changed to a single column, as shown in Figure 3.6.

We found Ptakopět to be very usable on mobile. The only issue was that occasionally deleting of already written text could not be done by holding the backspace key. This is due to the highlighter dependency. Ptakopět was also tested to work without any issues on very computationally limited device, namely Samsung Smart TV with Tizen OS.



Does Ptakopět work on mobile?

Kas Ptakopět töötab mobiiltelefonil?

Backward translation:

Does Ptakopět work on a mobile phone?

Paraphrases:

Does Ptakopět also work mobile?
Does Ptakopět work on a mobile phone?
Does Birdakofive work on a cell phone?

Figure 3.6: Mockup of Ptakopět in single column layout on phone.

4. Implementation

The whole project is split into two git repositories hosted at GitHub. The first repository[1] contains the code for frontend, experiment data as well as some miscellaneous files regarding the whole project. The second one[2] is focused strictly on the server, which is not the main focus of this thesis but is also described for completeness.

In this chapter we first describe the frontend, then the experiment architecture from a technical perspective (the experiment itself is the focus of Chapter 5) and then the backend. See documentation in Appendix B which describes all the implementation details as well as guides on how to build and setup the whole project.

4.1 Frontend

The web frontend is written in TypeScript because of the great scalability properties. DOM manipulation is done mostly with jQuery and the output is packed into one JavaScript file using Webpack. Packages are managed with npm (usually contained in the NodeJS package).

The diagram in Figure 4.1 shows the overall object structure of the most important objects. These objects are: `AsyncMessage`, `Translator` (and its derivatives), `Estimator`, `Aligner`, `Paraphraser`, `Tokenizer`, `Highlighter` and `Throttler`.

The task of `TranslatorSource` (and similarly for `TranslatorTarget`) is to translate the source sentence. Then `Estimator` has to assign word-level quality estimation scores for each target token. Finally, a word-level alignment must be computed in `Aligner`. At the end, the QE is rendered using `Highlighter`. Parallel to that, `Paraphraser` produces paraphrases for the source input and displays them.

Each of these computations could take place in the browser[3] (Ptakopět includes some such mock-up solutions, see Section 3.2), but the standard procedure is to relay the translation, estimation, paraphrase, alignment and tokenization requests to some server.

AsyncMessage

Since there can be multiple such requests of one type in one second, it is possible to get into a race condition. This is what happened in the Ptakopět-old projects and vastly worsened the user experience.

Sometimes a translation request, which was sent later than another one, would finish sooner. The new content would be presented, but then the old request would then finish and outdated content would be shown. This is highly undesirable and it is the reason why most of the messaging objects (`Translator`, `Estimator`, `Paraphraser`) make use of `AsyncMessage`. This class assigns a serial

[1]github.com/zouharvi/ptakopet

[2]github.com/zouharvi/ptakopet-server

[3]This one of the goals of the Bergamot project. browser.mt

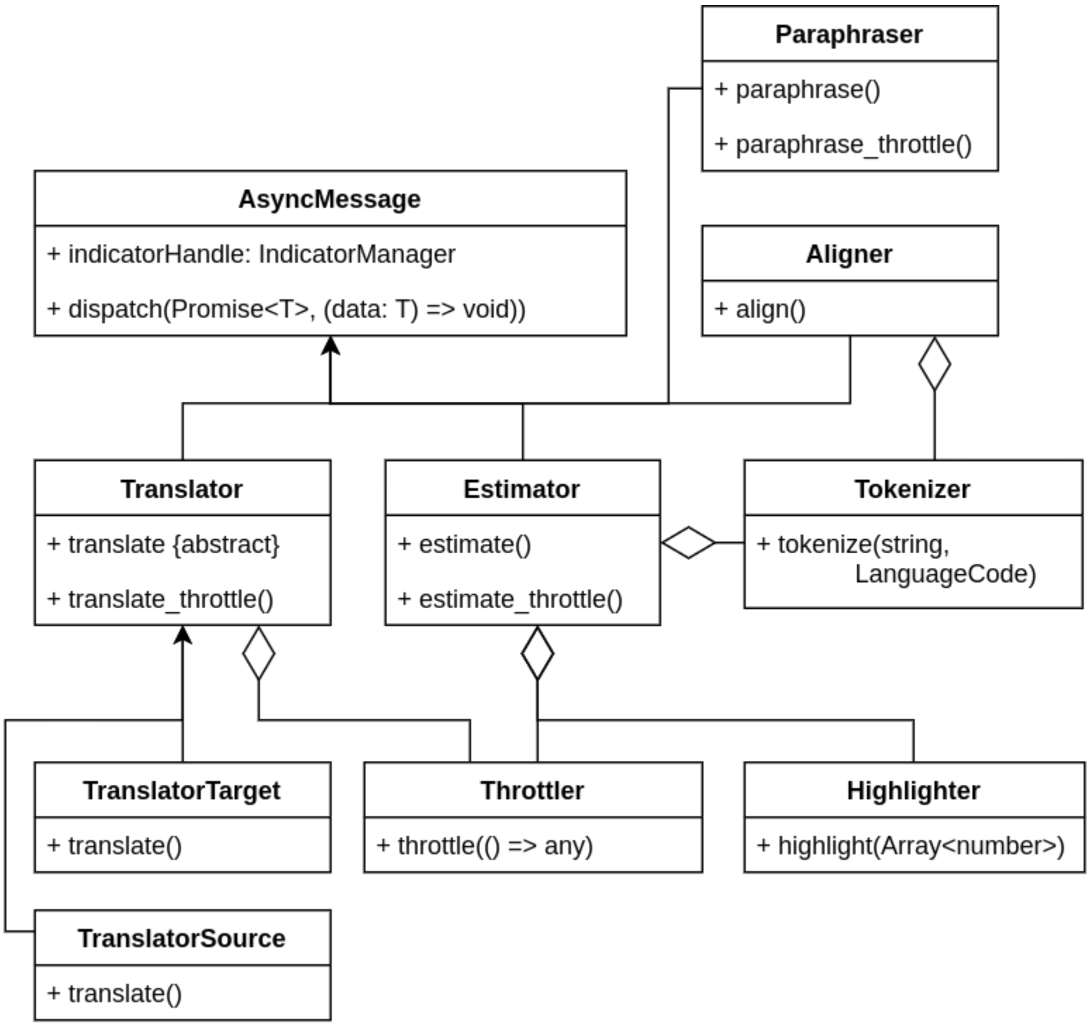


Figure 4.1: Object diagram of main component of the Ptakopět frontend

number to each request and drops delayed incoming responses. By keeping track of active requests, `AsyncMessage` derivatives can easily display an indicator to the user, signalizing whether it is still waiting for a response or not.

The other messaging components, `Tokenizer` and `Aligner`, do not have any callback, which results in any content manipulation (they are almost pure functions) that are invoked solely using `async/await`.

Backends

Each of the main messaging components contains a list of backends so that they can be easily interchanged and tested. These "backends" are not to be confused with the server backend, described in Section 4.3. A backend in this context is just an object, which contains a list of supported language pairs, a name and a function, which for some input returns a promise of the relevant output. A definition of a quality estimation **Random** backend is available in Appendix B.1.2.

Miscellaneous objects

Apart from the main object distinguished in the diagram in Figure 4.1, there are many others. We list a brief overview of their functionality.

- `Throttler` - a simple tool for handling throttling. The input event (on the source input field) fires up every time the user types in a character. It is undesirable to send a translation request after every keystroke, so the throttler sends only the last request and only if no input event happened for some duration. This is achieved by `window.setTimeout` which restarted every time input event happens.
- `Utils`, `TextUtils` - various helper functions, such as generating a set of all possible pairs from a set, language code database, parameter parsing, random string generation.
- `Settings` - globally accessible and simply stores the currently selected backend and language options.
- `SettingsSelector` - backend and language selection in the DOM.
- `SettingsProfiles` - defines common setting setups, which can be later applied (e.g. `default` and `pilot`)
- `Highlighter` - quality estimation DOM element highlighting
- `Tester` - contains functions used for testing

Testing

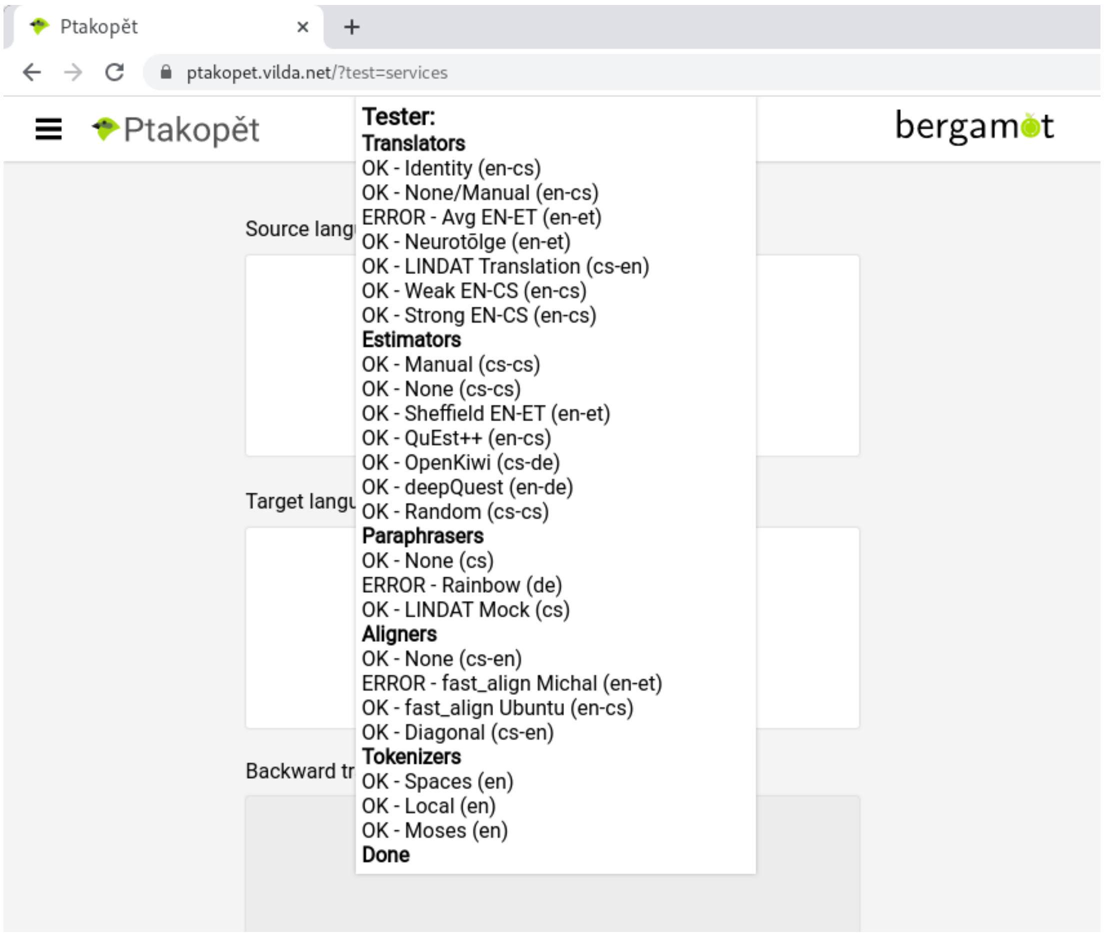
The `Tester` class contains two functions. The first one, `workload`, simulates the user's workflow by changing the source input field in some fixed interval. This was used for debugging memory leaks, such as the one which was present in the pilot study (Appendix B.4.2). It can be invoked by adding the `test=workload` URL GET parameter.

