J. Hlaváčová (Ed.): ITAT 2017 Proceedings, pp. 201–208
CEUR Workshop Proceedings Vol. 1885, ISSN 1613-0073, © 2017 R. Rosa

ITAT

MonoTrans: Statistical Machine Translation from Monolingual Data

Rudolf Rosa

Charles University, Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics,
Malostranské náměstí 25, 118 00 Prague, Czech Republic
rosa@ufal.mff.cuni.cz

Abstract: We present MonoTrans, a statistical machine translation system which only uses monolingual source language and target language data, without using any parallel corpora or language-specific rules. It translates each source word by the most similar target word, according to a combination of a string similarity measure and a word frequency similarity measure. It is designed for translation between very close languages, such as Czech and Slovak or Danish and Norwegian. It provides a low-quality translation in resource-poor scenarios where parallel data, required for training a high-quality translation system, may be scarce or unavailable. This is useful e.g. for cross-lingual NLP, where a trained model may be transferred from a resource-rich source language to a resource-poor target language via machine translation. We evaluate MonoTrans both intrinsically, using BLEU, and extrinsically, applying it to cross-lingual tagger and parser transfer. Although it achieves low scores, it does surpass the baselines by respectable margins.

1 Introduction

In machine translation (MT), the most common and most successful approach is to train a translation model from parallel text corpora, i.e. from a set of bilingual sentence pairs with corresponding meanings. This approach was pioneered by the IBM models [2], which led to the development of many phrase-based MT systems, with Moses [8] being the most well-known and wide-spread one. In recent years, Neural MT [1] is taking the lead, with one of the main representants being Nematus [15]. Still, all of these systems rely on parallel corpora as the key resource.

Fortunately, parallel text corpora are a naturally occurring resource. They can be mined from film subtitles, book translations, documents published by international institutions, software localization data, etc. Probably the largest freely available collection of parallel data is Opus [18], providing parallel corpora for roughly 100 languages for download,[1] comprising many smaller preexisting collections. However, rough estimates of the number of world's languages are in thousands, which means that for the vast majority of existing languages, parallel data are not available easily, or not available at all.

A common feature of language that is not usually exploited in main-stream MT, is interlingual word similarity. Typically, the systems treat the source language words and the target language words completely independently, usually by representing each word with a unique identifier, with source word identifiers and target word identifiers belonging to different domains. Only in case of out-of-vocabulary source words (OOVs), which are not part of the available source language vocabulary and therefore cannot be translated by the system, approaches that do acknowledge potential interlingual word similarity are sometimes applied, such as transliteration by a character-based translation model [4]; although, in most cases, OOVs are simply left untranslated.[2]

In our work, we create a data-driven MT system for very close languages, based on utilizing only monolingual corpora and a set of heuristics with a high level of language independence. In this way, we target languages which are very low on available resources: the only resource we require for both the source and the target language is a plain-text monolingual corpus, i.e. any text written in that language (even a short one). Arguably, this is the lowest possible requirement to perform any text-based processing of a language: at least a textual input must be available, otherwise there is nothing to process. The key assumption behind our approach is that corresponding words often have the following two properties:

- They are similar on the character level, i.e. the string similarity of the source word and the corresponding target word is often high.
- They appear in the language with a similar frequency, i.e. the frequency of the source word in a source language corpus and the frequency of the target word in a target language corpus is usually similar.

While these assumptions obviously do not hold in general, we believe that they are mostly valid in case of very close languages (such as Czech and Slovak or Danish and Norwegian, which we use in our evaluation).

Our general approach to translating a given source language word is to look through all of the target words present in our corpus, and to return the most similar one

[1] http://opus.lingfil.uu.se/

[2] This can also be understood as a very rough way of acknowledging interlingual word similarity, in the sense that it is implicitly assumed that the unknown source word may happen to be also used in the target language in an identical form. This assumption may often be true, especially in case of named entities, which constitute a major share of OOVs. Still, even in such cases, transliteration or similar transformations may be necessary to obtain the correct target word form.

as the most probable translation, based on the two dimensions of similarity noted above and described in detail in Section 3.

In practice, an exhaustive search over the full target vocabulary is not viable. Therefore, we introduce a number of heuristics to speed up the search, and describe the whole translation process, in Section 4.

