



Semantic and syntactic information for neural machine translation

Injecting Features to the Transformer

Jordi Armengol-Estapé¹  · Marta R. Costa-jussà²

Received: 25 February 2020 / Accepted: 1 April 2021 / Published online: 18 May 2021
© The Author(s) 2021

Abstract

Introducing factors such as linguistic features has long been proposed in machine translation to improve the quality of translations. More recently, factored machine translation has proven to still be useful in the case of sequence-to-sequence systems. In this work, we investigate whether this gains hold in the case of the state-of-the-art architecture in neural machine translation, the Transformer, instead of recurrent architectures. We propose a new model, the Factored Transformer, to introduce an arbitrary number of word features in the source sequence in an attentional system. Specifically, we suggest two variants depending on the level at which the features are injected. Moreover, we suggest two combination mechanisms for the word features and words themselves. We experiment both with classical linguistic features and semantic features extracted from a linked data database, and with two low-resource datasets. With the best-found configuration, we show improvements of 0.8 BLEU over the baseline Transformer in the IWSLT German-to-English task. Moreover, we experiment with the more challenging FLoRes English-to-Nepali benchmark, which includes both low-resource and very distant languages, and obtain an improvement of 1.2 BLEU. These improvements are achieved with linguistic and not with semantic information.

Keywords Transformer · Factored neural machine translation · Linguistic features · Semantic features

✉ Jordi Armengol-Estapé
jordi.armengol.estape@ed.ac.uk

Marta R. Costa-jussà
marta.ruiz@upc.edu

¹ Barcelona Super Computing Center, University of Edinburgh, Barcelona, Spain

² Universitat Politècnica de Catalunya, Barcelona, Spain

1 Introduction

Many classical Natural Language Processing (NLP) pipelines used either linguistic or semantic features (Koehn and Hoang 2007; Du et al. 2016), including machine translation applications. In recent years, the rise of neural architectures has diminished the importance of the aforementioned features. Nevertheless, some works have still shown the effectiveness of introducing linguistic information into neural machine translation systems, typically in recurrent sequence-to-sequence (seq2seq) architectures (Sennrich and Haddow 2016; García-Martínez et al. 2016; España-Bonet and van Genabith 2018).

The motivation for studying strategies for incorporating linguistic or semantic information into state-of-the-art neural machine translation systems is two-fold. On the one hand, it can slightly improve the results in generic settings. On the other hand, and most importantly, it can play a key role in major challenges for machine translation, such as low-resource settings and morphologically different languages. In this work, we provide successful use cases for both situations. We suggest a modification to adapt the current state-of-the-art neural machine translation architecture, the Transformer (Vaswani et al. 2017), to working with an arbitrary number of factors such as linguistic or semantic features. Specifically, we study the effect of incorporating such features at the embedding level or at the encoder level, with two different combination strategies and using different linguistic and semantic features, the former being extracted from NLP taggers and the latter from a linked data database. We report improvements in IWSLT and FLoReS benchmarks.

By factored Neural Machine Translation (NMT), we refer to the use of word features alongside words themselves to improve translation quality. Both the encoder and the decoder of a Seq2seq architecture can be modified to obtain better translations (García-Martínez et al. 2016). The most prominent approach consists of modifying the encoder such that instead of only one embedding layer, the encoder has as many embedding layers as factors, one for words themselves and one for each feature, and then the embedding vectors are concatenated and input to the rest of the model, which remains unchanged (Sennrich and Haddow 2016). The embedding sizes are set according to the respective vocabularies of the features. Notice that in the latter the authors applied Byte Pair Encoding (BPE) (Sennrich et al. 2016), an unsupervised preprocessing step for automatically splitting words into subwords with the goal of improving the translation of rare or unseen words. Thus, the features had to be repeated for each subword. García-Martínez et al. (2016) used the exact same architecture, except that this new proposal used concepts extracted from a linked data database, BabelNet (Navigli and Ponzetto 2012). These semantic features, synsets, were shown to improve zero-shot translations. All the cited works obtained moderate improvements with respect to the BLEU scores of the corresponding baselines.

The main goal of this work, and differently from previous works using NMT architectures based on recurrent neural networks, is to modify the Transformer to make it compatible with factored NMT with an architecture that we call Factored

Transformer and inject classical linguistic knowledge, especially from lemmas, which is the best performing feature in other works (Sennrich and Haddow 2016), and concepts extracted from the semantic linked data database, BabelNet (Navigli and Ponzetto 2012). We are focusing our attention on low-resource datasets.