The other function is used for testing the availability of backends. Since Ptakopět encompasses many backends it is necessary to monitor their status. To get a summary of their availability add the `test=services` URL GET parameter. A white window will then appear at the top of the screen. The result is visible in Figure 4.2. This function sends requests to all available backends (even local ones) with a prepared text input and language settings. Usually the backend status (up or down) is manifested on a single call, so checking text and language variations is not necessary.

Highlighting

Given a QE score from 0 to 1 the color is computed as in Equation (4.1). It is substracted from one, because QE of 1 correspond to high confidence and we want to highlight bad scores. It is also scaled down by $\frac{1}{3}$ for the highlighting to not be too disruptive in the user interface. Technical details of highlighting are described in Appendix B.1.2.

$$\text{color}(qe) = \text{RGBA}\Big(1, 0, 0, \frac{1 - qe}{3}\Big) \tag{4.1}$$


Ptakopět

ptakopet.vilda.net/?test=services

Ptakopět

bergamot

Source lang

Target langu

Backward tr

Tester:
Translators
OK - Identity (en-cs)
OK - None/Manual (en-cs)
ERROR - Avg EN-ET (en-et)
OK - Neurotõlge (en-et)
OK - LINDAT Translation (cs-en)
OK - Weak EN-CS (en-cs)
OK - Strong EN-CS (en-cs)
Estimators
OK - Manual (cs-cs)
OK - None (cs-cs)
OK - Sheffield EN-ET (en-et)
OK - QuEst++ (en-cs)
OK - OpenKiwi (cs-de)
OK - deepQuest (en-de)
OK - Random (cs-cs)
Paraphrasers
OK - None (cs)
ERROR - Rainbow (de)
OK - LINDAT Mock (cs)
Aligners
OK - None (cs-en)
ERROR - fast_align Michal (en-et)
OK - fast_align Ubuntu (en-cs)
OK - Diagonal (cs-en)
Tokenizers
OK - Spaces (en)
OK - Local (en)
OK - Moses (en)
Done

Figure 4.2: Results of testing of all Ptakopět backends. Three backends were down at the moment.

Source complexity

After the quality estimation and alignment is computed, every source word receives a possibly empty set of QE scores. This is because the alignment may map the source word to zero or more target words. Different aggregating functions can be chosen to get a single number, for example *maximum*, *minimum*, *average* or *weighted average (by position)*. Furthermore the default QE score has to be assigned to unaligned tokens. We found it reasonable to use the *average* aggregating function as to consider all the provided scores. The score for unaligned tokens was set to 0.9 to only slightly hint that something may be wrong.

4.2 Experiment definition

An experiment can be defined in a single JSON file. The format, details and relevant tools used for generating experiment materials are described in detail in Appendix B.2.

In the experiment definition we need to specify the number of users, their user IDs and the queues. In this context we use the term "baked queues" for every user which is just pre-generated random sequence of stimuli and their configurations. This way it is decided prior to the experiment what stimuli configurations and in what order will a given users encounter them. Baked queues are described more in detail in Section 5.1.4.

A stimuli is just a string containing a HTML code which gets pasted into the webpage. This is flexible enough to allow for both images and texts with highlights. Every stimuli can be also presented with a different configuration, such as a specific backend and set of modules enabled.

4.3 Server backend

The server backend's purpose is to make some of the MT related services (quality estimation, alignment, tokenization) available to the frontend. It is necessary, as most of these services are not publicly deployed, but is not the main focus of the Ptakopět project, nor of this thesis. Even though it was created to be portable in theory, it is not expected to be run on any other server than ours. It is written in Python 3.

To run the server (on `0.0.0.0:80`), launch the `server/run.sh` script. A common practice is to connect to a remote machine via SSH and launch the server. For that, there is a script `server/run_nohup.sh`, which disregards kill signals on user logout.

4.3.1 Architecture and API

The overall backend object architecture is shown in Figure 4.3. The dashed line from QuEst++ to Fast align means that QuEst++ uses this alignment tool also as part of its pipeline. Both GET and POST methods are accepted. The server offers the following API calls. In Listing 4.1 and Listing 4.2, we show example requests together with their responses. The sentence aligner Hunalign is not accessible publicly and is used only for internal purposes.

- `qe/[openkiwi, deepquest, questplusplus]/?` - for word-level QE.
  Requires: `sourceLang` `targetLang`, `sourceText` and `targetText`.
- `align/[fast_align]/?` - word alignment.
  Requires: `sourceLang` `targetLang`, `sourceText` and `targetText`.
- `tokenize/[moses]/?` - sentence tokenization.
  Requires: `text` and `lang`.
- `paraphrase/[mock]/?` - paraphrasing.
  Requires: `text` and `lang`.

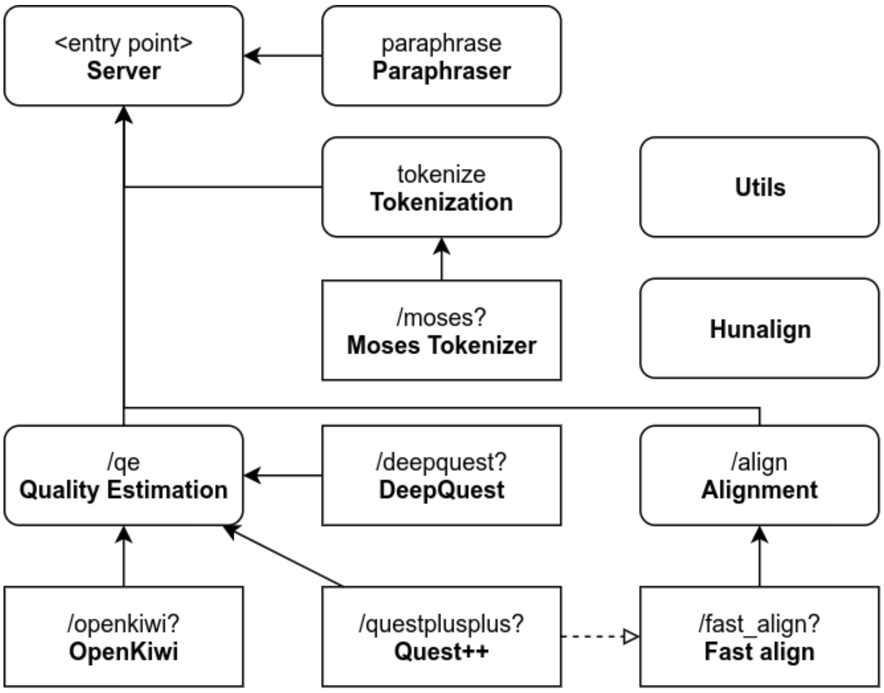


Figure 4.3: Object diagram of main components of the Ptakopět backend

```
qe/deepquest
    ?sourceLang=cs
    &targetLang=de
    &sourceText=Student gymnázia.
    &targetText=Ein Student der Gymnastik.

-> {"status": "OK",
    "qe": [0.8, 0.5, 0.2, 0.5, 1.0]}

tokenize/moses
    ?text=(z.B. Tomaten, Karotten usw.)
    &lang=de

-> {"status": "OK",
    "tokenization": ["(", "z.B.", "Tomaten",
                     ",", "Karotten", "usw.", ")"]}

align/fast_align
    ?sourceLang=en
    &targetLang=de
    &sourceText=Click the mouse button.
    &targetText=Klicken Sie mit der Maustaste.

-> {"status": "OK",
    "alignment": "0-0 0-1 2-2 1-3 3-4 4-5"}
```

List of Listings 4.1: Three examples of Ptakopět backend quality estimation, tokenization and alignment requests and responses

```
paraphrase/mock
    ?lang=cs
    &text=Jsem student posledního ročníku gymnázia.

-> {"status": "OK",
    "de": "Poslední rok studuji gymnastiku.",
    "fr": "Jsem v posledním ročníku střední školy.",
    "ru": "Jsem ve čtvrťáku na gymnáziu.",
    "en": "Jsem ve čtvrťáku na gymnáziu."}
```
List of Listings 4.2: An example of Ptakopět backend paraphrase request and response

4.3.2 Data and trained models

Word alignment

Fast align is the only alignment model we deployed. It simply requires a bilingual sentence aligned corpus. We add the incoming language pairs to the data and run Fast align. For the corpora, we make use of Ubuntu localization files.[4] We chose the IT domain, because data for the QE models are only in this domain. The server is distributed with the following language pairs: `cs-de`, `cs-en`, `cs-fr`, `en-de`, `en-es`, `en-et` and `en-fr`.

Quality estimation

The original feature extractor system in QuEst++ supports English→Spanish quality estimation. We experimented with feeding it English→Czech quality estimation data and expected that the ML part would disregard noisy or low information features caused by feeding the feature extractor unsupported language. We found that the performance regressed so considerably that we did not experiment further and focused on other QE systems.

Both DeepQuest (bRNN) and OpenKiwi (Predictor-Estimator) were trained on WMT 2017 English-German Word Level Quality Estimation dataset in the IT domain (Specia and Logacheva, 2017). These trained models are downloaded automatically when running the backend install script. OpenKiwi, in general, proved to be faster, more robust and easier to use than DeepQuest. Because of this, the experiment was conducted with OpenKiwi quality estimation backend.

Czech-German Quality Estimation Dataset

For the experiment, we also needed to train a Czech→German QE model. Since relevant Czech→German training data for QE were not available, we synthesized them from English→German data. We processed the WMT17 English-German data to obtain Czech→German data by translating the source language sentences using LINDAT Translation Popel (2018) from English to Czech. Given triplets (English, German, QE), we thus create triplets of (Czech, German, QE). An example of this can be seen in Figure 4.4.

[4]opus.nlpl.eu/Ubuntu-v14.10.php

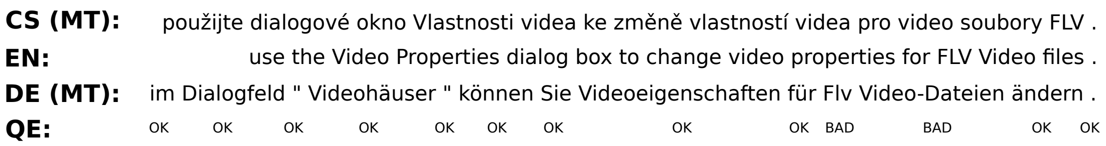

CS (MT): použijte dialogové okno Vlastnosti videa ke změně vlastností videa pro video soubory FLV .
EN: use the Video Properties dialog box to change video properties for FLV Video files .
DE (MT): im Dialogfeld " Videohäuser " können Sie Videoeigenschaften für Flv Video-Dateien ändern .
QE: OK OK OK OK OK OK OK OK OK BAD BAD OK OK

Figure 4.4: Quality estimation tags for tokens and gaps on German sentence translated from English (from WMT19 quality estimation shared task) together with synthetic Czech source (translated from English). MT systems are independent.