In Section 5.1, we evaluate MonoTrans instrinsically with BLEU [12]. As could be expected, our method delivers a very low quality translation, far beyond the best reported translation scores for the evaluated language pairs. However, our focus is on scenarios where none of the better-performing approaches are applicable, as neither parallel data nor a rule-based translation system are available. In that regard, the only baseline for us is to leave the text untranslated, which we surpass by large margins.

The quality of MonoTrans translation is too low to be useful when targeting humans; moreover, for a speaker of the target language, a similar close language is typically partially intelligible even without translation. In fact, it is exactly this partial cross-lingual intelligibility of similar languages, common with humans but generally inaccessible to machines, that we want to simulate with MonoTrans. We focus on the task of cross-lingual transfer of trained NLP tools, namely part-of-speech (POS) taggers and dependency parsers, where even a low-quality translation can provide the tools with a partial understanding of an unknown language and allow them to be applied to that language, even if their performance will inevitably be low. We evaluate MonoTrans extrinsically in this setup in Section 5.2.

2 Related Work

While being a stranger in data-driven machine translation, interlingual word similarity has often been utilized in rule-based MT, in particular when translating between similar languages. While rule-based MT has generally been superseded by statistical MT, a number of fully rule-based or hybrid[3] machine translation systems do exist, such as Apertium [5]; for an overview of MT systems for related languages, see e.g. [20]. Moreover, when focusing on special classes of words, such as technical terminology, systematic interlingual word similarity can be exploited even across very distant languages, such as Czech and English, as investigated already in [7]. However, these systems still require sets of language-specific rules, large bilingual dictionaries, and/or parallel corpora, to perform the end-to-end translation. To the best of our knowledge, devising a machine translation system for such a low-resource setting is rather unique.

The somewhat solitary work of Irvine and Callison-Burch [6] does go extraordinarily far in a similar direction to ours, estimating the correspondence of words based on a large number of predictors, including both orographic similarity and frequency similarity as we do,[4] and also using contextual similarity, temporal similarity, topic similarity, etc. However, most of these predictors rely on at least small amounts of bilingual data, in the form of parallel corpora, bilingual dictionaries, and/or comparable corpora; some also require other meta data, such as segmentation of the data into documents, or a time stamp marking the date of creation of the text. In our work, we omit the predictors which require such additional data, and focus on fine-tuning the two most resource-light predictors instead – the string similarity and the frequency similarity.

There is also a handful of older work attempting to construct a bilingual lexicon and/or to perform machine translation without parallel corpora, most notably by Koehn and Knight [9], Persman and Padó [13], Ravi and Knight [14], and Vulić and Moens [21].

3 Interlingual Word Similarity

The key component of MonoTrans is a word similarity measure, composed of a string similarity sim_{str} and a frequency similarity sim_f; the string similarity is itself composed of a Jaro-Winkler-based similarity sim_{jw*} and a length-based similarity sim_l:

$$\begin{aligned} sim(w_{src}, w_{tgt}) &= sim_{str}(w_{src}, w_{tgt}) \cdot sim_f(w_{src}, w_{tgt}) \\ sim_{str}(w_{src}, w_{tgt}) &= sim_{jw*}(w_{src}, w_{tgt}) \cdot sim_l(w_{src}, w_{tgt}) \end{aligned} \tag{1}$$

where w_{src} and w_{tgt} are the source and target word, respectively. The following subsections provide detailed descriptions of each of these components.

3.1 Jaro-Winkler-Based Similarity

Our string similarity measure is based on the Jaro-Winkler (JW) similarity [22], which has an interesting property of giving more importance to the beginnings of the strings than to their ends. This nicely suits our setting, as in flective languages, most of the inflection usually happens at the end of the word, while the beginning of the word tends to carry more of the lexical meaning. Thus, we expect the JW similarity to give more weight to the similarity of the meanings of the words than to the particular inflected forms in which they appear.[5]

However, JW similarity does not account for a number phenomena that are common in interlingual word similarity. We believe the following two to be of the highest importance:

[3] A hybrid MT system is a system which combines rule-based and statistical components.

[4] Interestingly, the authors seem to use a measure of frequency similarity very close to ours, although the provided formula (4) seems to be inverted by mistake, measuring frequency *dissimilarity* instead.

[5] From another perspective, we could say that JW similarity implicitly performs a simple soft stemming of its arguments.

- diacritical marks tend to be cross-lingually inconsistent, and languages are usually intelligible even when diacritics are stripped from the text,
- consonants tend to carry more meaning than vowels and tend to be more cross-lingually consistent.