2 Factored transformer

Unlike the vanilla Transformer (Vaswani et al. 2017), the Factored Transformer can work with factors; that is, instead of just being input the original source sequence, it can work with an arbitrary number of feature sequences. Those features can be injected at embedding-level, as in the previous works we described above (but in a Transformer instead of a recurrent-based seq2seq architecture), or at the encoder level. We have implemented the two model variants.

2.1 1-Encoder model

Similarly to Sennrich and Haddow (2016), this architecture consists of only one encoder, but multiple embedding layers (one for each factor and one for the words themselves). The embedding vectors are looked up for each factor separately. Then, the respective outputs are combined. In Sect. 2.3, we will discuss the different combination strategies. Finally, positional embeddings are summed to the combined embedding vector, and the first attention/feed-forward layer of the Transformer's encoder is input this vector as in the vanilla model. Positional embeddings are summed to the final vector and not to the individual embeddings because we are modifying the individual token embeddings, not the length of the sequence, which remains unchanged, so the positions in the sequence are the same. We can think of these embedding layers as a *factored embedding layer*. Apart from the embedding layer of the encoder, the rest of the Transformer remains unchanged. This model variant is depicted in Fig. 1.

2.2 N-encoder architecture

Using multiple encoders for factored NMT, one for each factor, is not a common approach in the literature. In fact, we are not aware of any other work that does so. Instead, it has been used for multi-source MT, which is a task that consists in translating from multiple sources (ie. languages) referring to the same sentence into a single target sequence, always with RNN-based models as far as we know, as Zoph and Knight (2016). We had the intuition that having one encoder for each factor could improve the results, particularly for features that have a vocabulary size comparable to the ones of the words (specially, lemmas and semantic features).

In this architecture, we have N encoders, one for each factor. These encoders are not different from the one of the vanilla Transformer. Instead of combining the outputs of the embedding layers of one encoder as in the single-encoder architecture, in this architecture the outputs of the multiple encoders are combined (*factored*

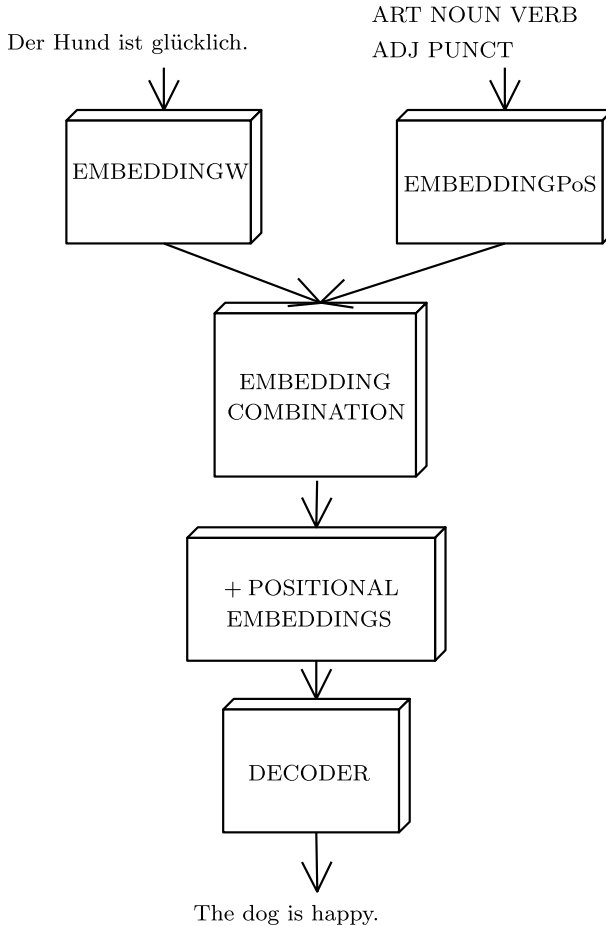


Fig. 1 1-Encoder model: In this example, a tokenized sentence in English and the corresponding Part-of-Speech tags are input to the encoder. The respective embedding outputs are combined, which can be done in different ways. Then, positional embeddings, which in the Transformer are used for encoding the position of each token in the sequence, are summed, and these vectors are input to the remaining layers of the encoder. The rest of the model remains unchanged. Notice that positional embeddings are summed to the combined embeddings and not to the individual embeddings separately because we are modifying the individual token embeddings, not the length of the sequence, which remains unchanged, so the positions do not change

encoder). The rest of the model remains unchanged. This architecture variant is depicted in Fig. 2.

