Crosslingual Sentence Embeddings for Neural Machine Translation

Anonymous EMNLP submission

Abstract

Corpus-based approaches to machine translation rely on the availability and quality of parallel corpora. In the case of neural machine translation, a large neural network is trained to maximise the translation performance on a given parallel corpus. This paper describes an unsupervised method for building cross lingual sentence embeddings that optimises word alignments. Embeddings can then be used to compute semantic similarity between sentences of different languages. We report competitive performance on two different tasks. Firstly, sentence embeddings are used to detect translation divergencies in parallel sentence pairs, hence allowing for filtering out noisy pairs. Then, we use sentence embeddings to discover parallel sentences on comparable corpora. We evaluate the presented model for different language pairs and data resources. The model can be used on any parallel corpus without any manual annotation.

1 Introduction

Parallel sentence pairs are the only necessary resource to build Machine Translation (MT) systems. In the case of neural MT, a large neural network is trained through maximising translation performance on a given parallel corpus. Therefore, the quality of an MT engine is heavily dependent upon the amount and quality of parallel sentences. Note that recent work on neural MT remove completely the need of parallel data following unsupervised methods (**??**) obtaining accuracy improvements over word-by-word statistical MT systems.

Unfortunately, parallel texts are scarce resources. There are relatively few language pairs for which parallel corpora of reasonable sizes are available, and even for those pairs, the corpora come mostly from few domains. To alleviate the lack of parallel data several approaches have been developed in the last years. They range from methods using non-parallel, or comparable data (**?????**) to techniques that produce synthetic parallel data from monolingual corpora (**??**) using in all cases automated alignment/translation engines that are prone to the introduction of noise in the resulting parallel sentences. Mismatches on parallel sentences extracted from translated texts are also reported (**??**). This problem is mostly ignored in machine translation, where parallel sentences are considered to convey the exact same meaning, and is particularly important for neural MT engines as suggested by (**?**).

Table 1 illustrates some examples of English-French parallel sentences which are not completely semantically equivalent. Divergencies are outlined using bold letters. Examples extracted from the OpenSubtitles corpus (**?**).

en	*What do you feel*, **Spock***?*
fr	*Que ressentez-vous?*
gl	*What do you feel?*
en	*How much do you get paid?*
fr	*T'es payé combien* **de l'heure***?*
gl	*How much do you get paid per hour?*
en	**That seems a lot.**
fr	**40 livres?**
gl	*40 pounds?*
en	*I brought you* **french fries***!*
fr	*Je t'ai rapporté des* **saucisses***!*
gl	*I brought you sausage!*

Table 1: Examples of semantically divergent parallel sentences. English (en), French (fr) and gloss of French (gl).

Different types of translation divergencies can be found in a parallel corpus: Additional segments are included on either side of the parallel sentences (first and second rows) most likely due to errors in sentence segmentation tools; Some translations may be completely uncorrelated (third row); Inaccurate translations are also included (fourth row). Note that divergent translations can be due to many different reasons (**?**), the study of which is beyond the scope of this paper.

In this work, we present an unsupervised

method for building cross lingual sentence embeddings in the form of continuous vectors. Embeddings are then used to measure semantic equivalence of their corresponding sentences. We evaluate the method on two different tasks: First, we show that translation accuracy can be improved after filtering out divergent sentence pairs in an English-to-French and an English-to-German translation task. Then, we also show competitive results when identifying parallel sentences in comparable corpora using the French-English data made available for the BUCC shared task (**?**).

The remainder of this paper is structured as follows. Section 2 overviews related work. We describe in detail the core of the neural similarity classifier in Section 3. We report experiments with the presented model on two different tasks in Section 4. Section 5 evaluates results. Finally, conclusions are drawn in Section 6 and further work is outlined in Section 7. All the code used in this paper is freely available[1].

2 Related Work

2.1 Translation Divergencies

Prior work on translation divergencies focused on reflecting that translation of one language into another results in very different forms (**?**) according to several categories (e.g. lexical, structural, categorical).