Therefore, we introduce two preprocessing steps that can be employed to simplify the word forms before the computation of the JW similarity: transliteration to ASCII, and devowelling.

The **transliteration to ASCII**, provided by the `unidecode` Python module,[6] maps all characters into ASCII, trying to replace each non-ASCII character by one "near what a human with a US keyboard would choose". However, it does not handle non-alphabetic scripts, such as Chinese or Japanese. We denote transliteration of a word w by $T(w)$.

The **devowelling** strips all vowel characters, i.e. all characters that, after transliteration to ASCII, belong to the following group: *a, e, i, o, u, y*. We denote devowelling of a word w by $D(w)$.

The JW-based similarity sim_{jw*} is then computed as a multiplication of several components:

$$sim_{jw*}(w_{src}, w_{tgt}) = \prod_{J \in \{jw, jwT, jwD, jwDT\}} sim_J(w_{src}, w_{tgt}) \quad (2)$$

The first component, sim_{jw}, is the JW similarity without any preprocessing. However, as it is undefined for empty words (ε), we modify it slightly:

$$sim_{jw}(w_{src}, w_{tgt}) = \begin{cases} \frac{1}{1+len(w_{src})} & \text{if } w_{tgt} = \varepsilon \\ \frac{1}{1+len(w_{tgt})} & \text{if } w_{src} = \varepsilon \\ sim'_{jw}(w_{src}, w_{tgt}) & \text{otherwise} \end{cases} \quad (3)$$

where $len(w)$ is the number of characters in word w, and sim'_{jw} is the Jaro-Winkler similarity provided by the `pyjarowinkler` Python module.[7][8]

The following components are the JW similarity of transliterated words (sim_{jwT}), the JW similarity of devowelled words (sim_{jwD}), and the JW similarity of transliterated and devowelled words (sim_{jwDT}):

$$\begin{aligned} sim_{jwT}(w_{src}, w_{tgt}) &= sim_{jw}(T(w_{src}), T(w_{tgt})) \\ sim_{jwD}(w_{src}, w_{tgt}) &= sim_{jw}(D(w_{src}), D(w_{tgt})) \\ sim_{jwDT}(w_{src}, w_{tgt}) &= sim_{jw}(D(T(w_{src})), D(T(w_{tgt}))) \end{aligned} \quad (4)$$

3.2 Length-Based Similarity

A target word that is significantly shorter or longer than a given source word is unlikely to be its translation; however, we found that the Jaro-Winkler similarity for such a pair of words is often too high. Therefore, we introduce an additional penalty for words that differ in length:

$$sim'_l(w_{src}, w_{tgt}) = \frac{1}{1 + L \cdot |len(w_{src}) - len(w_{tgt})|} \quad (5)$$

where $len(w)$ is the length of word w. The length importance L is used to put less weight on length similarity than on the other string similarity components; we use $L = 0.2$.

The length similarity is computed both on the original words as well as on their devowelled variants:

$$sim_l = sim'_l(w_{src}, w_{tgt}) \cdot sim'_l(D(w_{src}), D(w_{tgt})) \quad (6)$$

3.3 Frequency Similarity

We expect corresponding words to appear with a similar frequency in the monolingual corpora. Of course, one of them may be several times more frequent than the other, but their frequencies should be similar at least in orders of magnitude. Thus, we compare the logarithms of the frequencies to calculate the similarity:

$$sim'_f(w_{src}, w_{tgt}) = \frac{1}{1 + |log(f_{w_{src}}) - log(f_{w_{tgt}})|} \quad (7)$$

The occurrence frequencies are computed from the monolingual corpora:

$$f_{w_{corpus}} = \frac{count_{corpus}(w) + S}{size_{corpus}} \quad (8)$$

using a smoothing factor S that allows us to output, with a low probability, even unknown words; we use $S = 0.1$.

Such a measure seems to be appropriate for corpora of similar sizes. However, if one of the corpora is significantly smaller than the other, the frequencies of words in the smaller corpus are somewhat boosted due to the smaller number of word types appearing in the corpus among which the total mass is distributed. Therefore, we downscale the frequencies computed on the smaller corpus by upscaling its size used in (8):

$$size_A = \begin{cases} |A| \cdot \sqrt{\frac{|B|}{|A|}} & \text{if } |A| < |B| \\ |A| & \text{otherwise} \end{cases} \quad (9)$$

where $|X|$ denotes the number of words in the corpus X.