2.3 Combination strategies

Sennrich and Haddow (2016) proposed that features and words embeddings were combined by the means of concatenating. However, it is not the only possible combination strategy. OpenNMT (Klein et al. 2017) allows both summation and

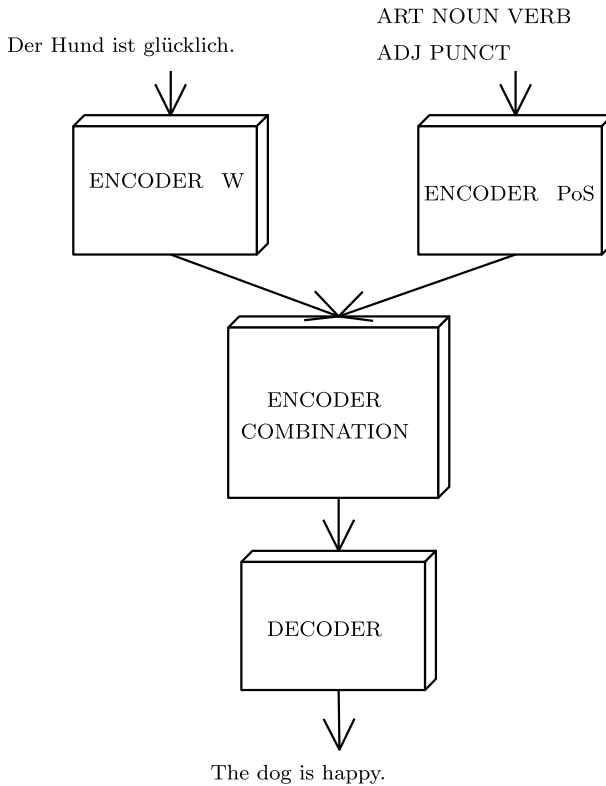


Fig. 2 N-encoder model: In this example, a tokenized sentence in German and the corresponding Part-of-Speech tags are input to their respective encoders. The encoders remain unchanged with respect to the vanilla Transformer. Each encoder has its own embedding layer and positional embeddings. The encoder outputs are combined, which can be done in different ways. The rest of the model remains unchanged. The decoder for the target sequence generates the sentence from the combined encoder outputs in the target language, in this example, English

concatenation. In the multi-source MT work we mentioned, the authors tried both concatenation and a tangent activation (as in LSTMs) and a full-blown LSTM for combining the outputs of the two encoders.

If we have N neural modules, F_i with $1 \leq i \leq N$, one for each feature and one for the words themselves, an input to each of them of dimensions $[\# \text{ tokens in sentence}]^1$ and an output from each of them, \mathbf{H}_i with $1 \leq i \leq N$, of dimensions $[\# \text{ tokens in sentence}, F_i \text{ embedding_size}]$, we can consider the following combination strategies:

¹ Here we are considering the inputs to the neural modules themselves. Usually, inputs are batched into a number of sentences, such that we have dimensions $[\# \text{ sentences}, \# \text{ tokens per sentence}]$, but this fact does not affect our observations in this section.

- Concatenation: The respective outputs of each F_i , \mathbf{H}_i , are concatenated, such that the resulting dimension is:

$$\left[\# \text{tokens in sentence}, \sum_{i=1}^N \text{embedding size of } F_i \right]$$

and input to the next layer. Obviously, all F_i share the number of tokens (and the number of sentences if we were considering the full batch), because they are referring to the same sentences, but notice that is not necessary for the embedding sizes to be equal. The operation can be represented as:

$$[\mathbf{H}_1; \mathbf{H}_2; \dots; \mathbf{H}_N]$$

where ';' is the tensor concatenation along the corresponding axis.

- Summation: The respective outputs of each F_i are summed and input to the next layer, instead of concatenated. Thus, the embedding size of each factor must be equal, and the resulting dimension is [# tokens in sentence, embedding size of each F_i]:

$$\mathbf{H}_1 + \dots + \mathbf{H}_N = \sum_{i=1}^N \mathbf{H}_i$$

2.4 Embedding sizes

The bigger the embedding size, the more vocabulary can be learnt. However, at some point the embedding layer can start to overfit. In addition, we do not want the embedding size to be that big that it makes the model too computationally expensive.