To make sure the data did not lose quality, we performed the following experiment: We manually annotated 30 Czech-German and 20 English-German sentences for word-level quality estimation, in the same format as the original English-German dataset, i.e., labeling German words with OK/BAD labels given the source sentence. The original English-German annotation served as the golden standard. Our annotation for English-German was created independently of it and it served as a benchmark for our agreement with the original.

All	
TP=74.57%	FP=2.68%
FN=12.98%	TN=9.76%
Czech→German	
TP=77.58%	FP=3.68%
FN=11.03%	TN=7.71%
English→German	
TP=69.81%	FP=1.11%
FN=16.07%	TN=13.02%

Table 4.1: Confusion matrix for word-level quality estimation annotations of Czech-German and English-German.

Table 4.1 shows the confusion matrices of our annotations compared to the golden standard. The distributions for both language pairs are similar. The sample is very small and the sets of underlying sentences (20 English and 30 Czech) had to be different because the annotation was carried out by a single person, but the results nevertheless indicate that this transfer of QE data by machine-translating the source is viable. The similarity of confusion scores can mean one of the following. Either the German sentence itself was representative enough for the annotator to produce classes with similar distributions, or that both the English and the Czech sentences provided the same level information. In both cases, the pairs (EN, DE) and (CS, DE) seem equally usable, which means that we should be able to train a similarly good quality estimation model based on the synthetic Czech source.

5. Experiment

To test the usability of the Ptakopět tool proposed in this thesis, we designed an experiment. This experiment aimed to classify and describe strategies users take when tasked to do outbound translation, as well as the final quality of the produced texts. It was the main focus of one of the papers connected to this thesis (Zouhar and Bojar, 2020).

5.1 Setup

The experiment was carried out remotely, in two phases. In the first phase, annotators were presented with a sequence of web pages and asked to produce a German sentence given a stimulus at each of them. In the second phase (Section 5.2), a highly-skilled speaker of German validated the outputs of the first phase.

QE highlighting in Ptakopět was enabled only for the first section, because the QE model did not perform well on out-of-domain sentences.

5.1.1 Annotators

There were 8 annotators in total, divided into two groups. The first one was composed of 4 people without advanced knowledge of English[1] and the second one consisted of 4 people with English level of at least C1 on the CEFR scale. All of the annotators had German knowledge of at most A1. We refer to these groups as bilingual and monolingual, respectively. The annotators' knowledge of German and English is summarized in Table 5.1. Annotators with an English level equal to or below B2 were opted out of stimuli from the original SQuAD 2.0.

Annotator	English	German
C1	B2	A0
C2	C1-C2	A0
C3	B2	A0
C4	B2	A0
C5	B1-B2	A0-A1
C6	C2	A0-A1
C7	C1	A0

Table 5.1: Language proficiency of English and German on the CEFR scale

5.1.2 Data

For our experiment, we gathered input data and prompted users to reformulate a specific question or work with the text in some way. Each data section was meant to correspond to some real-life situations.

[1]Note that the annotators never needed to produce any English text in the experiment. Only one subset of the test data needed English comprehension.

Seeking help in technical issues

For the best match with the QE training data (Section 4.3.2), we extracted 35 stimuli (in Czech) from WMT 2017 English-German quality estimation dataset. The sentences describe technical issues when using common office or desktop publishing programs.

The annotators were expected to translate the description of the issue to German relying on machine translation and quality estimation tools. Furthermore, we think that explaining technical issues to IT support in an unknown language is a common outbound translation use case. An example of a technical issue is in Figure 5.1 (translated to English).

Issue description:
The date format cannot be changed from Month-Day-Year to Day-Month-Year.

Figure 5.1: Example description of a technical issue from the experiment dataset.

Common administrative issues

The next 30 test stimuli in the experiments provided a source text in Czech with a piece of factual information (a short span in the text) highlighted. The annotators were supposed to formulate questions that ask for this factual information.

This data was collected from the instructions on how to proceed in various administrative topics at the Municipal District of Prague 6.[2] This use case is inspired by the day to day problems of citizens living in a foreign city. With the help of MT, they can get the gist of regulation or relevant document but they may need to ask the administration for some clarification or a specific detail.

An example of an administrative issue stimulus can be seen in Figure 5.2. For presentation purposes, we again translate the stimulus into English but the annotators saw Czech text and were expected to formulate the question in Czech so that MT produces a good German version.

Paragraph with highlighted span:
Applicant pays 100 CZK when changing a surname that is derogatory, eccentric, ridiculous, garbled, or foreign.

Figure 5.2: Example administrative topic and the factual information to ask for (the price) highlighted

Encyclopedic knowledge: SQuAD 2.0

The last section of the experimental data was based on the Stanford Question Answering Dataset 2.0 Rajpurkar et al. (2018) and its (machine-translated) Czech version.[3] The basic unit of SQuAD are paragraphs with spans. In the context of SQuAD 2.0, this means that there already existed a question for this span. In our

[2]praha6.cz/codelat/index.php

[3]Translation provided by Matúš Žilinec; the dataset is available at zilinec.me/dl/datasets/squad/train-cs-v2.1.json

experiment, we disregard the existing questions and ask our annotators to ask for the highlighted information again. We are thus creating additional questions for the SQuAD dataset, now in Czech.

An example of a paragraph from SQuAD 2.0 and questions we collected from the Ptakopět pilot study (again translated to English) can be seen in Figure 5.3.

Paragraph with highlighted span:
All of Chopin's compositions include the piano. Most are for solo piano, though he also wrote two piano concertos, a few chamber pieces, and some songs to Polish lyrics.
Sample questions asked by our annotators:
What do all Chopin's songs include?
What musical instrument will we hear in virtually all Chopin's compositions?

Figure 5.3: Paragraph from SQuAD with two questions for the underlined span

We were mostly interested in spans of text which had more questions in SQuAD already because such spans seemed easier to create questions for. The distribution of questions per span in SQuAD can be seen in Table 5.2: the vast majority of spans have only one question and having more than four questions per span is very rare. The rightmost column shows how many of such spans were included in our experimental data.

Questions per span	Number of spans in SQuAD 2.0	Occurences in experiment data
1	81619	15
2	2303	15
3	166	15
4	13	10
5	8	5
6	1	0
Total:	84110	60

Table 5.2: SQuAD 2.0 span distribution

In total, 60 paragraphs were chosen from SQuAD 2.0 randomly but respecting the intended distribution in the third column in Table 5.2. These paragraphs were machine-translated to Czech and the spans were transferred to Czech manually. Bilingual users then had half of the SQuAD paragraphs in Czech and half in English, monolingual users saw only the Czech paragraphs. No user saw the same paragraph in both English and Czech.

Annotation task composition

The overall composition of types of stimuli is shown in Table 5.3. The bilingual group received half of the SQuAD stimuli in Czech and half in English. The monolingual group received all the SQuAD stimuli in Czech.

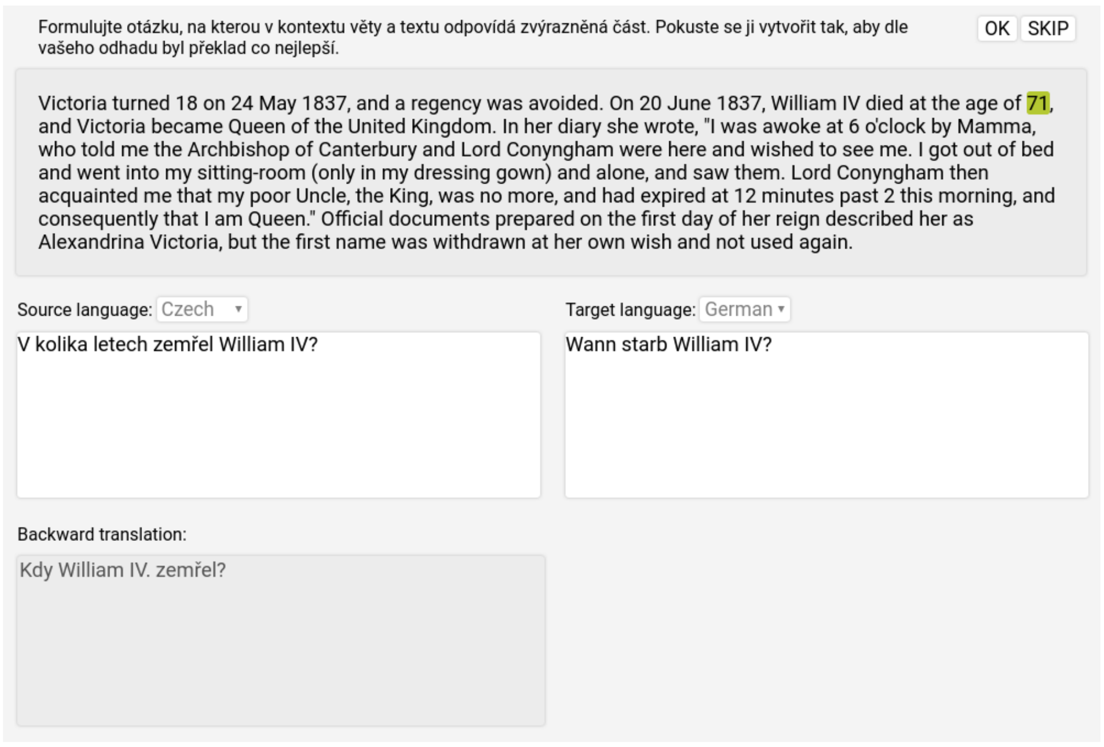
Stimuli	monolingual	bilingual
Technical issues	35	35
Administrative issues	30	30
SQuAD 2.0	0	30
SQuAD 2.0 Czech	60	30
Total	125	125

Table 5.3: Overall composition of the input stimuli

All of the annotators overlap fully in technical and administrative issues. The monolingual annotators overlap entirely within the group and 50% with the bilingual group. Such overlaps are necessary for studying the same stimulus answers variations.

5.1.3 User Interface

The Ptakopět frontend version used for the experiment is stored in the git repository under `v-pilot` tag. It was later improved and the most up to date public version is described thoroughly in Chapter 3. The user interface is different for experiments than for public use. The experiment user interface can be seen in Figure 5.4. Apart from the standard three text boxes, the settings section is hidden and at the top, there is a stimulus briefly describing the task, as well as the text in question.


Formulujte otázku, na kterou v kontextu věty a textu odpovídá zvýrazněná část. Pokuste se ji vytvořit tak, aby dle vašeho odhadu byl překlad co nejlepší.

Victoria turned 18 on 24 May 1837, and a regency was avoided. On 20 June 1837, William IV died at the age of 71, and Victoria became Queen of the United Kingdom. In her diary she wrote, "I was awoke at 6 o'clock by Mamma, who told me the Archbishop of Canterbury and Lord Conyngham were here and wished to see me. I got out of bed and went into my sitting-room (only in my dressing gown) and alone, and saw them. Lord Conyngham then acquainted me that my poor Uncle, the King, was no more, and had expired at 12 minutes past 2 this morning, and consequently that I am Queen." Official documents prepared on the first day of her reign described her as Alexandrina Victoria, but the first name was withdrawn at her own wish and not used again.