We found that the definition of frequency similarity in (7) does a good job in removing many bad target language candidates; usually these are very infrequent words that are by chance string-wise similar to the source word. However, we also found that it inappropriately boosts target words with a low similarity to the source word that by chance have an extremely similar frequency. Thus, we need to keep the similarity harsh for low values, but soften it for high values. Therefore, if the value of sim'_f is higher than a threshold T_f, we push the part of it which is above

[6] `https://pypi.python.org/pypi/Unidecode`

[7] `https://pypi.python.org/pypi/pyjarowinkler`

[8] Interestingly, the module method `get_jaro_distance` does *not* provide the Jaro-Winkler *distance* d_{jw}, but the Jaro-Winkler *similarity* $1 - d_{jw}$; i.e., a value of 1 corresponds to identical strings, and the value of 0 to completely dissimilar strings.

the threshold down, by multiplying it by a decay factor D_f:

$$sim_f = \begin{cases} T_f + D_f \cdot (sim'_f - T_f) & \text{if } sim'_f > T_f \\ sim'_f & \text{otherwise} \end{cases} \quad (10)$$

We use $T_f = 0.5$ and $D_f = 0.1$.

3.4 Discussion

While the similarity measure we use is intended to be language-independent, we do acknowledge that it was hand-tuned particularly on the Czech-Slovak language pair, as the authors have a strong knowledge of both of these languages, and may not be fully adequate for all language pairs. In particular, we expect it to work best for flective languages with a preference for word-final inflection. For an optimal performance, it should be hand-tuned or machine-tuned on a more diverse set of languages.

The transliteration component is useless for languages that only use ASCII characters. Moreover, due to its implementation, it cannot handle non-alphabetic languages, such as Chinese or Japanese.

Also, we fail to identify systematic differences in the languages, such as "w" in Polish consistently corresponding to "v" in Czech or Slovak. We believe that the method would highly benefit from being able to find such correspondences automatically in the monolingual data, e.g. by exploring the distributions of character unigrams or *n*-grams, and/or by employing an EM-like approach to find a likely mapping.

Finally, the measure completely ignores the fact that words with similar meanings can be expected to appear in similar contexts. While reflecting on that fact would most probably make the computation of the similarity much more complicated and slower, it could allow the method to detect even corresponding words that are dissimilar according to the string similarity measures. A more viable approach could be to at least account for the fact that a given target word is likely to appear in similar target contexts, as mediated e.g. by a language model.

4 The Translation System

The MonoTrans translation system consists of two components: a training component, and a translation component.

The **training component** creates a pair of word frequency lists, based on source and target monolingual corpora. Any monolingual corpora can be used for the training, with larger corpora leading to better results. The frequency similarity measure works more reliably when the source and target corpora are of similar sizes, at least in orders of magnitude. However, if the source language is very low on available resources, the input text to be translated can by itself also serve as the only source corpus.

The **translation component** then performs a word-based 1:1 monotone translation, trying to translate each source word by the most similar target word from the target word list, based on the similarity measure described in the previous section. The translation of each word is performed independently.

4.1 Computational Efficiency

In theory, for each input source word, the translation component could always go through all target words in the target language word list, measure the similarity of the source word and each candidate target word, and then emit the most similar target word as the translation.

However, this is only feasible in cases where the target word list is very small, containing hundreds or at most thousands of words, allowing us to translate each source word in a matter of seconds at most. Once the target word list goes into tens of thousands of words and beyond (which it usually does in our experiments), the translation times become far too long for an exhaustive search to be practical.

Therefore, we introduce a range of heuristics and technical measures, both lossless and lossy, to sufficiently speed up the translation process while trying to keep the translation quality as high as possible. We describe the most important two of them in the following subsections; we also use other less interesting measures, such as caching of method calls.

4.2 Word List Partitioning

The main speedup comes from a hard partitioning of the word lists, which is the only lossy procedure we employ. Following our observations in Section 3, we assume that for a pair of corresponding source and target words:

- the lengths of the devowelled transliterated words differ by at most 1,
- the first two characters of the devowelled transliterated words are identical.

None of these assumptions hold universally, but we believe that they do hold for a vast majority of words that can be translated by our system (i.e. words that are sufficiently similar to their target counterparts). Most importantly, they let us only go through a tiny part of the target word list when translating a source word, bringing a key speedup to the translation system.