In OpenNMT, the embedding size cannot be specified for each feature. Instead, either all of them have the same size, or the embedding size is computed from this formula²:

$$\text{embedding size}_i = \text{vocabulary size}_i^{\text{feature exponent}} \quad \forall i \in \text{Features}$$

With feature exponent = 0.7 by default. This formula tries to capture the principle that the embedding size should grow with the vocabulary size. In practice, it can result in large, infeasible embeddings for features with big vocabulary sizes, and it is not flexible enough to let introduce a different exponent for each feature.

Sennrich and Haddow (2016) used a total size of the embedding layer of 500. Features with a few dozens of possible values were assigned an embedding size of 10, for instance, while words had an embedding size of 350. However, in practice the Transformer is known to need bigger embeddings (Popel and Bojar 2018). It

² OpenNMT training options: <http://opennmt.net/OpenNMT-py/options/train.html>.

could be the case that a word embedding size smaller than 512 (without counting the features) was not enough for learning the words vocabulary.

Needless to say, the total embedding size of the encoder should match the embedding size of the decoder, because otherwise there would be a dimension mismatching.

Another issue to consider is the fact that while RNN-based embeddings can be of arbitrary dimensions, in the Transformer they must be a multiple of the number of attention heads. Recall from previous sections that the multi-head attention mechanism applies multiple blocks in parallel (one for each head), concatenates their outputs and finally applies a linear transformation. Notice that as many outputs as the number of heads will be concatenated. Thus, the embedding dimension must be divisible by the number of heads. Otherwise, there would be a dimension mismatching.

The reader will notice that the last restriction affects the total embedding size of the encoder, but not the individual embedding sizes. That is to say, in the N-encoder architecture, each factor should verify the restriction. In contrast, this does not need to be the case for each factor in the 1-encoder architecture (the total sum must be a multiple of the number of heads, but not each embedding individually). Ideally, the embedding size of each factor should be optimized by some search procedure. In practice, we will have computational limitations for doing so.

One alternative for avoiding overfitting would be to freeze the weights (ie. stop updating weights) at an early epoch, but only for the features with small vocabulary size, and keep training and updating the rest of the model. We implemented this feature as well, although it did not obtain noticeable gains.

3 Linguistic and semantic features

In the previous section, we detailed the neural architectures that we are proposing in this work. Their key addition is that they allow to introduce multiple sequences referring to the same source sentence, *factors*. Thus, we are going to be able to input both the words (or subwords) themselves, the first factor, and a set of features associated to them, the other factors.

In this section, we go through the features that we are going to consider to use alongside words in order to try to improve MT. Intuitively, linguistic information should be useful for getting a better representation of the text.

However, one novelty presented in this work is the study of the use linked data, which has barely been exploited in NMT (García-Martínez et al. 2016). In particular, we extracted semantic and language-agnostic information. In addition, it is worth noting the implications of using features alongside subwords (instead of whole words).

3.1 Linguistic features

Sennrich and Haddow (2016) proposed the use of the following features:

- Lemmas.
- Part-Of-Speech tags.
- Word dependencies
- Morphological features.

The vocabulary size of each feature, which as we have said in the previous section is very relevant to the size of their respective embeddings, is very small in most of the cases, except in the case of lemmas. Lemmas have a vocabulary size of the same order of magnitude than the original words (without subword splitting), but smaller, because all the different inflections and derivations of a word will collapse into the same base form.

There are different taggers available and nowadays some of them are based on deep learning. Sennrich and Haddow (2016) used Stanford's CoreNLP tagger Manning et al. (2014). For this work, we started with TreeTagger, a Part-Of-Speech tagger that also offers lemmatization, but later on we pivoted to Stanford models, through a Spacy Python wrapper³.

As a matter of fact, linguistic taggers are not infallible. Even though they are supposed to help the NLP system by providing additional information, at least in some cases they could be providing wrong annotations. In addition, tagging can be a resource-intensive task, and the corresponding neural network will need more parameters at least for dealing with the feature embeddings.

3.2 Linked data

As we have said, by linked data we mean structured data that forms a graph, that is, an ontology-based database. In the context of NLP, linked data is often used to describe a dataset such that thanks to its interlinked nature, semantic information can be extracted.

There are many different linked data databases that we could use with our proposed architectures, without any modifications, provided that words could be tagged by discrete identifiers. For instance, domain-specific linked data could be used for domain adaption of an NMT system, by inputting domain features, as we will further suggest as future work, in Sect. 5. Nevertheless, for this work we explore linked data for the general domain.