Source language: Czech

V kolika letech zemřel William IV?

Target language: German

Wann starb William IV?

Backward translation:

Kdy William IV. zemřel?

Figure 5.4: Screenshot of Ptakopět for annotators with a stimulus from the translated version of SQuAD 2.0

5.1.4 Technical details

All the relevant experiment files and scripts together with their usage are discussed in detail in Appendix B.4.

Data gathering

Throughout the Ptakopět annotation, we logged various data, while the users interacted with Ptakopět. The list of each logged information is shown in Table 5.4 and the description of each information type is in Table 5.5. Additionally, each logged action contained Unix timestamp. The logs are stored for in files with `<userID>.log` scheme.

The logged data for each user is stored in CSV format in one file, thus interleaving multiple tables. Each row is prefixed with an extra column, describing the logged action. Thus the file can be easily grepped to extract correctly formatted tables.

Action code	Logged information	Description
START	-	The user logs in
NEXT	SID	A stimuli is shown
CONFIRM	SID, TEXT1, TEXT2	User accepts solution
SKIP	SID, REASON	User skips stimuli
TRANSLATE1	SID, TEXT1, TEXT2	Forward translation is displayed
TRANSLATE2	SID, TEXT2, TEXT3	Backward translation is displayed
ESTIMATE	SID, ESTIMATION	Quality estimation is highlighted
ALIGN	SID, ALIGNMENT	Source complexity is highlighted
PARAPHRASE	PARAPHRASES	Paraphrases are displayed
NOTE	SID, MESSAGE	User typed in a note

Table 5.4: Logged information from Ptakopět users for each of their actions. Last two logging actions were not used for the pilot experiment.

Logged information	Description
SID	Identifier of the relevant stimuli
TEXT1	Content of the source text area
TEXT2	Content of the target text area
TEXT3	Content of the backward translation text area
ESTIMATION	Quality estimation data
ALIGNMENT	Source to target word alignment
REASON	User's motive for skipping answering the stimuli

Table 5.5: Description of logged information from Ptakopět users

Baked queue stimuli preparation

We want the stimuli to be distributed randomly but in a fixed manner. We also want to know the distribution of stimuli to users beforehand so that we can regenerate it in case of any anomalies. For this, we use the concept of *baked queues*. For every user, we generate a fixed array of stimuli that will be shown to them.

The pool size of stimuli per domain is hardcoded as Technical issues: 35, Administrative issues: 30, SQuAD Czech: 60, SQuAD: 60.

5.2 Results

The results of this experiment were published in Zouhar and Bojar (2020) and this section uses whole paragraphs and tables from this paper.

5.2.1 Basic statistics

We refer to sequences of log entries related to the same stimulus as segments. The number of finished segments, as well as their average duration in every domain, is shown in Table 5.6. Since the differences in duration between each segment were not high (min 90s, max 106s), we concluded that the users employed similar strategies across all domains and that no domain was exceptionally difficult nor easier than the others.

Domain	Segments	Average duration
SQuAD 2.0	141	100s
SQuAD 2.0 Czech	346	94s
Technical issues	268	107s
Administrative issues	246	90s
All	1001	98s

Table 5.6: Number of segments and average duration across all users per domain in collected data

5.2.2 Types of edits

Some of the stimuli were skipped, mostly because the annotators did not have enough confidence in the MT system's performance (for a given stimulus) and were unable to produce a better result. We describe such segments as *skipped* as opposed to *finished*. From the finished ones, about a quarter of the segments were written linearly (no edits or deletions in already written text). Such segments are denoted as *linear* as opposed to segments, which had some edits in already written parts (*with edits*). The number of skipped, finished, linear and edited segments can be seen in Table 5.7.

We see that the proportion of skipped segments (i.e., segments where the annotator failed to produce an output they could accept) is not excessively high. The easiest to process were administrative issues (5.7 % skipped segments) and the hardest was the technical issues (10.8 %). SQuAD reached 7.8 % (English) and 7.5 % (Czech) of skipped segments.

Of the finished segments, most (72%) were edited and not just linearly written (28%). Additionally, in technical issues, the stimulus was the description of the technical problem itself, so the annotators could choose to simply copy this text and paste it in the input window. The number of occurrences of this behavior is described in the table as *init copy* (60% of all edited). We also measured the number of final inputs, which matched the initial stimulus (*Copy & submit*, 6% of all edited).

Domain	Description	Segments	Ratio
SQuAD 2.0	Skipped	11 (8%)	(of all)
	Finished	130 (92%)	
	Linear	52 (40%)	(of fin.)
	With edits	78 (60%)	
SQuAD 2.0 Czech	Skipped	26 (8%)	(of all)
	Finished	320 (92%)	
	Linear	110 (34%)	(of fin.)
	With edits	210 (66%)	
Tech issues	Skipped	29 (11%)	(of all)
	Finished	239 (89%)	
	Linear	27 (11%)	(of fin.)
	With edits	212 (89%)	
	Init copy	127 (60%)	(of edt.)
	Copy & submit	13 (6%)	
Administrative issues	Skipped	14 (6%)	(of all)
	Finished	232 (94%)	
	Linear	70 (30%)	(of fin.)
	With edits	162 (70%)	
All	Skipped	80 (8%)	(of all)
	Finished	921 (92%)	
	Linear	259 (28%)	(of fin.)
	With edits	662 (72%)	

Table 5.7: Number of skipped, finished, linear and edited segments per domain in collected data together with percentage of all, finished or edited segments.

We then focused on the segments, which were further edited. We tried to extract the first input the annotator expected to be successful. We call this input *first viable* and choose it heuristically as the longest nonfinal input ending with a punctuation mark. We then compute the similarity between the first viable source/translation and the final source/translation version as confirmed by the annotator using Gestalt Pattern Matching on word-level (implemented in Python's difflib). This similarity is shown per domain in Table 5.8.

Domain	Source sim.	Translation sim.
SQuAD 2.0	69%	55%
SQuAD 2.0 Czech	75%	60%
Tech issues	78%	67%
Administrative issues	74%	57%
All	75%	61%

Table 5.8: Similarity between first viable and final versions of inputs (source texts) and outputs (translations) (only on segments with edits)

From Table 5.8, we can see that even though the first viable and final inputs are quite similar (75% on average across all domains), the first viable and final translations are less similar (61% on average). This indicates that the edits had a considerable effect on the translation.

5.2.3 Evaluation survey

At the end of the experiment, we asked the annotators to fill in a short survey. The results are shown in Table 5.9.

Question	Domain	Average
What confidence do you have in the translations you have created?	SQuAD 2.0 (both)	1.14
	Technical issues	2.86
	Administrative issues	2.29
	All	2.10
How useful was the highlighting of problematic words in technical issues?		2.29
How useful was the environment for these tasks, compared to other web interfaces (Google Translate, Bing Translator and others)?		1.71

Table 5.9: Annotator survey results (1 - most, 5 - least)

We suspect that the overall results are affected by the relatively low quality of the MT system. Most of the annotators complained of this, stating that the MT system made obvious mistakes, such as adding random words. Should we deploy a better MT system, the average scores would probably go up. At the same time, it seems that we have chosen the right level of MT quality for the experiment: MT was not too good (edits were needed) and not too bad (at most 10.8 % of segments were given up).

The perceived confidence per domain confirms that technical issues were the hardest (probably because of vocabulary deficiency of the MT system in the IT domain) and it was the highest for encyclopedic questions.

The good news is that the overall usefulness of Ptakopět compared to standard web interfaces to MT was rated as 1.71 on the 1–5 scale, although the perceived utility of QE was lower (2.29).

In a questionnaire we also inquired about the users' strategies. Most of them focused on the backtranslation to validate the output. If they suspected that the result might not be preferable (either by the backtranslation or by looking at the result itself), they tried reformulating the input by using synonyms. If that did not help, they tried simplifying the sentence, even beyond the threshold of a grammatically sound output sentence, attempting just to communicate the meaning properly.

It is worth noting that the backward translation can in principle fix previously introduced errors, thus hiding the problem. In these cases, the users could get a false sense of confidence in the translation. For such occasions, an external tool (e.g., MT quality estimator) is needed.

5.2.4 Output validation

After we collected data from the previous annotation phase, we extracted final translations and translations of the first viable inputs for each segment (if possible). We then asked another annotator with a good command of German (C2 on the CEFR scale) to rate each translation on the scale of 1 to 5 (best to worst), estimating to what extent a native German would understand the message.

Validation results

	First viable		Final	
Domain	Avg.	Var.	Avg.	Var.
SQuAD 2.0	3.43	2.56	1.91	2.00
SQuAD 2.0 Czech	3.95	2.18	2.64	2.67
Tech issues	3.77	1.79	3.10	2.23
Administrative issues	4.05	1.91	2.92	2.55
All	3.85	2.07	2.77	2.55

Table 5.10: Average quality ratings across domains for first viable and final translations (1 - best, 5 - worst)

The results for each domain for the final and first viable translations are in Table 5.10. In each domain, the final translations were much better than the translations for the first viable inputs. The average score improves from 3.85±1.44 to 2.77±1.6. The paired t-test showed that the difference is highly statistically significant ($p < 0.0001$ for 0.75 difference between final and first viable ratings).

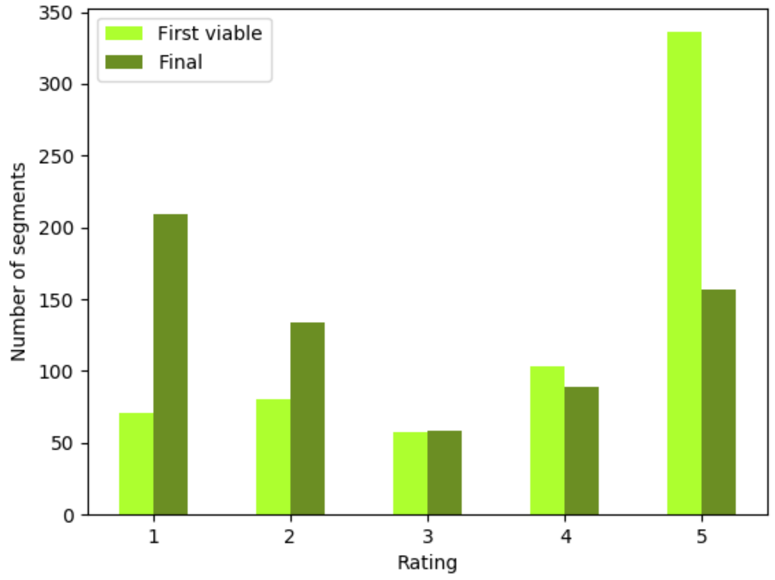


Figure 5.5: Histogram of ratings for first viable and final translations (1 - best, 5 - worst)

The validation scores assigned to the individual segments using the histogram is presented in Figure 5.5. We see that the first viable translations received mostly the worst rating while final hypotheses are bimodal: the majority received a favorable validation score but a considerable portion (24%) had the worst score. We assume that in these cases, our setup was unreliable and fooled the user in accepting a misleading translation. A possible explanation of this is offered in Section 3.3.