Thus, instead of using a flat word list, the training component stores each word in a specific partition, addressed by a compound key, consisting of the first two transliterated consonants of the word and the length of the transliterated devowelled word. The translation component then only traverses three of these partitions, corresponding to the first two transliterated consonants of the source word and the length of the transliterated devowelled source word, increased by +1, 0, and -1.

4.3 Frequency-Based Early Stopping

Even after the partitioning, many of the partitions are too large to be traversed exhaustively for each matching source word. We can deal with that issue thanks to the following two observations:

- word frequency similarity is a powerful component of our similarity measure,
- words in a language have a Zipf-like distribution, with a small number of frequent words and a high number of rare words.

Therefore, we can achieve a significant speedup by ordering the words in each partition descendingly by frequency, and introducing the following early-stopping criterion: once we reach a target word so infrequent that its frequency-based similarity to the source word alone is lower than the total similarity of the most similar target word found so far, we can stop processing the current partition, as none of the remaining target words would be able to surpass the currently best candidate; i.e., we stop once:

$$f_{w'_{tgt}} < f_{w_{src}} \wedge sim_f(w_{src}, w'_{tgt}) < sim(w_{src}, w^*_{tgt}) \quad (11)$$

where w'_{tgt} is the current target word candidate, and w^*_{tgt} is the best target word found so far.

As we, by definition, encounter frequent words more frequently than rare words, this allows us to often skip the processing of the long tail of infrequent words; it only gets processed if the source word is a rare one, or if it is not sufficiently similar to any of the frequent target words.

5 Evaluation

In this section, we evaluate MonoTrans both intrinsically and extrinsically. However, for the real under-resourced languages with no parallel data and no annotated corpora available, there is no way for us to perform an automatic evaluation, and a manual evaluation would be difficult to obtain for us. Therefore, as is usual in these scenarios, we simulate the under-resourced setting by evaluating on pairs of similar but resource-rich languages, allowing us to use standard automatic evaluation measures. Specifically, we use the following language groups in our experiments:

- Czech (cs) and Slovak (sk),
- Danish (da), Norwegian (no) and Swedish (sv),
- Catalan (ca) and Spanish (es).

Please note that we hand-tuned our method partially based on brief manual inspections of the results on the sk-cs pair.

5.1 Instrinsic Evaluation

We first evaluate the quality of the MonoTrans translation itself with BLEU [12]. We use the OpenSubtitles2016 subcorpus of the Opus collection [10], which contains translated movie subtitles from the OpenSubtitles website.[9] We use the first 10,000 target sentences and the last 10,000 source sentences for training, and then evaluate the source-to-target translation quality on the last 10,000 sentences; i.e. the source side of the evaluation data is used for training, but the target side, which serves as the reference translation, is not.

Table 1: Evaluation of MonoTrans with BLEU.

Langs	SrcLex	MonoTrans	Rel.diff.
cs-sk	10.1	13.1	30%
sk-cs	10.1	14.8	46%
da-no	16.8	18.3	8%
sv-no	7.7	14.7	90%
no-da	16.6	17.7	7%
no-sv	7.7	12.5	63%
ca-es	5.5	8.3	51%
es-ca	5.4	7.9	46%
AVG	**10.0**	**13.4**	**43%**

Table 2: Evaluation of MonoTrans with BLEU, using large monolingual corpora for training.

Langs	SrcLex	MonoTrans	Rel.diff.
cs-sk	10.1	15.6	55%
sk-cs	10.1	17.1	70%
AVG	**10.1**	**16.4**	**62%**

Table 1 shows the BLEU scores achieved by MonoTrans, compared to the SrcLex baseline, i.e. to performing no translation at all; thanks to the high similarity of the languages, even the baseline achieves a non-trivial translation score. The BLEU scores are rather low, reaching 13.4 on average, whereas a state-of-the-art MT system trained on large amounts of parallel data could easily reach scores around 30 BLEU points (or probably even more, provided that the source and target languages are very similar). However, we can see a large and consistent improvement over the baseline of 3.4 BLEU points in average. We also report the relative improvement over the baseline which, thanks to the very low scores achieved by the baseline, is very high, reaching up to 90% (for sv-no) and 43% on average.

Generally, we do not expect very large corpora to be available for under-resourced languages. Still, to measure the scaling potential of our method, we also evaluated MonoTrans trained on significantly larger Czech and Slovak monolingual corpora. For this experiment, we used large web corpora, namely CWC for Czech [16] and skTenTen for Slovak [3]. We used the first 100 millions of words from each of the corpora for training, i.e. roughly a thousand times larger datasets, and then evaluated the translation system on the same OpenSubtitles data as in the previous experiments.