Attempts to measure the impact of translation divergencies in MT systems have focused on the introduction of noise in sentence alignments (**?**), showing that statistical MT systems are highly tolerant to noise, and that performance only degrades seriously at very high noise levels. In constrast, neural MT engines seem to be more sensitive (**?**), as they tend to assign high probabilities to rare events (**?**).

Efforts have been devoted to characterising the degree of semantic equivalence between two snippets of text in the same or different languages (**?**), a workshop devoted to an objective similar to our work. In (**?**), a monolingual sentence similarity network is proposed, making use of a simple LSTM layer to compute sentence representations. The authors show that a simple SVM classifier can be built on top of the sentence representations to achieve state-of-the-art results in a semantic entailment classification task. With the same objective, the system presented in (**?**) uses multiple convolutional layers and models pairwise word interactions.

Our work is inspired by (**?**) where the authors train a cross-lingual divergence detector using word alignments and sentence length features to train a linear SVM classifier. Their work shows that an NMT system trained only on non-divergent sentences yields slightly higher translation quality scores and requires clearly less training time. The same authors have recently updated their work in (**?**). The objective is the same as the neural network presented in (**?**) and their network further outperforms their previous work. Our work differs from the previous as we make use of a network with different topology. We model sentence similarity by means of optimising word alignment and in addition, we propose a simple method that employs alignment scores to not only detect but correct divergent sentences.

2.2 Extracting Parallel Sentences

As previously outlined, parallel texts are scarce resources as well as the only necessary resource to build Machine Translation systems. In contrast, there is an increasing amount of comparable corpora available on the Internet, a potential solution to alleviate the data scarcity problem.

Several approaches exist to extract parallel sentences from comparable corpora. In (**?**) is presented a maximum entropy classifier that can reliably determine whether a pair of sentences are translations of each other. A similar approach is proposed by (**?**), where a SMT system is used to translate the source language side of a comparable corpus to find candidate sentences on the target side. Word error rate and translation error rate measures are used to decide when a candidate is finally considered parallel. More recently, (**?**) present a method based on seed lexical translations, simple set expansion operations and the Jaccard similarity coefficient. The method obtains state-of-the-art results in the task of parallel sentence extraction from comparable corpora.

Closely related to the method proposed in this paper, different works employ continuous vector sentence representations as a preceding step to compute sentence similarity. In (**?**), word representations from bilingual word embeddings are used to build a convolutional network that predicts wether a sentence pair is aligned or not. (**?**) propose a recurrent network to estimate sentence em-

[1] https://github.com/anonymised

beddings followed by dense layers that estimates the conditional probability distribution that two sentences are parallel.

These approaches are different from ours FINISH

This strong interest in comparable corpora was further boosted after the apparition of the BUCC workshop series on Building and Using Comparable Corpora[2] which helped to provide a task definition, datasets and evaluation methods to assess the state of the art.

3 Neural Similarity Classifier

In this section we describe the architecture of our model, very much inspired by the work on Word Alignment in (**?**). Figure 1 illustrates the network.

In the following, we consider a source-target sentence pair (s, t) with $s = (s_1, ..., s_I)$ and $t = (t_1, ..., t_J)$. The model is composed of 2 Bi-directional LSTM subnetworks, net_s and net_t, which respectively encode source and target sentences. Since both net_s and net_t take the same form we describe only the source architecture.

The source-sentence Bi-LSTM network outputs forward and backward hidden states, $\overrightarrow{h}_i^{src}$ and $\overleftarrow{h}_i^{src}$, which are then concatenated into a single vector encoding the i^{th} word of the source sentence, $h_i^{src} = [\overrightarrow{h}_i^{src}; \overleftarrow{h}_i^{src}]$.