Overall, this is a clear success, as our technique helps people to produce better messages in a language they do not speak. Nevertheless, it is important to mention the limitations of our pilot study. Our heuristics for picking first viable inputs may include sentences, which were actually not thought to be viable by the user. Maybe the sentences contained obvious errors, such as typos, which the user would fix anyway but maybe the user would not notice if we did not present the backtranslation. A more thorough exploration is needed to isolate such effects.

Relation to sentence length

One could expect that shorter sentences are generally easier to process by MT (except for very ambiguous very short sentences). To analyze this assumption in our setting, we plot the average validation score assigned to sentences based on the source length.

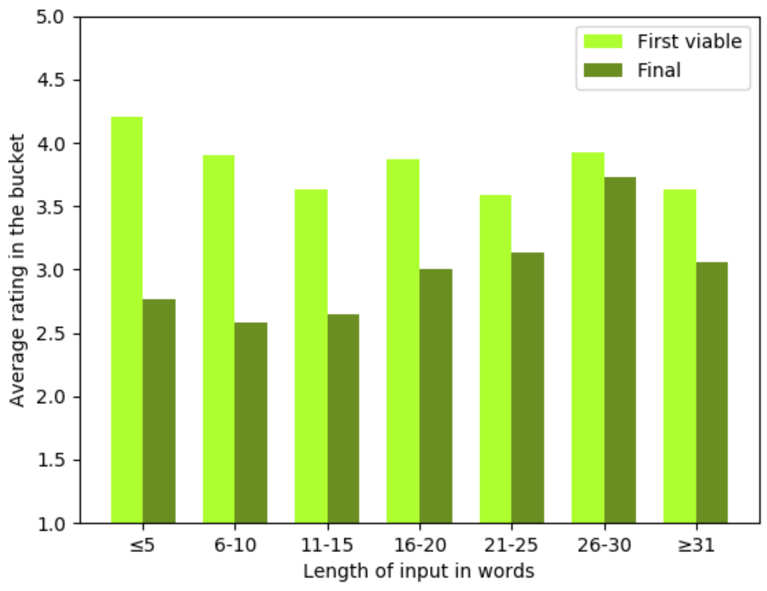


Figure 5.6: Average rating for first viable and final translations based on the translated sentence length (1 - best, 5 - worst)

Figure 5.6 indicates that the assumed effect is not apparent in our case, at least not with our estimation of the first viable hypotheses. The shorter sentences generally receive worse validation scores than longer ones, but the differences are not very big.

For final hypotheses, the assumption seems more accurate: The best validation score was assigned to sentences of 6–10 words and the worst to sentences over 25 words. A noteworthy observation is that for these long sentences, the improvement in the validation scores from first viable to the final hypothesis is very low.

5.2.5 Conclusion

In this pilot experiment, users who did not know German were tasked to use this system for real-world use cases (communication with IT support, describing administrative issues and asking encyclopedic questions).

Across these domains, 5–10 % of inputs could not have been translated (our annotators have given up). For the submitted translations, the average self-reported confidence in the translations was 2.1 on a 1–5 (best–worst) scale and the tool was found more useful than standard web interfaces to MT (average usefulness of 1.71, same scale).

The majority of inputs were edited and while initial inputs and the final inputs were quite similar in the source language (word-level Gestalt Pattern Matching similarity of 75 %), the translations of them differed more (average similarity of 61 %).

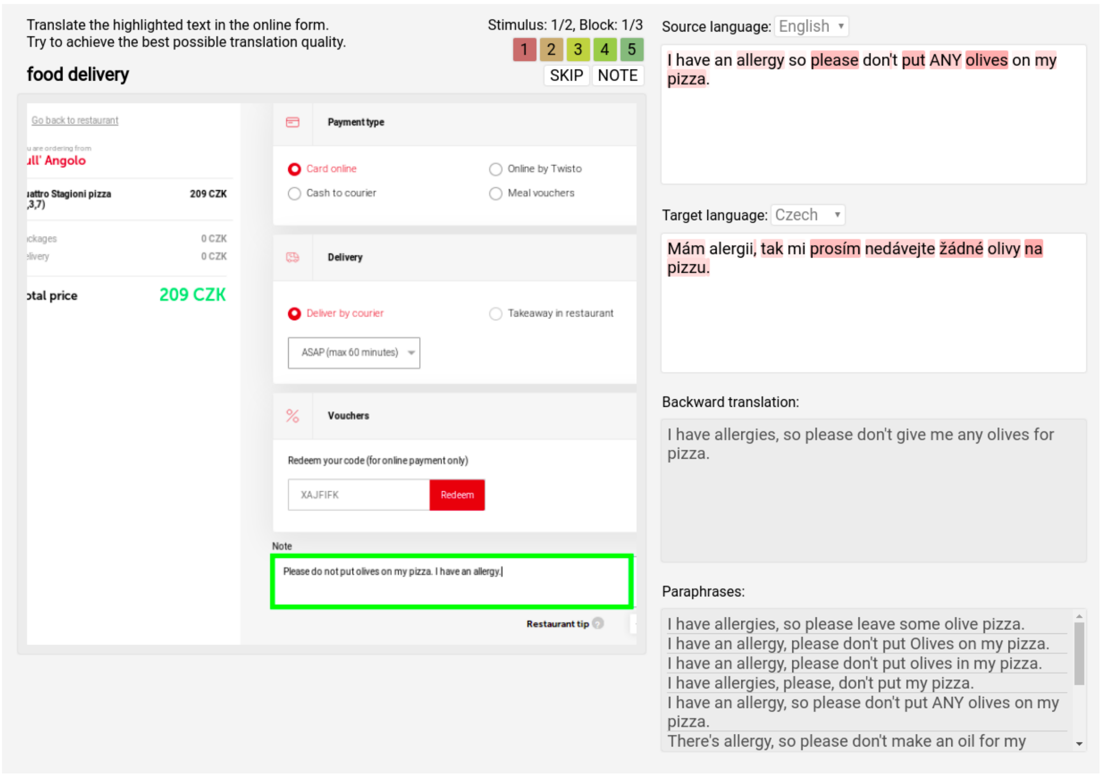
The second, validation, phase of our experiment confirmed that the overall understandability of the translations improved from 3.9 to 2.71 on the 1–5 (best–worst) scale.

5.3 Changes in Ptakopět

During the experiment, we acknowledged the need for separating experiment definitions from the rest of the code. As a result, the whole experiment content can be specified by a single JSON file instead of having to be hardcoded in the project. This is described in detail Section 4.2. We also restructured the whole codebase, so that the experiment and settings profiles are more separated. The original version commit on which the pilot experiment took place got the `v-pilot` tag in the `zouharvi/ptakopet` repository.

In preparation for the next experiment we added a paraphrasing module visible in the bottom right corner in Figure 5.7

The experiment user interface also changed to accommodate better more modules and stimulus in the form of images. A stimuli is now confirmed by clicking one of the five buttons. They correspond to the confidence the user itself has of their produced texts (1 lowest – 5 most). We added the ability to leave a note anywhere in the experiment by clicking the NOTE button below the rating scale. The progress is tracked visibly on the experiment page above the scale. This way, the annotators have a better overview of the amount of work left. All of the new features are shown in Figure 5.7.



Translate the highlighted text in the online form.
Try to achieve the best possible translation quality.

food delivery

Stimulus: 1/2, Block: 1/3

1 2 3 4 5

SKIP NOTE

Go back to restaurant

u are ordering from
ıll' Angolo

ıattro Stagioni pizza ,3,7) 209 CZK

ıckages 0 CZK
elivery 0 CZK

otal price 209 CZK

Payment type

[✓ Card online] Online by Twisto
Cash to courier Meal vouchers

Delivery

[✓ Deliver by courier] Takeaway in restaurant

ASAP (max 60 minutes)

Vouchers

Redeem your code (for online payment only)

XAJFIFK Redeem

Note

Please do not put olives on my pizza. I have an allergy.

Restaurant tip

Source language: English

I have an allergy so please don't put ANY olives on my pizza.

Target language: Czech

Mám alergii, tak mi prosím nedávejte žádné olivy na pizzu.

Backward translation:

I have allergies, so please don't give me any olives for pizza.

Paraphrases:

I have allergies, so please leave some olive pizza.
I have an allergy, please don't put Olives on my pizza.
I have an allergy, please don't put olives in my pizza.
I have allergies, please, don't put my pizza.
I have an allergy, so please don't put ANY olives on my pizza.
There's allergy, so please don't make an oil for my

Figure 5.7: Screenshot of the update Ptakopět interface for annotators with an image stimuli

The baked stimuli queue is now split into blocks. They were added only for management purposes so that it is easier for users to split their work into several phases. They are notified by an alert box that they completed a block.

From the server point of view, nothing has been changed. Only the logs are now stored with `<userID>-<block>.log` scheme.

Conclusion

In this thesis we described the issue of outbound translation as a complement to gisting. We briefly described technologies related to machine translation quality estimation and presented a new system Ptakopět for both real usage and for experiments in this area.

We also conducted an experiment on human annotators, which proved that cues such as backward translation or quality estimation can increase the user's confidence in the produced translation, but also improve the final translation itself.

We found that enhancing MT with QE improves the user experience. We expect some form of quality estimation to start appearing more in publicly available MT solutions.

Future work

In future experiments we would like to quantitatively measure which cues are most relevant for outbound translation and how they project on user confidence in the translation. This can be done by providing different cues to different users on the same task and seeing how it affects their performance and trust in the MT system.

From the described experiment we already know that not all errors get recovered in the backward translation. This is a proof that backward translation is useful for the task of outbound translation, at least to some extent. We wish to explore this issue of backtranslation errors in general and see for example how many errors and of what kind get recovered.

Lastly we would like to explore how to gather more QE data, because at this time only a very small dataset of manually annotated QE data for several language pairs is publicly available by WMT. This QE data synthesis is a work in progress.[4]

[4]github.com/zouharvi/MosQEto

Bibliography

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

Ondrej Bojar and Magdalena Prokopová. Czech-english word alignment. In *LREC*, pages 1236–1239, 2006.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

Leshem Choshen and Omri Abend. Automatically Extracting Challenge Sets for Non local Phenomena in Neural Machine Translation. *arXiv e-prints*, art. arXiv:1909.06814, Sep 2019.

Jan Cuřín. Statistical methods in czech-english machine translation. 2006.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

Ondřej Dušek, Jan Hajič, Jaroslava Hlaváčová, Michal Novák, Pavel Pecina, Rudolf Rosa, Aleš Tamchyna, Zdeňka Urešová, and Daniel Zeman. Machine translation of medical texts in the khresmoi project. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 221–228, 2014.

Chris Dyer, Victor Chahuneau, and Noah A Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL-HLT*, pages 644–648, 2013.

Erick Fonseca, Lisa Yankovskaya, André F. T. Martins, Mark Fishel, and Christian Federmann. Findings of the WMT 2019 shared tasks on quality estimation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 1–10, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5401. URL `https://www.aclweb.org/anthology/W19-5401`.

William A Gale and Kenneth W Church. A program for aligning sentences in bilingual corpora. *Computational linguistics*, 19(1):75–102, 1993.