The results in Table 2 show that increasing the data size improves the translation quality, with the improvement over the SrcLex baseline being nearly doubled. However, considering the factor by which we increased the training data size, we find the improvement to be rather moderate.

[9] `http://www.opensubtitles.org/`

Table 3: Evaluation of MonoTrans with BLEU, using smaller corpora for training; "0" corresponds to SrcLex, "10,000" is identical to Table 1.

Training sentences	cs-sk	sk-cs
0	10.1	10.1
10	10.0	10.0
100	11.3	11.4
1,000	12.4	12.3
10,000	13.1	14.8

Table 4: Examples of outputs of the Czech–Slovak and Slovak–Czech translation.

Language	Text
sk source	Mal ho len vystrašiť aby ho udržal mlčať.
cs transl.	Měl ho loni vystrašit aby ho udržel mlčet.
cs correct	Měl ho **jen** vystrašit aby ho udržel mlčet.
sk source	Počúvaj, nemám rád prípady, ako je tento, ale som vďačný za to, že ste ho chytili.
cs transl.	**Počkají**, nemám rád případy, **ako** je tento, ale **sám** vděčný za to, že **set** ho chytili.
cs correct	**Poslouchej**, nemám rád případy, **jako** je tento, ale **jsem** vděčný za to, že **jste** ho chytili.
cs source	Když zemřela, neměla jsem chuť o tom mluvit.
sk transl.	**Kde** zomrela, nemala **jsem** chuť o tom **milovať'**.
sk correct	**Keď'** zomrela, nemala **som** chuť o tom **hovorit'**.

Next, we tried to downscale the training data instead, training and evaluating the system identically to Table 1 but using only a part of the training data. Table 3 shows that already with 100 monolingual non-corresponding sentences for each of the languages, i.e. very modest data, improvements of over +1 BLEU over the non-translation baseline can be achieved.

Finally, in Table 4, we show several examples of the inputs and outputs of the MonoTrans system, taken from the evaluation datasets translated by the systems trained on the large datasets; the "correct" translation is not the reference translation, but a corrected version of the MonoTrans output, and errors are highlighted. We can see many correctly translated words in the outputs, as well as many words correctly left untranslated. Moreover, many of the errors can be easily accounted to the word list partitioning that we employ, making it impossible for MonoTrans to perform translations such as "len–jen", "som–jsem", or "počúvaj–poslouchej". This suggests that many of the errors are actually search errors, not scoring errors, and could be eliminated if we had a better way of efficiently searching for candidate target translations.

5.2 Extrinsic Evaluation

We also evaluated MonoTrans extrinsically, in the task of cross-lingual transfer of trained NLP models across closely related languages, inspired by the cross-lingual parsing shared task of the VarDial 2017 workshop [23].

Table 5: Evaluation of MonoTrans in cross-lingual POS tagger transfer, using tagging accuracy.

Langs	SrcLex	MonoTrans	Supervised	Err.red.
cs-sk	78.0	82.7	94.1	29%
sk-cs	70.9	76.7	98.3	21%
da-no	76.8	78.3	97.0	8%
sv-no	64.7	72.8	97.0	25%
no-da	78.7	80.4	95.5	10%
no-sv	56.0	72.6	95.1	42%
ca-es	76.7	78.1	96.2	7%
es-ca	69.9	81.1	98.0	40%
AVG	**71.5**	**77.8**	**96.4**	**23%**

Table 6: Evaluation of MonoTrans in cross-lingual parser transfer, using LAS.

Langs	SrcLex	MonoTrans	Supervised	Err.red.
cs-sk	46.6	56.3	68.7	44%
sk-cs	35.9	42.9	73.1	19%
da-no	45.8	49.0	79.4	9%
sv-no	30.2	40.0	79.4	20%
no-da	46.3	48.7	71.4	10%
no-sv	24.0	41.7	69.4	39%
ca-es	44.4	48.2	77.4	11%
es-ca	39.5	51.3	80.3	29%
AVG	**39.1**	**47.3**	**74.9**	**23%**

The task constitutes of using an annotated corpus of a resource-rich source language to train an NLP model in such a way that it can be applied to analyzing a different but very similar resource-poor target language.