In addition, the last forward/backward hidden states (outlined using dark grey in Figure 1) are also concatenated into a single vector to represent whole sentences $h_{src} = [\overrightarrow{h}_I^{src}; \overleftarrow{h}_1^{src}]$. At this point a measure of similarity between sentence pairs can be obtained by cosine similarity:

$$sim(h_{src}, h_{tgt}) = \frac{h_{src} \cdot h_{tgt}}{||h_{src}|| * ||h_{tgt}||} \quad (1)$$

where two vectors (embeddings) with the same orientation have a cosine similarity of 1, while two vectors with opposed orientation have a similarity of -1, independent of their magnitude.

Similar to (**?**) our model extracts context information from source and target sentences and then computes simple dot-products to estimate word alignments. The objective function is computed at the level of words. To enable unsupervised training, we use an aggregation operation that summarizes the alignment scores for a given target word. A soft-margin objective increases scores for true target words while decreasing scores for target words that are not present. The aggregation function combines the scores of all source (or target) words for a particular target (or source) word and promotes source words which are likely to be aligned with a given target word according to the knowledge the model has learned so far.

Alignment scores $S(i, j)$ are given by the dot-product $S(i, j) = h_i^{src} \cdot h_j^{tgt}$, while aggregation functions are defined as:

$$\begin{aligned} aggr_s(i, S) &= \frac{1}{r}\, log\left(\sum_{j=1}^{J} e^{r * S(i,j)}\right) \\ aggr_t(j, S) &= \frac{1}{r}\, log\left(\sum_{i=1}^{I} e^{r * S(i,j)}\right) \end{aligned} \quad (2)$$

The loss function is defined as:

$$\begin{aligned} \mathcal{L}(src, tgt) = & \\ & \sum_{i=1}^{I} log\left(1 + e^{aggr_s(i,S) * \mathcal{Y}_i^{src}}\right) + \\ & + \sum_{j=1}^{J} log\left(1 + e^{aggr_t(j,S) * \mathcal{Y}_j^{tgt}}\right) \end{aligned} \quad (3)$$

where $\mathcal{Y}_i^{src}$ (or $\mathcal{Y}_j^{tgt}$) is a vector with reference labels containing -1 when the i^{th} source word (or j^{th} target word) is present in the translated target (or source) sentence, and $+1$ for divergent (unpaired) words.

For comparison purposes we also implemented the model introduced in (**?**). The network employs the same initial Bi-LSTM layers outlined in Figure 1 which encode source and target sentences into h_{src} and h_{tgt} vectors. The remaining of the network is illustrated by Figure 2, where sentence pair matching information is captured by using their element-wise product and absolute element-wise difference. Sentence similarities are finally estimated by feeding the matching vectors into a fully connected layer.

$$\begin{aligned} h^{(1)} &= h_{src} * h_{tgt} \\ h^{(2)} &= |h_{src} - h_{tgt}| \\ h^{(3)} &= tanh(W[h^{(1)}; h^{(2)}] + b^{(3)}) \\ h_{sim} &= W h^{(3)} + b^{sim} \end{aligned} \quad (4)$$

[2] https://comparable.limsi.fr/bucc2018/

This approach is different from our model since it uses a single end-to-end model to estimate the conditional probability distribution that two sentences are parallel.

For this second network the loss function is defined as:

$$\mathcal{L}(src, tgt) = log(1 + e^{h_{sim} * \mathcal{Y}}) \quad (5)$$

where $\mathcal{Y}$ is the reference label containing -1 when the given sentence pair is non-divergent (parallel) and $+1$ for divergent (unpair) examples.

Note that different from the network described in the original work, our implementation does not estimates a probability distribution but FINISH.

3.1 Training with Negative Examples

Training is performed by minimising Equation 3, for which examples with annotations for source $\mathcal{Y}_i^{src}$ and target $\mathcal{Y}_j^{tgt}$ words are needed.