3.2.1 BabelNet

In particular, we are keen on BabelNet (Navigli and Ponzetto 2012), the linked data used by García-Martínez et al. (2016). BabelNet is a very large multilingual encyclopedic dictionary and semantic network.

³ spaCy + StanfordNLP: <https://github.com/explosion/spacy-stanfordnlp>.

BabelNet is automatically built from the combination of many sources⁴, such as Wikipedia, WordNet (a computational lexicon of different languages), Wikidata (a machine-readable knowledge base), Wiktionary (Wikimedia's multilingual dictionary), and ImageNet (an image database organized according to the WordNet hierarchy).

Apart from the large amount of data, its potential comes from the way the different sources are integrated. For instance, as we have said, ImageNet is organized according to WordNet's hierarchy, while Wikipedia entries are linked to Wikidata and Wiktionary terms.

Thanks to this deep integration, BabelNet makes available a large semantic network. In particular, there are approximately 16 million entries, called synsets. Synsets are language-agnostic, that is to say, the same concept will have the same synset assigned in all the 271 languages in which BabelNet is available. Perhaps even more importantly, synsets are shared for synonyms as well.

The reader will notice that BabelNet has a great potential for both entity recognition (identifying target entities in text, eg. identifying countries) and word sense disambiguation (determining to which particular meaning a polysemous word or expression may be referring to), which intuitively should be useful for improving MT, particularly the latter. In addition, unlike linguistic taggers, BabelNet is uniformly available for many languages.

Nevertheless, BabelNet itself is just a large and interlinked database with REST and Java APIs to extract word-level information. Plain BabelNet will retrieve all the possible synsets that a particular word may have, and links to possibly linked concepts. It is up to the developer to actually perform the sense disambiguation by building a rule based system or a MT application.

In García-Martínez et al. (2016), the authors used plain BabelNet id's, without sense disambiguation. Instead, for this work we are going to use Babelfy.

3.2.2 Babelfy

Babelfy⁵Moro et al. (2014) is a service based on BabelNet that performs both entity recognition and word sense disambiguation. Both its Java and REST APIs retrieve only one synset for each word given its context (that is to say, its sentence), instead of returning the list of all the possible synsets. We used Babelfy instead of plain Babelnet for obtaining the disambiguated synsets.

In Fig. 3 we provide an example of a sentence tagged with synsets.

⁴ See Wikipedia: <https://www.wikipedia.org/>, WordNet: <https://wordnet.princeton.edu/>, Wikidata: https://www.wikidata.org/wiki/Wikidata:Main_Page, Wiktionary: <https://www.wiktionary.org/> and ImageNet: <http://www.image-net.org/>, among others.

⁵ Babelfy: <http://babelfy.org/about>.

```

in|APPR der|ART zwischenzeit|NN ,|$, wurde|VAFIN die|ART alte|bn:00107787a
mentalität|bn:00054380n ,|$, mit|APPR grossem|ADJA a|FM ,|$, von|APPR
seinem|PPOSAT vater|bn:00009616n repräsentiert|bn:00090234v ,|$, dem|ART
vorsitzenden|ADJA der|ART nigerianischen|bn:00107300a bank|bn:00008364n ,|$,
der|PRELS die|ART cia|bn:00017182n vor|APPR seinem|PPOSAT sohn|bn:03854737n
warnte|bn:00095642v ,|$, der|PRELS kurz|bn:00098854a vor|APPR dem|ART
angriff|bn:00006994n stand|bn:00021644n ,|$, und|KON diese|PDAT
warnung|bn:17105905n traf|bn:00085339v auf|APPR taube|bn:00062350n ohren|NN
.|$.

```

Fig. 3 Synsets of a German sentence from IWSLT14 German-to-English. Notice that prepositions, articles and punctuation signs, among others, have been assigned their respective morphological features, since they did not have a defined synset. Features starting with `bn:` are BabelNet synsets. Most importantly, we can see that the word `bank`, which in German is polysemous as in English (the financial institution or the bench for sitting), is mapped to the synset `bn:00008364n`. If we search this synset in BabelNet (BabelNet: `bn:00008364n`: <https://babelnet.org/synset?word=bn:00008364n>), we can see that it is the one corresponding to financial institutions, as it should, since the sentence is talking about the chairman of a Nigerian bank.

3.3 Features and subword encoding

As we have seen, some form of subword encoding is required for achieving state of the art results in most MT tasks. However, linguistic features are usually word-level features. The same holds for BabelNet synsets, which in fact can even refer to a set of words instead of only one. In addition, if the number of word tokens and the number of token features did not agree, in most configurations we would probably end up with a dimension disagreement and we would not even be able to input that information to the neural network.