Jan Hajič. Ruslan - an mt sysstem between closely related languages. In *Third Conference of the European Chapter of the Association for Computational Linguistics*, 1987.

Julia Ive, Frédéric Blain, and Lucia Specia. Deepquest: a framework for neural-based quality estimation. *In the Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics, Sante Fe, New Mexico, USA*, 2018.

Fábio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics–System Demonstrations*, pages 117–122, Florence, Italy, July 2019. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P19-3020`.

Hyun Kim, Hun-Young Jung, Hongseok Kwon, Jong-Hyeok Lee, and Seung-Hoon Na. Predictor-estimator: Neural quality estimation based on target word prediction for machine translation. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 17(1):3:1–3:22, September 2017. ISSN 2375-4699. doi: 10.1145/3109480. URL `http://doi.acm.org/10.1145/3109480`.

Zdeněk Kirschner and Alexandr Rosen. Apac—an experiment in machine translation. *Machine translation*, 4(3):177–193, 1989.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 316–322, 2015.

Veronica Lawson. *The METEO System.* ASLIB, 1982.

Joël Legrand, Michael Auli, and Ronan Collobert. Neural network-based word alignment through score aggregation. *arXiv preprint arXiv:1606.09560*, 2016.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. Results of the wmt19 metrics shared task: Segment-level and strong mt systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 62–90, 2019.

André F. T. Martins, Ramón Astudillo, Chris Hokamp, and Fabio Kepler. Unbabel's participation in the WMT16 word-level translation quality estimation shared task. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 806–811, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2387. URL `https://www.aclweb.org/anthology/W16-2387`.

André F.T. Martins, Marcin Junczys-Dowmunt, Fabio N. Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. Pushing the limits of translation quality estimation. *Transactions of* *the Association for Computational Linguistics*, 5:205–218, May 2017. URL https://www.microsoft.com/en-us/research/publication/pushing-limits-translation-quality-estimation/.

Rada Mihalcea and Ted Pedersen. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond*, pages 1–10, 2003.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Jan Niehues and Ngoc-Quan Pham. Modeling confidence in sequence-to-sequence models, 2019.

Naoaki Okazaki. Crfsuite: a fast implementation of conditional random fields (crfs), 2007. URL http://www.chokkan.org/software/crfsuite/.

Robert Östling and Jörg Tiedemann. Efficient word alignment with Markov Chain Monte Carlo. *Prague Bulletin of Mathematical Linguistics*, 106:125–146, October 2016. URL http://ufal.mff.cuni.cz/pbml/106/art-ostling-tiedemann.pdf.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Martin Popel. CUNI transformer neural MT system for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 482–487, Belgium, Brussels, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6424. URL https://www.aclweb.org/anthology/W18-6424.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL http://arxiv.org/abs/1806.03822.

Carolina Scarton and Lucia Specia. A reading comprehension corpus for machine translation evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3652–3658, 2016.

Rico Sennrich and Martin Volk. Iterative, mt-based sentence alignment of parallel texts. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, pages 175–182, 2011.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

Harold Somers. Round-trip translation: What is it good for? In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 127–133, 2005.

Lucia Specia and Varvara Logacheva. WMT17 quality estimation shared task training and development data, 2017. URL `http://hdl.handle.net/11372/LRT-1974`. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Lucia Specia, Gustavo Paetzold, and Carolina Scarton. Multi-level translation quality prediction with quest++. In *ACL-IJCNLP 2015 System Demonstrations*, pages 115–120, Beijing, China, 2015. URL `http://www.aclweb.org/anthology/P15-4020`.

Jörg Tiedemann. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Peter Toma. Systran as a multilingual machine translation system. In *Overcoming the language barrier*, 1977.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Qian Yu, Aurélien Max, and François Yvon. Revisiting sentence alignment algorithms for alignment visualization and evaluation. In *The 5th Workshop on Building and Using Comparable Corpora*, page 10, 2012.

Vilém Zouhar and Ondřej Bojar. Outbound translation user interface Ptakopět: A pilot study. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6969–6977, Marseille, France, May 2020. European Language Resources Association. URL `https://www.aclweb.org/anthology/2020.lrec-1.860`.

Pierre Zweigenbaum, Serge Sharoff, and Reinhard Rapp. Overview of the second bucc shared task: Spotting parallel sentences in comparable corpora. In *Proceedings of the 10th Workshop on Building and Using Comparable Corpora*, pages 60–67, 2017.

List of Figures

List of Tables

List of Listings

List of Abbreviations

NLP Natural Language Processing
RBMT, SMT, NMT Rule Based Machine Translation, Statistical Machine Translation, Neural Machine Translation
QE Quality Estimation
RTT Round Trip Translation
WMT Workshop on Statistical Machine Translation[5]
BUCC Workshop on Building and Using Comparable Corpora[6]
SVM Support-vector Machine, a machine learning model[7]
W3C World Wide Web Consortium
OPUS The Open Parallel Corpus[8]
HTER Human-targeted Translation Error Rate - the minimum number of edits, so that the result is similar enough in meaning to the reference text
(bi)RNN (bidirectional) Recurent Neural Network
GRU, LSTM Gated Reccurent Units and Long Short-Term Memory are building blocks of RNN

[5]statmt.org
[6]comparable.limsi.fr
[7]en.wikipedia.org/wiki/Support-vector_machine
[8]opus.nlpl.eu/

A. Attachments

A.1 Snapshot of Ptakopět frontend repository

Besides the source code for the web application, the frontend repository contains also tools for the experiment and documentation source. The snapshot was created at `15dab17f`. It was mirrored from github.com/zouharvi/ptakopet. The main repository directory structure is discussed in the technical documentation in Appendix B.

A.2 Snapshot of Ptakopět backend repository

The backend repository, mirrored from github.com/zouharvi/ptakopet-server, contains server code at `f099a35b`. The repository directory structure is discussed in the technical documentation in Appendix B.

A.3 Snapshot of SlowAlign repository

We used SlowAlign vilda.net/s/slowalign to produce alignment figures. It is a snapshot of github.com/zouharvi/SlowAlign at `0a158baf`.

A.4 Trained models

The attachment also contains two trained DeepQuest models, Czech→German and English→German and one OpenKiwi model, Czech→German. They are, however, downloaded automatically when running the backend install script.

B. Development documentation

In this appendix we aim to describe all technical details regarding the deploymend and implementation of Ptakopět. It is a complement to Chapter 4 which deals with the implementation.

B.1 Frontend

The `zouharvi/ptakopet` repository contains the following directory structure:

- `dist/` - output directory, which contains all of the build files.
- `docs/` - source code for the Ptakopět documentation; the content is located in `docs/src/` and is written in simple markdown[1]
- `meta/` - files related to the whole project and not just the frontend
  - `meta/logo` - the Ptakopět bird logo and relevant files
  - `meta/object_design` - diagrams, which describe the project workings. They are used also in the documentation
  - `meta/study_pilot` - all of the files connected with the pilot experiment. The contents are further described in Appendix B.4.2
  - `meta/synth_qe_test` - data and evaluation script for the experiment, which tested the quality of transferred QE data (described in Section 4.3.2)
- `node_modules/` - dependencies downloaded automatically by npm[2]
- `package.json` - the build commands and list of dependencies
- `package-lock.json` - list of all dependencies together with their versions in the project.
- `README.md` - introductory text about Ptakopět
- `src/` - source files; their structure is described in detail in Section 4.1
- `tsconfig.json` - TypeScript compiler settings.
- `webpack.config.js` - Webpack settings

B.1.1 Build and compilation

The code in Listing B.1 clones and builds the whole frontend in production mode:

```
git clone https://github.com/zouharvi/ptakopet
cd ptakopet
git submodule update --init
npm install
npm run build
```

List of Listings B.1: Shell code for building the whole frontend

All of the files necessary for running the frontend should then be present in the `dist/` directory. The web page does not need to be served from the Internet and can be run locally.[3] To do so, it is sufficient to load `dist/index.html` in any

[1]Documentation is generated dynamically from the markdown by docsify.js.org.

[2]www.npmjs.com

[3]It still needs a server to respond to various MT related requests.

modern browser.

Moreover, if one wishes to rebuild the code upon every change in the source files, they may run `npm run dev`. This starts a small HTTP server and makes the application accessible from localhost. Upon code changes, the code gets recompiled in development mode.

B.1.2 Architecture

The source files directory has the following structure:

- `src/messages/` - code for anything related to backend requests (machine translation, quality estimation, alignment and paraphrasing)
- `src/misc/` - miscellaneous files (code utilities, current page config and profiles)
- `src/page/` - files relevant to DOM manipulation
- `src/study/` - code for displaying and collecting experiment data
- `src/main.ts` - entry point for the frontend application

Backends

As mentioned in Section 4.1, a backend in this context is just an object, which contains a list of supported language pairs, a name and a function, which for some input returns a promise of the relevant output. For example in case of `Estimator` one may define a **Random** backend as in Listing B.2.

```
{
composeRequest(
    [lang1, lang2]: [LanguageCode, LanguageCode],
    [text1, text2]: [string, string],
    extra: ExtraTranslationInfo
): Promise<Estimation> {
    return new Promise<Estimation>(async (resolve, reject)
    => {
        let tokens = await tokenizer.tokenize(
            targetText, Settings.language2 as LanguageCode)
        let estimation: Estimation = []
        for (let i in tokens)
            estimation.push(Math.random())
        resolve(estimation)
    })
},

languages:
    Utils.generatePairsSet<LanguageCode>(Utils.Languages),

name:
    'Random'
}
```

List of Listings B.2: Random quality estimation backend definition

Highlighting

As part of the quality estimation and source complexity presentation, we wanted to highlight suspicious parts of the target and source texts, respectively. Even though this seems like a trivial and prevalent task in web development, we found a lack of robust and functional solutions. Our goal was to extend the functionality of the `textarea` element so that we could display the highlighting, but keeping the user experience unaffected.

The first option we explored was based on editable `div` elements. A `div` can be made editable using the attribute `contenteditable="true"`. The `div` would then be filled by tokens, each in its `span`. The tokens could then be styled, for example with an attribute like `style='background-color:  red'`. Even though this would potentially offer extensive functionality, we found it very difficult to work with. One of the biggest issues was that element focus and cursor position was lost on rerender.

The other approach, which we eventually chose, was also used in Ptakopět-old-2. It is based on a plugin[4] by Will Boyd named highlight-within-textarea. It is a simple jQuery plugin, which uses a placeholder div (overlapping with the textarea), which is redrawn according to the current highlight values. This way, the active textarea element does not lose focus nor cursor position because it is not being manipulated. It is also beneficial that it works on mobile browsers as well.

Unfortunately, we could not use this plugin out of the box, as it was created to highlight keywords in the text. We thus forked[5] this project and added the desired functionality. This project is incorporated in the main Ptakopět repository as a git submodule.

During the pilot experiment, we found out that there is a serious memory leak in this plugin. This did not manifest in the original plugin, as the set of keywords to highlight needed to be defined only once. For our use case, we needed to display different highlight values roughly every second. After bisecting the bug, we committed the fix into the forked repository.