As positive examples we use **paired** sentences of a parallel corpus. All words of the pair of sentences are labelled as parallel, $\mathcal{Y}_i^{src} = -1$ and $\mathcal{Y}_j^{tgt} = -1$. As negative examples we use random **unpaired** sentences. In this case, all words are labelled as divergent, $\mathcal{Y}_i^{src} = +1$ and $\mathcal{Y}_j^{tgt} = +1$. Since negative examples may be very easy to classify and we want our network to detect less obvious divergencies, we further create negative examples following the next two methods:

We **replace** random sequences of words on either side of the sentence pair by a sequence of words with the same part-of-speeches. The rationale behind this method is to keep the new sentences as grammatical as possible. Otherwise the network can learn to detect non-grammatical sentences to predict divergence. Words that are not replaced are considered parallel (-1) while those replaced are assigned the divergent label ($+1$). Words aligned to some replaced words are also assigned the divergent label ($+1$). For instance, given the original sentence pair:

src:	`What do you feel ?`
tgt:	`Que ressentez-vous ?`

We may replace '`you feel`', with part-of-speech tags '`PRP VB`', by another sequence with same tags (i.e. '`we want`'):

Divergent words are shown in bold. Note that the new sentences are not assured to be grammatical after replacing sequences with the same part-of-speech. We use word alignments to identify

src:	`What do` **`we want`** `?`
$\mathcal{Y}^{src}$:	`-1 -1` **`+1 +1`** `-1`
tgt:	`Que` **`voulez-vous`** `?`
$\mathcal{Y}^{tgt}$:	`-1` **`+1`** `-1`

as divergent the sequence '`ressentez-vous`' since it is aligned to the replaced sequence '`you feel`'.

Finally, motivated by sentence segmentation errors observed in many corpora, we also build negative examples by **inserting** a second sentence at the beginning (or end) of the source (or target) sentence pair. Words in the original sentence pair are assigned the parallel label (-1) while the new words inserted are considered divergent ($+1$). Given the original sentence pair previously shown, we may want to build the next negative example by inserting the sentence '`Not .`' at the end of the original source sentence:

src:	`What do you want ?` **`Not .`**
$\mathcal{Y}^{src}$:	`-1 -1 -1 -1 -1` **`+1 +1`**
tgt:	`Que ressentez-vous ?`
$\mathcal{Y}^{tgt}$:	`-1 -1 -1`

In order to avoid that negative examples are easily predicted just by looking at the difference in length of training sentences we constraint all negative examples to have a difference in length not exceeding 2.0. Very short sentences, up to 4 words, are accepted if the length ratio does not exceeds 3.0.

The network outlined in Figure 2 uses a single label for each sentence pair. Hence, positive examples are labelled $\mathcal{Y} = -1$ while any of the previously described negative examples are labelled $\mathcal{Y} = +1$.

3.2 Fixing Translation Divergencies

We observed in our training corpus that many divergent sentences follow a common pattern, consisting of containing some extra leading/trailing words. See the first and second examples of table 1. Accordingly, we implemented an algorithm that discards sequences of leading/trailing words making use of alignment scores $S(i, j)$. Hence considering as parallel $s_{[}^{]}$ and $t_{\langle}^{\rangle}$. To find the optimal source ($[,]$) and target ($\langle, \rangle$) indexes that enclose the corrected segment within the original

sentence, we implement:

$$\arg\max_{[,],\langle,\rangle} \Big\{ \sum_{[\leq i \leq]} \max_{\langle \leq j \leq \rangle} \{S(i,j)\} \Big\}$$

The $\mathcal{N}$-best sequences following the previous function ($s_{[}^{]}$, $t_{\langle}^{\rangle}$) are considered as valid corrections, but only the highest ranked according to their similarity score is used as replacement for the original (s_1^I, t_1^J). Short sentences are not considered. This is, $] - [> \tau$ and $\langle - \rangle > \tau$.