We loosely follow the approach of Tiedemann et al. [19], proceeding in the following steps:

- Translate the words in an annotated source corpus into the target language by an MT system.
- Train a lexicalized model on the resulting corpus.
- Apply the model to target language data.

Specifically, we employ the Universal Dependencies (UD) v1.4 treebanks [11] as the annotated data, MonoTrans as the translation tool, and the UDPipe tagger and parser [17] as the models to be trained.

The MonoTrans system is trained using the word forms from the training part of the source treebank and the development part of the target treebank, and applied to translate the training part of the source treebank into the target language. Then, the UDPipe tagger and parser are trained on the resulting corpus; the tagger is trained to predict the Universal POS tag (UPOS) based on the word form, and the parser is trained to predict the labelled dependency tree based on the word form and the UPOS tag predicted by the tagger. Finally, both the tagger and the parser are applied to the development part of the target language treebank, and evaluated against its gold-standard annotation.

We report the tagger accuracy in Table 5, and the parser LAS[10] in Table 6. As a baseline, we also include SrcLex, i.e. using a tagger and parser trained on an untranslated source treebank, and as an upper bound, we include a supervised tagger and parser, trained on the training part of the target treebank; as the languages are very similar, the baselines are quite strong. This allows us to also compute the error reduction, i.e. the proportion of the gap between the baseline and the upper bound filled by our method.

The taggers reach an average accuracy of 77.8% and the parsers an average LAS of 47.3%, which is not much in absolute terms – when large parallel data are available, LAS scores around 60% can be reached. However, in relative terms, the scores are rather impressive, obtaining an average 23% error reduction in both the tagging and the parsing, and reaching average absolute improvements of +6.3 in tagging accuracy and +8.2 in parsing LAS. Given our setting, we find the results to be wonderful, provided that we only used small monolingual corpora to train the MT system; in fact, in the target language, we only used the evaluation input data, which is probably the lowest imaginable data requirement.

6 Further Possible Improvements

6.1 Language Model Scoring

As we mentioned in Section 3.4, a clear shortcoming of our method is the fact that the translation is performed in a context-independent way. Employing an n-gram language model is a standard way of getting better machine translation outputs, only plaintext target-language data are needed to create one, and there already exists a plethora of state-of-the-art ready-to-use language modelling tools. Therefore, it may seem straightforward to employ a language model in MonoTrans as well.

However, there is a range of technical issues that need to be overcome. If we were able to generate a translation lexicon, we could then even easily plug it into a full-fledged MT system, such as Moses, easily combining it with a language model; however, generating the lexicon would be computationally prohibitive in our case, for reasons mentioned in Section 4.1. At best, we could potentially try to generate a translation lexicon only for the words that appear in the test data. Moreover, even using a beam search in MonoTrans decoding is too costly for us, as it prohibits the employment of the early stopping mechanisms.

So far, we have only managed to perform a set of preliminary experiments, adding a simple trigram language model and using its score as an additional scoring component; as we found that using the score directly had a too strong and negative effect on the translations, we weakend it by taking its fourth root. When evaluated on the large Czech and Slovak corpora in both directions, we observed only negligible improvements around +0.1 BLEU. We believe that this is mainly due to the fact that our approach in these preliminary experiments was too rough and simplistic, and that with proper tuning and a more sophisticated implementation, clear improvements may be gained.

6.2 Better Searching for Candidate Translations

Based on inspection of the translation outputs, as well as from the examples in Table 4, it is clear that the word list partitioning is way too crude, preventing the system from generating the correct translation in many cases, even though its similarity to the source word is sufficiently high. On the other hand, it is completely impossible for the system to search through all possible translations, and some kind of harsh pruning of the search space is vital.

As a quick remedy, we tried to use the trigram language model to generate additional translation candidates. Specifically, for each source word we also investigated N candidate translations taken from N words that are, according to the language model, the most likely to follow the words selected as translations of the previous words. With $N = 20$, we observed a promising improvement of +0.6 BLEU for cs-sk, while the translation times remained competitive (they doubled). With $N = 1000$, the improvement for sk-cs further jumped to +1.3 BLEU; however, at this point, the translation became about 50 times slower (taking 10 hours to translate 10,000 sentences), showing that this approach is somewhat promising in terms of translation quality but too computationally demanding. For sk-cs, negligible or no improvements were observed.

An interesting possibility of clustering the search space which was suggested to us is to use a standard clustering algorithm, such as k-means or hierarchical k-means, with the word similarity used as the distance of the target words. This is expected to be permissibly fast to compute as well as to allow a sufficiently fast search for translation candidates; however, due to time constraints, we have not been able to design an experiment to test that.