Therefore, we must align the tokens corresponding to the features to the tokens corresponding to subwords. That is to say, we must have the same number of subword tokens than word features, and the n -th feature must refer to the corresponding word of the n -subword. Also, we can take into account that the fact that one word feature belongs to, say, an intermediate subword, can have a slightly different meaning than the same feature belonging to the final subword. That is to say, the same way BPE adds the tag '@@' to the subword tokens in order to encode where the original word started, we could find an equivalent way of encoding this information in the feature side. We can consider different strategies for doing so or at least tackling this difficulty:

- Not using BPE: We discarded this option because the baseline would be weaker.
- Just repeating the word features for each subword: This approach has the benefit of being simple, trivially compatible with the proposed architectures and not increasing the vocabulary size of the features. However, it does not encode the fact that consecutive tags are part of the same word.
- Using '@@' in word features, the same was that this tag is used in BPE: Again, this approach has the benefit of being trivially compatible with architectures such as the multi-encoder architecture and the single-encoder architecture with summation instead of concatenation, and not requiring an additional feature, but it

has the drawback that it approximately doubles the vocabulary. The latter happens because the same feature can appear in different positions of the word. Therefore, there will be one token with '@@' and one token without this tag.

- The approach used by Sennrich and Haddow (2016): Repeating the word features for each subword and introducing a new factor, subword tags, to encode the position of the subword in the original word. The 4 possible tags are:
 - B: Beginning of word.
 - I: Intermediate subword.
 - E: End of word.
 - O: Whole word (not split into subwords).

This approach has the benefit of both encoding the position of the word which the subword belongs to and not increasing the vocabulary of the features. However, it is not compatible with the multi-encoder architecture, at least not trivially, because it would not make sense to have a specific encoder just for learning the 4 tokens of the vocabulary of subword tags. However, it could be used with a modified multi-encoder architecture by the means of a hybrid approach (each feature encoder having its own subword tags as an additional feature).

Each approach presents its own advantages and disadvantages and it is unclear which one should be followed. The procedure of choice should be the most practical one (eg. the one that it is compatible with our architecture, or the simplest one) or the one that performs better experimentally.

4 Experiments

4.1 Preliminary experiments

We experimented with BPE alignment strategies (including the approaches from Sect. 4.2), and different classical linguistic features (lemmas, part-of-speech, word dependencies, morphological features). The preliminary experiments showed that BPE alignment strategies were not very relevant, so we adopted the alignment with BPE by repeating the word feature. In addition, we found that the most promising classical linguistic feature was lemmas, consistently with the results obtained by Sennrich and Haddow (2016).

4.2 Data

The first experiments were conducted with a pair composed of similar languages, the German-to-English translation direction of the IWSLT14 (Cettolo et al. 2014), which is a low-resource dataset (the training set contains about 160,000 sentences). For cleaning and tokenizing, we use the data preparation script proposed by the authors of Fairseq (Ott et al. 2019). As test sets, we took the test sets from the corpus released for IWSLT14 and IWSLT16. The former was used to test the

best configuration, and the latter was used to see the improvement of this configuration in another set. A joint BPE (ie. German and English share subwords) of 32,000 operations is learned from the training data, with a threshold of 50 occurrences for the vocabulary. The second round of experiments was conducted with the English-to-Nepali translation direction of the FLoRes Low Resource MT Benchmark (Guzmán et al. 2019). Although this pair has more sentences than the previous one (564,000 parallel sentences), it is considered to be extremely low-resource and far more challenging because of the lack of similarity between the involved languages. In this case, we learn a joint BPE of 5000 operations (both with an algorithm based on BPE, sentencepiece (Kudo and Richardson 2018), as proposed by the FLoRes authors, and with the original BPE algorithm).

4.3 Hyperparameters and configurations

In the case of German-to-English, we used the Transformer architecture with the hyperparameters proposed by the Fairseq authors: specifically, 6 layers in the encoder and the decoder, 4 attention heads, embedding sizes of 512 and 1024 for the feedforward expansion size, a dropout of 0.3 and a total batch size of 4000 tokens, with a label smoothing of 0.1. For English-to-Nepali, we used the baseline proposed by the FLoRes authors: specifically, 5 layers in the encoder and the decoder, 2 attention heads, embedding sizes of 512 and 2048 for the feedforward expansion size and a total batch size of 4000 tokens, with a label smoothing of 0.2. In both cases, we used the Transformer architecture with the corresponding parameters we described above as the respective baseline systems, and we introduced the modifications of the Factored Transformer without modifying the rest of the architecture and parameters. As mentioned previously, linguistic features were obtained through StanfordNLP (Qi et al. 2018) and regarding the Babelnet synsets, we found that approximately 70% of the tokens in the corpus we used did not have an assigned synset and were therefore assigned part-of-speech.