Figure 3 (left) shows an example of an alignment matrix $s(i,j)$ for a given sentence pair. A possible correction is: `What do you feel ?` $\Leftrightarrow$ `Que ressentez-vous ?`. Hence, with indexes $[= 1,] = 5, \langle = 1$ and $\rangle = 3$.

4 Experiments

In this section we report on the experiments conducted to evaluate the proposed network. We begin with details of the corpora employed.

4.1 Corpora

Different corpora are used in this work to evaluate our similarity classifier.

We filter out divergencies found in the English-French OpenSubtitles corpus (**?**), which consists of a collection of movie and TV subtitles, aligned at the sentence level after a number of automatic preprocessing steps. The corpus presents many potential divergencies as outlined in Table 1. To evaluate performance we used the En-Fr Microsoft Spoken Language Translation task, created from actual conversations over Skype (**?**). In order to better assess the quality of our classifier when facing different types of word divergencies we collected from the original corpus and annotated at the word level 500 sentences containing: 200 paired sentences; 100 unpaired sentences; 100 sentences with replace examples; and 100 sentences with insert examples as detailed in Section 3.1.

We also use the Paracrawl corpus, which consists of a very noisy 1 billion word (English token count) German-English corpus crawled from the web as part of the Paracrawl project[3]. Performance is evaluated on the publicly available newstest-2017 En-De test set, corresponding to news stories selected from online sources (**?**).

To evaluate the ability of our sentence embeddings to identify parallel sentences in comparable corpora we followed the data conditions made available for the BUCC shared task (**?**). For training we used the French-English European Parliament corpus from WMT15[4]. For testing we used the test set made available by the task organisers, consisting of about $300,000$ monolingual sentences per language, French and English, that contains near $10,000$ parallel sentences. Test data come from Wikipedia[5] and News Commentary[6].

4.2 Training

All data used in this paper is preprocessed with an in-house toolkit that performs minimal tokenisation, basically splitting-off punctuation.

4.2.1 Neural Similarity Classifier

After tokenisation, each out-of-vocabulary word is mapped to a special UNK token. Our similarity classifier is described in Section 3. Word embeddings (LT_s and LT_t) are initialised using `fastText`[7], further aligned by means of `MUSE`[8] following the unsupervised method detailed in (**?**). Size of embeddings is $E_s = E_t = 256$ cells. Both Bi-LSTM use 256-dimensional hidden representations ($E = 512$). Optimization of the parameters is done using the stochastic gradient descent method along with gradient clipping (rescaling gradients whose norm exceeds a threshold) to avoid the exploding gradients problem (**?**). For each epoch we randomly select 1 million sentence pairs that we place in batches of 32 examples. Negative examples are created following the strategies detailed in 3.1.We run 10 epochs and start decaying at each epoch by 0.8 when score on validation set increases.

Two networks are evaluated. The first is the work presented in this paper, it optimises the loss shown by equation 3, and uses Equation 1 to compute similarity scores (`wemb`). The second implements the work in (**?**), it optimises the loss shown by Equation 5 which in an end-to-end fashion predicts wether two sentences are parallel (`semb`). To fix translation divergencies we used $\mathcal{N} = 20$ and $\tau = 3$.

[3] http://paracrawl.eu/

[4] http://www.statmt.org/wmt15/translation-task.html

[5] http://ftp.acc.umu.se/mirror/wikimedia.org/dumps

[6] http://www.casmacat.eu/corpus/news-commentary.html

[7] https://github.com/facebookresearch/fastText

[8] https://github.com/facebookresearch/MUSE

4.2.2 Neural MT

In addition to the basic tokenisation we perform Byte-Pair Encoding (**?**) with $30,000$ merge operations learned from both English and French data.

We build our NMT systems based on the open-source project `OpenNMT`[9]. We use a bidirectional RNN encoder with 4 LSTM layers with each containing $1,000$ cells. Word embeddings have a size of 300 cells. We set the dropout probability to 0.3. Batch size is set to 64. The maximum length of both source and target sentences is set to 80. The default optimiser is SGD with the starting learning rate of 1.0. We start to decay the learning rate from epoch 10 or when we detect increasing perplexity as compared to the previous epoch on a validation set. We stop training after 20 epochs.