7 Conclusion

We presented MonoTrans, a data-driven translation system trained only on plaintext monolingual corpora, intended for low-quality machine translation between very similar languages in a low-resource scenario.

We showed that even with very small training corpora available, the system shows respectable performance according to both intrinsic and extrinsic evaluation, consistently surpassing the no-translation baseline by large margins. Moreover, we showed that the system performance scales with larger training data, even though rather slowly.

In particular, when evaluated extrinsically as a component of cross-lingual tagger and parser transfer, employing MonoTrans leads to high improvements in both tagging accuracy and parser LAS with respect to the baselines, achieving an average 23% error reduction in both of

[10] Labelled Attachment Score, i.e. the number of correctly predicted labelled dependency relations in the output tree.

the tasks when supervised models are taken as the upper bounds.

Acknowledgments

This work has been supported by the grant No. DG16P02B048 of the Ministry of Culture of the Czech Republic, the grant No. CZ.02.1.01/0.0/0.0/16_013/0001781 of the Ministry of Education, Youth and Sports of the Czech Republic, and the SVV 260 453 grant. This work has been using language resources and tools developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071). We would also like to thank the anonymous reviewers and our colleagues from the ÚFAL MT group (especially Jindřich Libovický) for helpful comments and suggestions for improvement.

References

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

[2] Peter Brown, John Cocke, S Della Pietra, V Della Pietra, Frederick Jelinek, Robert Mercer, and Paul Roossin. A statistical approach to language translation. In *COLING*, pages 71–76. Association for Computational Linguistics, 1988.

[3] Masaryk University NLP Centre. skTenTen, 2011. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

[4] Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. Integrating an unsupervised transliteration model into statistical machine translation. In *EACL*, volume 14, pages 148–153, 2014.

[5] Mikel L Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M Tyers. Apertium: a free/open-source platform for rule-based machine translation. *Machine translation*, 25(2):127–144, 2011.

[6] Ann Irvine and Chris Callison-Burch. End-to-end statistical machine translation with zero or small parallel texts. *Journal of Natural Language Engineering*, 22:517–548, 2016.

[7] Zdeněk Kirschner. On a device in dictionary operations in machine translation. In *COLING*, COLING '82, pages 157–160, Czechoslovakia, 1982. Academia Praha.

[8] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *ACL*, pages 177–180. Association for Computational Linguistics, 2007.

[9] Philipp Koehn and Kevin Knight. Learning a translation lexicon from monolingual corpora. In *ULA*, ULA '02, pages 9–16, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[10] Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC*, 2016.

[11] Joakim Nivre et al. Universal dependencies 1.4, 2016. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

[12] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. Association for Computational Linguistics, 2002.

[13] Yves Peirsman and Sebastian Padó. Cross-lingual induction of selectional preferences with bilingual vector spaces. In *HLT-NAACL*, HLT '10, pages 921–929, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[14] Sujith Ravi and Kevin Knight. Deciphering foreign language. In *ACL-HLT*, HLT '11, pages 12–21, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[15] Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. Nematus: A toolkit for neural machine translation. *EACL 2017*, page 65, 2017.

[16] Johanka Spoustová and Miroslav Spousta. A high-quality web corpus of Czech. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *LREC*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[17] Milan Straka, Jan Hajič, and Jana Straková. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, pos tagging and parsing. In *LREC*, Paris, France, May 2016. European Language Resources Association (ELRA).

[18] Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In *LREC*, volume 2012, pages 2214–2218, 2012.

[19] Jörg Tiedemann, Željko Agić, and Joakim Nivre. Treebank translation for cross-lingual parser induction. In *CoNLL*, 2014.

[20] Jernej Vičič, Vladislav Kuboň, and Petr Homola. Česílko goes open-source. *The Prague Bulletin of Mathematical Linguistics*, 107(1):57–66, 2017.

[21] Ivan Vulic and Marie-Francine Moens. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *EMNLP*, pages 1613–1624. ACL, 2013.

[22] William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods (American Statistical Association)*, pages 354–359, 1990.

[23] Marcos Zampieri, Shervin Malmasi, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Jörg Tiedemann, Yves Scherrer, and Noëmi Aepli. Findings of the VarDial evaluation campaign 2017. In *VarDial*, Valencia, Spain, 2017.