4.4 Reported results

After the preliminary research, we report experiments with features (lemmas and synsets), architectures (1-encoder and N-encoders systems), and combination strategies (concatenation and summation). Regarding this preliminary research, we include the results using the exact same configuration as Sennrich and Haddow (2016) but with the Transformer instead of a recurrent neural network. We found that while this setting outperformed the baseline, using just lemmas with summation was slightly better, so we dropped the rest of the features, being the lemmas-only configuration both simpler and obtaining better results.

Then, for the best feature, lemmas, Table 1 compares different architectures, and it is shown that the best architecture is the 1-encoder with summation. Finally, the best performing system (lemmas with a 1-encoder and summation) is evaluated in another test set, IWSLT16. The selected model is relatively efficient, because it only

Table 1 BLEU results

IWSLT14			
MODEL	COMB.*	FEATURE	BLEU
Baseline	–	–	34.08
1-Encoder	Concat	All in Sennrich and Haddow (2016)	34.21
1-Encoder	Sum	Lemmas	34.35
1-Encoder	Sum	Babelnet	33.66
1-Encoder	Concat	Lemmas	27.10
N-encoders	Concat	Lemmas	33.58
N-encoders	Sum	Lemmas	9.71
IWSLT16			
Baseline	–	–	36.67
1-Encoder	Sum	Lemmas	37.46
FLoRES			
Baseline	–	–	3.06
1-Encoder	Sum	Lemmas	4.27

In bold, best results

needs an additional embedding layer with respect to the baseline, while the total embedding size does not have to be increased because the embeddings are summed instead of concatenated.

Once we had found that the 1-encoder Factored Transformer with summation and lemmas was a configuration performing well for low-resource settings, we applied this combination the more challenging Facebook Low Resource (FLoRES) MT Benchmark. Specifically, we wanted to compare how this architecture performs against the baseline reported in the original work of this benchmark. The authors report the results before applying backtranslation and with sentence piece, which is 4.30 BLEU. We reproduced that baseline and we got slightly better results (up to 4.38 BLEU). However, our system is designed to work with BPE, not sentencepiece, which is more challenging to align to features (since subwords coming from different words can be combined into a single token). Table 1 shows that our configuration clearly outperformed the baseline with BPE (almost 40% up), and was very close to the results with sentencepiece.

5 Conclusions

We have shown that the Transformer can take advantage of linguistic features. We have not found any configuration with the semantic ones that outperformed the baseline settings, but it should be further investigated. We conclude that the best configuration for the Factored Transformer was the 1-encoder model (with multiple embedding layers) with summation instead of concatenation. For the German-to-English IWSLT task, the best configuration for the Factored Transformer shows an improvement of 0.8 BLEU, and for the low-resource English-to-Nepali task, the improvement is 1.2 BLEU.

With regard to the proposed architectures, in general, the results show that the N-encoder architecture is not an effective strategy for factored NMT, at least the way we have implemented and used it, because the obtained translations are worse than the ones obtained with the single-encoder architecture (and even the baseline), while being more complex. It seems to be more prone to overfitting because features, which are not providing that much information, instead of having just an embedding layer have multiple self-attention layers. Moreover, we suspect that having a different encoder for each factor seems to produce disentangled representations and the decoder struggles at combining the different sources. This seems to be the reason why concatenation performs better than summation in the case of the multiple-encoder architecture, because with the former at least the decoder can learn to ignore or treat differently different parts of the factored sequence. In the case of the single-encoder architecture, summing seems to work better because it produces a more compact factored embedding and it allows the decoder embedding size not to be doubled (if there are two source factors), preventing overfitting.

As far as the used features are concerned, we hypothesize that lemmas outperform synsets because of Babelnet. When tagging, a big proportion of the tokens do not get a synset (as detailed before, in this case we apply a backup linguistic feature, namely part-of-speech). Instead, all words can be tagged with lemmas (even if tagging is not perfect and can give wrong lemmas in some cases). In addition, the use of semantic features (BabelNet) was intended to help at disambiguating, but some recent papers have shown that the Transformer is already good at disambiguating (Tang et al. 2018). Instead, lemmas can help by providing the normalized term of a given word that may be very infrequent in the training corpus (but its respective lemma might be frequent enough).