[4]github.com/lonekorean/highlight-within-textarea
[5]github.com/zouharvi/highlight-within-textarea

B.2 Experiment definition

In this section we describe the experiment definition on the frontend side. The tools used for generating queues and stimuli and log evaluation are discussed in Appendix B.4.1 and Appendix B.4.2.

An experiment can be defined in a single JSON file. It is included compile-time in the `src/study/baked_study.ts` file. The `src/study/data/` directory contains two files:

- `study_pilot.json` - definition for the experiment described in this thesis
- `study_edin.json` - definition for an upcoming experiment which is part of this thesis

The JSON file has to contain a top-level object containing the keys `users`, `stimuliID` and `stimuliRules`. We explain every item type in a separate paragraph and provide a TypeScript type definition for the whole document in Listing B.3.

```
interface BakedStudyType {
    users: {
        [id: string]: Array<Array<string>>
    },
    stimuli: {
        [id: string]: string
    },
    stimuliRules: Array<{
        rule: string,
        message?: string,
        profile?: SettingsProfile,
    }>,
}

interface SettingsProfile {
    settings?: SettingsObject,
    qe?: boolean, // quality estimation
    mt?: boolean, // machine translation (translate 1)
    bt?: boolean, // backward translation (translate 2)
    pp?: boolean, // paraphrasing
    manual?: boolean, // allow further manual settings
}
```

List of Listings B.3: Experiment definition type

`users`

`users` is an object with `userID` keys. These keys map to an array of arrays of `stimuliIDExtendeds`. This defines the experiment blocks (every innermost array is a separate block) and corresponding stimuli references. The `users` object can look like in Listing B.4.

```
"users": {
    "alice": [
        [ "003#bt.n", "007#bt.n", "012#bt.y", ... ],
        [ "020#bt.y", "011#bt.y", "050#bt.y", ... ]
    ],
    "bob": [
        [ "101#bt.n", "003#bt.y", "084#bt.n", ... ],
        [ "054#bt.n", "059#bt.y", "102#bt.n", ... ]
    ],
},
```

List of Listings B.4: `users` experiment definition example

`stimuli`

`stimuli` is an object with `stimuliID` keys mapping to a HTML string, which gets pasted into the page when the given stimuli is to be shown. It is important to note, that it can by any HTML, like images, not just text. The example in Listing B.5 contains excerpts from stimuli of the experiment. The strings are paragraphs with `<mark>` tags which highlight parts of the shown text.

```
"stimuli": {
    "001": "<p>A new style of wafers composed of gallium-
        nitride-on-silicon (GaN-on-Si) is being used to
        produce white LEDs using <mark>200-mm</mark>
        silicon wafers. ... </p>",
    "002": "<p>Average annual precipitation is <mark>15
        inches</mark> (380 mm), but great variations
        are seen. ... </p>",
    ...
}
```

List of Listings B.5: `stimuli` experiment definition example

`stimuliRules`

`stimuliRules` is an array of `stimuliRule` objects, which contain the regex and a `settings` object. If the regex matches the current `stimulIDExtended`, the settings are applied. Multiple settings objects can be applied and can even override each other. The last rule in the array gets applied last. This allows the `stimuliIDExtended` (in the `users` section) to contain different information about e.g., whether the paraphraser should be shown to a given user, while still sharing the same `stimuliID`.

An example of a `stimuliRules` array is shown in Listing B.6.

`stimuliIDExtended`

`stimuliIDExtended` is composed of two parts separated by `#`. The first part is the actual `stimuliID` by which a lookup to this object is done, while the rest is used for any symbols, that can be later recognized by `stimuliRules`.

The string can then look like: `"003#bt.y"` and `"003#bt.n"`. In both of these examples, the same stimulus would be shown according to the lookup in `stimuli`, but `stimuliRules` could have two rules: `"\d+\#bt\.y"` and `"\d+\#bt\.n"` which would turn the backtranslation on and off respectively.

This example is also demonstrated in Listing B.6. In this example a `stimuliRule` with the regex `".*"` is used. This will match anything, so other rules will only partially override this rule in case they also match. When combined with the `users` definition in Listing B.4, the user `alice` would be shown the stimuli `003` without backward translation and `bob` would see the same stimuli but with backward translation enabled.

`stimuliRule`

Every `stimuliRule` object has to contain a `rule` key with the relevant regex and optionally a `settings` object or a `message` string. The later is shown above the stimuli that matches the rule. It can also contain HTML markup.

`settingsProfile` object

`settings` objects have a type defined in Listing B.3. It contains variables, which define whether a given module should be active or not and also the backends. The manual variable defines, whether users are allowed to change the backends themselves. It defaults to `false`.

```
"stimuliRules": [
    {
        "rule": ".*",
        "profile": {
            "settings": {
                "backendTranslator": "ufalTransformer",
                "backendEstimator": "random",
                "backendAligner": "fast_align_ubuntu",
                "language1": "en",
                "language2": "cs"
            },
            "qe": true,
            "mt": true,
            "bt": true,
            "pp": false,
            "manual": false
        },
        "message": "Translate the highlighted text in
            the provided text."
    },
    {
        "rule": ".*#.*bt\\.n.*",
        "profile": {
            "bt": false
        }
    },
    {
        "rule": ".*#.*bt\\.y.*",
        "profile": {
            "bt": true
        }
    },
    ...
]
```

List of Listings B.6: `stimuliRules` experiment definition example

B.3 Server backend

The zouharvi/ptakopet-server repository contains several folders and files:

- align/ - models and config files for bitext alignment
- data/ - data which the models make use of
- install.sh - the root install script
- qe/ - models and config files for quality estimation
- README.md - an introductory text about the server
- server/src/ - the server source files
- server/run.sh and server/run_nohup.sh - used for launching

B.3.1 Installation

The server needs to be run on a modern UNIX machine. It was tested on Ubuntu 19.10 and Fedora 31. The installation is done via a shell script install.sh. It first checks whether all requirements are met. The requirements are on packages installed via pip2 and pip3 (Theano, mosestokenizer, etc.), but also on system tools (CMake, g++, tar, nohup). If all requirements are satisfied, it proceeds to launch the installation scripts for alignment, quality estimation and the server itself. According to modern development standards, the server should be dockerized. On the other hand, since only usually one instance would be running at a time, dockerizing it would be overengineering.

To run the server (on 0.0.0.0:80), launch the server/run.sh script. A common practice is to connect to a remote machine via SSH and launch the server. For that, there is a script server/run_nohup.sh, which disregards kill signals on user logout.

B.4 Experiment

All experiment data (both preparation and collected data) are stored in `ptakopet/meta/study_pilot`. The directory is structured as:

- `logs/`
  - `logs/raw` anonymized collected raw log files
  - `logs/qe_annotation` quality annotation of the collected data
  - `logs/stable` contains the finished product of all processing scripts
- `prepare_questions/` scripts used to genereate stimuli and also the finished baked queues
- `processing_scripts/` contains all log processing scripts
- `README.md` introductory text about the experiment

B.4.1 Stimuli preparation

Baked queue

The Python program `prepare_questions/bake.py` contains exactly this logic. Given a number of monolingual (`--cusers`) and bilingual (`--busers`) users, it will try to distribute the stimuli from the available pool as uniformly as possible. By default the output is stored in a json file `baked.json`, but this can be changed using the `--file` argument. The seed and maximum number of stimuli per user can be specified using `--seed` and `--per_user` respectively.

The program also prints the domain sizes for every user (important only for bilingual users), the intersection between users' baked queues and a histogram. All three sections are displayed in an example output in Listing B.7. Users are named from `u1`, with first being all the monolingual ones and then all the bilingual ones.

These preparatory scripts assume that the current working directory contains two dataset files. The first is `squad-train-v2.0.json`, which is available from the SQuAD 2.0 website[6]. The second one is the Czech translation of the same file with structure preserved. It is hosted by Matúš Žilinec[7] and should be stored with the name of `zilinec-train-v2.1.json`.

[6]rajpurkar.github.io/SQuAD-explorer

[7]zilinec.me/dl/datasets/squad/train-cs-v2.1.json

```
> ./prepare_questions/bake.py --cusers 2 --busers 2

s: SQuAD, z: SQuAD Czech, t: Tech issues, p: Praha 6
     s  z  t  p
u0: 00 60 35 30
u1: 00 60 35 30
u2: 33 27 35 30
u3: 20 40 35 30

Intersections (in %)
     u0  u1  u2  u3
u0  100 100  73  84
u1  100 100  73  84
u2   73  73 100  75
u3   84  84  75 100

Histogram: X stimuli used by Y users
31: 1, 22: 2, 31: 3, 83: 4
```
List of Listings B.7: Example output of baked queue generator

Data extraction

Stimuli for the technical and administrative domains were chosen manually. For SQuAD and SQuAD Czech, we extracted the stimuli at random using `prepare_questions/data_prep.py`. We used this program to explore the SQuAD structure and also to pick stimuli of interest. As described by Table 5.2, we wanted to focus on how many questions SQuAD had per given stimuli and sample within a specific distribution. When using this data extraction program with the `extract_distribution` command, it prints the SQuAD distribution of questions per span. It also samples the spans according to the hardcoded distribution. This is shown in Listing B.8.

```
> ./prepare_questions/data_prep.py \
    extract_distribution \
    output_raw.json

SQuAD 2.0 per span question distribution:
    {1: 81619, 2: 2303, 3: 166, 5: 8, 4: 13, 6: 1}
Total selected spans: 60
Total SQuAD 2.0 spans: 84110
```
List of Listings B.8: Output of SQuAD data extraction tool

The output of this command is a JSON array, which contains tuples of the original SQuAD 2.0 and SQuAD 2.0 Czech paragraphs. For the purposes of our experiment, we also need to assign stimuli keys which we call `SID`. They are in the form of `<domain letter><number>`. The domain letters are shown in Listing B.7. For example the first stimuli of the SQuAD domain has `SID` of `s00`. To achieve this, we create a new JSON with the keys: `tech_issues`, `praha_6` and `squad`. The values for the first two are just arrays of strings and for `squad` it is the previous output. This file can then be used as an argument for the last data preparation command which adds these keys. Assuming the new edited file is stored in `all_questions_raw.json`, the keys can be added using as in Listing B.9.

```
> ./prepare_questions/data_prep.py \
    add_keys \
    all_questions\_raw.json \
    all_questions.json
```

List of Listings B.9: Usage of data preparation tool command which adds `SIDs` to stimuli

The file `all_questions.json` from Listing B.9 should now contain the JSON object which can be added to the main experiment definition file described in Section 4.2.

B.4.2 Logs

Files

The raw logs are stored in an interleaved CSV file (grepping it by the first token results in a regular CSV file). The collected files are stored with anonymized userIDs at `meta/study_pilot/logs/raw/`. The file `meta/study_pilot/logs/a0_fixed.csv` contains the quality annotation of produced outputs.

Processing scripts

For the purpose of this thesis we use files with the `.blog` extension. They simply contain pickled Python data. The logs are split into segments (each corresponding to one stimulus) and the timestamps are normalized concerning the segment's beginning. Furthermore, the lines also contain the annotator's ID.