In future work, we suggest adapting the alignment algorithm to sentencepiece by combining features coming from different words into a single feature, provided their respective subwords have been merged into a single token. In addition, whether the advantage provided by linguistic features still holds once backtranslation has been applied and up to what point this holds should be investigated.

Acknowledgements This work is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 947657).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Cettolo M, Niehues J, Stüker S, Bentivogli L, Federico M (2014) Report on the 11th IWSLT evaluation campaign, IWSLT 2014. In: Proceedings of the international workshop on spoken language translation, Hanoi, Vietnam, p 57
- Du J, Way A, Zydron A (2016) Using BabelNet to improve OOV coverage in SMT. In: Proceedings of the tenth international conference on language resources and evaluation (LREC'16), European Language Resources Association (ELRA), Portorož, Slovenia, pp 9–15. <https://www.aclweb.org/anthology/L16-1002>
- España-Bonet C, van Genabith J (2018) Multilingual semantic networks for data-driven interlingua seq2seq systems. In: Proceedings of the LREC 2018 MLP-Moment Workshop, Miyazaki, Japan, pp 8–13
- García-Martínez M, Barrault L, Bougares F (2016) Factored neural machine translation. CoRR [arXiv: abs/1609.04621](https://arxiv.org/abs/1609.04621)
- Guzmán F, Chen P, Ott M, Pino J, Lample G, Koehn P, Chaudhary V, Ranzato M (2019) Two new evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. CoRR. [arXiv:abs/1902.01382](https://arxiv.org/abs/1902.01382)
- Klein G, Kim Y, Deng Y, Senellart J, Rush AM (2017) OpenNMT: open-source toolkit for neural machine translation. In: Proceedings on ACL. <https://doi.org/10.18653/v1/P17-4012>
- Koehn P, Huang H (2007) Factored translation models. In: Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL), Association for Computational Linguistics, Prague, Czech Republic, pp 868–876. <https://www.aclweb.org/anthology/D07-1091>
- Kudo T, Richardson J (2018) SentencePiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. In: Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations, association for computational linguistics, Brussels, Belgium, pp 66–71. <https://doi.org/10.18653/v1/D18-2012>
- Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D (2014) The Stanford CoreNLP natural language processing toolkit. In: Association for computational linguistics (ACL) system demonstrations, pp 55–60. <http://www.aclweb.org/anthology/P/P14/P14-5010>
- Moro A, Raganato A, Navigli R (2014) Entity linking meets word sense disambiguation: a unified approach. *Trans Assoc Comput Linguist* 2:231–244
- Navigli R, Ponzetto SP (2012) BabelNet: the automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif Intell* 193:217–250
- Ott M, Edunov S, Baevski A, Fan A, Gross S, Ng N, Grangier D, Auli M (2019) fairseq: a fast, extensible toolkit for sequence modeling. In: Proceedings of NAACL-HLT 2019: demonstrations
- Popel M, Bojar O (2018) Training tips for the transformer model. CoRR [arXiv:abs/1804.00247](https://arxiv.org/abs/1804.00247)
- Qi P, Dozat T, Zhang Y, Manning CD (2018) Universal dependency parsing from scratch. In: Proceedings of the CoNLL 2018 shared task: multilingual parsing from raw text to universal dependencies, association for computational linguistics, Brussels, Belgium, pp 160–170. <https://nlp.stanford.edu/pubs/qi2018universal.pdf>
- Sennrich R, Haddow B (2016) Linguistic input features improve neural machine translation. CoRR [arXiv: abs/1606.02892](https://arxiv.org/abs/1606.02892)
- Sennrich R, Haddow B, Birch A (2016) 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), Association for Computational Linguistics, Berlin, Germany, pp 1715–1725. <https://doi.org/10.18653/v1/P16-1162>
- Tang G, Müller M, Rios A, Sennrich R (2018) Why self-attention? A targeted evaluation of neural machine translation architectures. CoRR [arXiv:abs/1808.08946](https://arxiv.org/abs/1808.08946)
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008. <https://papers.nips.cc/paper/7181-attention-is-all-you-need>
- Zoph B, Knight K (2016) Multi-source neural translation. CoRR [arXiv:abs/1601.00710](https://arxiv.org/abs/1601.00710)