To create the `.blog` file, use the `meta.py` command. The file can then be used by other scripts that analyze the logs. Quality estimation annotation has to be added to these files as well using the `extract_qe_annotation`. This process is shown in Listing B.10. We list the processing files along with their brief description:

- `meta.py` aggregates all of the raw logs into a single pickled file.
- `domains.py` explores phenomena in segments across domains, such as the number of linearly written stimuli.
- `prep_qe_annotation.py` creates a markdown (`qe_annotation/a0.md`) and a CSV file which can be used by a proficient annotator to assess the quality of the produced output. We turned the markdown file into a HTML page (`qe_annotation/a0.html`) using Pandoc.[8] These files were

[8]pandoc.org

then used by this annotator and afterward, their work was merged using `extract_qe_annotation`.

- `process_qe_annotation.py` processes quality annotations on collected data with graphics output which is used in this thesis and Zouhar and Bojar (2020). The matplotlib module is required.
- `segments.py` aggregates the log to some human-readable form that can be later analyzed. Examples are only written inputs (SR1) and the last translation request before confirmation (SR5). This was useful for us to examine the data.
- `load.py`, `create_blog.py` and `utils.py` are used only as modules from other scripts.

```
> ./processing_scripts/meta.py ./logs/raw/* -b main.blog

>  ./processing_scripts/prep_qe_annotation.py ./main.blog \
    ./prepare_questions/questions_flat.json \
    --a0md a0.md --a0csv a0.csv

> ./processing_scripts/extract_qe_annotation.py \
    main.blog main.blog \
    ./logs/qe_annotation/a0_fixed.csv

> ./processing_scripts/domains.py ./main.blog

SQuAD 141 (100s)
- skipped: 11
- finished: 130
- - linear: 50
- - with edits: 78
- - - avg similarity: 68.86%
- - - equal: 68.31%
- - - replace: 26.39%
- - - insert: 0.00%
- - - delete: 5.31%
SQuAD-cs 346 (94s)
- skipped: 26
- finished: 320

...

> ./processing_scripts/segments.py ./main.blog \
    -sr1 out.sr1 -sr2 out.sr2 -sr3 out.sr3 \
    -sr4 out.sr4 -sr5 out.sr5
```

List of Listings B.10: Example of binary log file creation and manipulation

Errors

During the experiment, we committed three mistakes. The first concerns a memory leak which made the user interface less responsive over time. This was fixed after the experiment.

The rest regards the annotation of produced translations. The second error is connected to the IDs in the CSV and HTML file given to the proficient annotator. Luckily this could be solved later by running the script `./processing_scripts/fix_qecsv_usid.py ./main.blog ./logs/qe_annotation/a0_bad.csv ./logs/qe_annotation/a0_fixed.csv`. The file `a0_bad.csv` is the one returned by the annotator and `a0_fixed.csv` is the file with the correct mappings that can be used with other scripts.

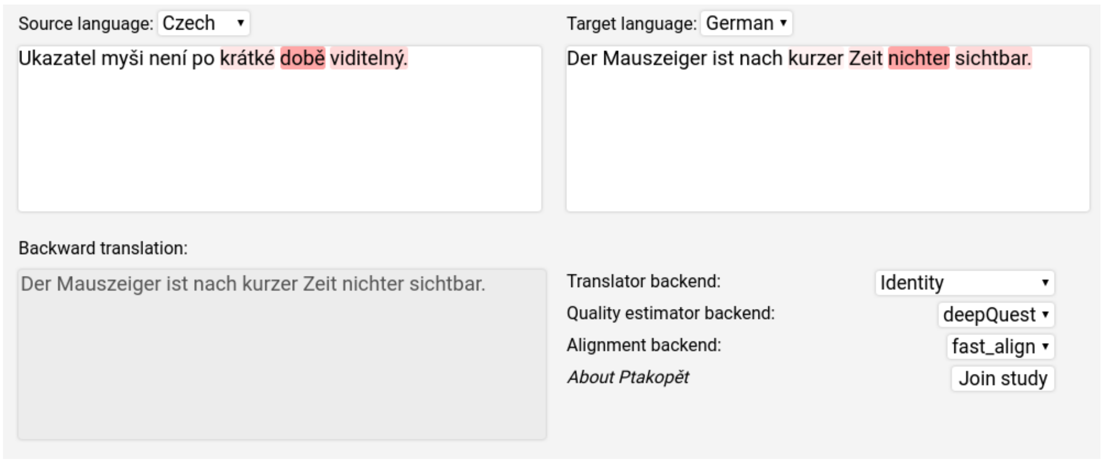
The other error is a methodological one. The markdown/HTML file used by the annotator by which they rated the translation quality contained visible distinctions in IDs between first viable and final outputs. Since the first viable were thought to be of lesser quality, the annotator could get biased in their ratings.

C. Instructions for annotators

The following document in Czech is what was send (with minor styling edits) to the annotators from the experiment described in Chapter 5. It briefly introduces the user interface and the annotation task with an example.

Ptakopět Pilot Experiment

Ptakopět je nástroj pro práci s překladačem. Hlavní je levé horní textové pole, obsahující zdrojový text, pravé horní zobrazující text přeložený strojovým překladem a levé spodní, zobrazující zpětný překlad (z cizího jazyka zpět do zdrojového). První dvě zmíněná pole je možné editovat.


Source language: Czech

Ukazatel myši není po krátké době viditelný.

Target language: German

Der Mauszeiger ist nach kurzer Zeit nichter sichtbar.

Backward translation:

Der Mauszeiger ist nach kurzer Zeit nichter sichtbar.

Translator backend: Identity

Quality estimator backend: deepQuest

Alignment backend: fast_align

About Ptakopět

Join study

Červené zabarvení v druhém poli indikuje, že překlad je v tomto místě nějakým způsobem problematický, jde ovšem o automatický odhad, který může být zavádějící. V prvním okně se pak podbarveně zobrazují slova, která pravděpodobně odpovídají těm problematickým v překladu.

Studie

Úvod

Systém je nasazen na webové adrese: ptakopet.vilda.net. Doporučujeme krátké seznámení se systémem mimo samotnou studii. V případě nejasností se, prosím, obraťte na mail zouhar@ufal.mff.cuni.cz. Studie se týká použití Ptakopětu k překladu do němčiny pro uživatele, kteří německy neumí.

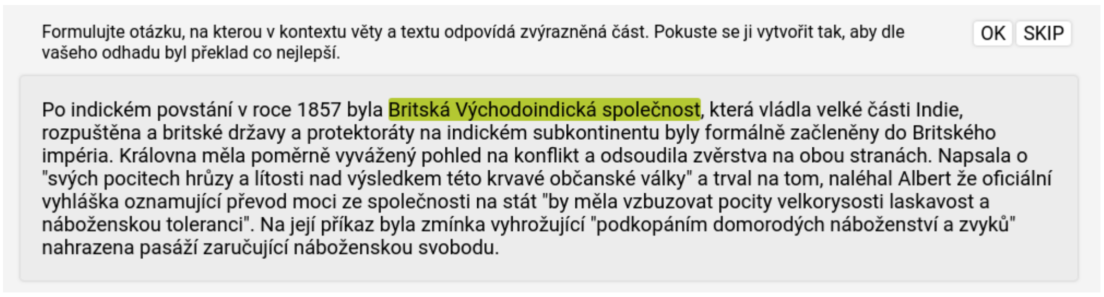
Po celou dobu práce je zapotřebí být připojen k internetu. Po zmáčknutí tlačítka Join study se zobrazí dialog, do kterého je potřeba vložit vaše ID, které jste od nás dostali mailem. Pokud se vám po potvrzení zobrazí hláška Unknown user ID., bylo zadáno špatné ID a je třeba stránku načíst a zadat ID znovu.

Příklady

V případě úspěšného přihlášení se zobrazí první z vašich příkladů spolu s krátkou instrukcí. Ty jsou čtyř charakterů:

1. Popis daného problému technické podpoře, která komunikuje pouze německy.
2. Formulace otázky v němčině, na kterou v kontextu věty odpovídá zvýrazněná část českého odborného textu.
3. Formulace otázky v němčině, na kterou v kontextu věty odpovídá zvýrazněná část anglického odborného textu.
4. Formulace otázky v němčině, na kterou v kontextu věty odpovídá zvýrazněná část českého administrativního textu.

První druh příkladu obsahuje navíc červené podbarvení, které signalizuje kvalitu překladu, ovšem ne směrodatně. Ukázka příkladu druhého druhu:


Formulujte otázku, na kterou v kontextu věty a textu odpovídá zvýrazněná část. Pokuste se ji vytvořit tak, aby dle vašeho odhadu byl překlad co nejlepší.

Po indickém povstání v roce 1857 byla Britská Východoindická společnost, která vládla velké části Indie, rozpuštěna a britské državy a protektoráty na indickém subkontinentu byly formálně začleněny do Britského impéria. Královna měla poměrně vyvážený pohled na konflikt a odsoudila zvěrstva na obou stranách. Napsala o "svých pocitech hrůzy a lítosti nad výsledkem této krvavé občanské války" a trval na tom, naléhal Albert že oficiální vyhláška oznamující převod moci ze společnosti na stát "by měla vzbuzovat pocity velkorysosti laskavost a náboženskou toleranci". Na její příkaz byla zmínka vyhrožující "podkopáním domorodých náboženství a zvyků" nahrazena pasáží zaručující náboženskou svobodu.

Prvním řešením (pište do prvního textového pole) může být např. *Co bylo rozpuštěno po indickém povstání v roce 1857?* Po překladu se však může ukázat, že různé obraty jsou zpětně špatně přeloženy. V takovém případě je třeba původní otázku nadále reformulovat, dokud nebudete s překladem do němčiny spokojeni (jde samozřejmě jen o váš odhad, německy neumíte).

Pohyb mezi příklady

Po dokončení práce s konkrétním příkladem klikněte na tlačítko *OK* a zobrazí se další. V případě, že jsou s nějakým příkladem problémy, můžete ji přeskočit tlačítkem *SKIP* v takovém případě je však třeba vhodné udat důvod. Přesnější popisy jsou pro studii přínosnější.

Přerušení

Váš postup je uchováván v rámci jednoho prohlížeče na jednom zařízení. Tj. zařízení i konkrétní prohlížeč lze vypínat. Po znovunačtení stránky v prohlížeči a zadání uživatelského ID by se měl zobrazit příklad, u které jste naposledy skončili. Informace o aktuálním příkladu bude ztracena v případě smazání historie, cookies, nebo dat z webu (většinou v nastavení prohlížeče).

Konec

Množství otázek přesahuje váš čas pro tento projekt (zhruba 6 hodin). Až to nastane, napište mail na zouhar@ufal.mff.cuni.cz. Zde můžete buď skončit, nebo dále pokračovat, dokud nedojde zásoba otázek. Mzda je hodinová a přesný čas, který jste prací s příklady strávili se měří automaticky a ukládá na serveru.

Poznámka

Během průběhu experimentu jsme stihli zaznamenat jeden technický problém. Některým uživatelům se po 5-10 minutách používání Ptakopětu snižuje responzibilita stránky Ptakopětu. Pokud se vám to stane, načtěte prosím stránku znovu. Jedná se o technický nedostatek, který se nám bohužel na poslední chvíli nepodařilo odstranit.