5 Results

We evaluate first the ability of our similarity classifier to predict different types of divergencies. We use the test set manually annotated for that purpose and train our model on the OpenSubtitles corpus. The same number of paired examples (P) is used than for any other class (U, R or I).

Accuracy		Test examples				
		P	U	R	I	PURI
Train examples	PU	**0.996**	**0.994**	0.671	0.673	0.874
	PR	**0.995**	0.033	**0.951**	0.689	0.746
	PI	**0.998**	0.071	0.697	**0.725**	0.705
	PUR	**0.994**	**0.989**	**0.919**	0.710	0.932
	PUI	**0.995**	**0.996**	0.662	**0.769**	0.887
	PRI	**0.991**	0.161	**0.924**	**0.719**	0.768
	PURI	**0.995**	**0.980**	**0.916**	**0.788**	**0.942**

Table 2: Word divergence accuracies for networks trained with different combinations of negative examples. P, U, R and I stand respectively for pair, unpair, replace and insert.

Table 2 shows accuracies obtained by our model when trained over different combinations of negative examples on the task of predicting wether words are divergent. We consider divergent those words for which the similarity score predicted by our model, Equation 1, is negative.

As it can be seen, non-divergent words in parallel sentences (column `P`) are easy to identify. Divergent words in unpaired sentences (column `U`) are also easy to identify as far as the model has seen unpaired examples in training. The accuracy drops dramatically when the model is not used to see unpaired sentences (rows `PR`, `PI` and `PRI`).

[9] http://opennmt.net

Regarding columns `R` and `I`, accuracies are lower since these sentences contain a mix of divergent and non-divergent words. Hence, word divergence is more difficult to predict. Again, models that were trained with the corresponding examples obtain the highest accuracies (outlined in bold letters). Column `PURI` shows accuracy results over the entire test set. This is, mixing all type of examples. The best accuracy is obtained by the system trained on all type of negative examples.

Figure 3 shows an example of word and sentence similarity predictions when our network is trained using all types of negative examples (left); and trained using only paired and unpaired sentences (right). As it can be seen, the network trained using examples where all words are either divergent or non-divergent (right), fails to correctly predict the class for words in sentences with mixed divergent/non-divergent words. In contrast, the model trained with all types of examples (left) correctly predicts the right class for each word. Furthermore, it correctly assigns a lower similarity score to the sentence pair, as both sentences do not convey the exact same meaning.

5.1 Divergent Sentences Identification Task

Next, we evaluate the ability of our neural similarity classifier to detect divergent sentence pairs. Since we lack of a gold standard test set to directly evaluate sentence similarity, we measure the accuracy of neural MT systems trained over the original corpus and after filtering divergent sentences. Notice that some divergent sentences may not be completely useless to train a neural MT system. Consider for instance the example of Figure 3. Despite not conveying the exact same meaning it still contains useful information when training a neural MT engine. It is not the case of the example in the third row of Table 1.

$sim(h_{src}, h_{tgt})$	**Size (M)**	**Test (BLEU)**
$(-\infty, +\infty)$	27.2	42.18
$[0.000, +\infty)$	24.0	42.68
$[0.003, +\infty)$	21.5	42.56
$[0.076, +\infty)$	18.0	**43.19**
$[0.100, +\infty)$	15.5	

Table 3: BLEU scores obtained by a neural MT network when trained over the OpenSubtitles corpus filtered using different similarity thresholds.

Table 3 shows BLEU (**?**) scores of the same neural MT network when trained over different samples of the OpenSubtitles corpus. Samples result from filtering sentences of the original corpus

using different similarity thresholds. As it can be seen, the best performing training sample is obtained when using sentence pairs with similarity score in the range $[0.076, +\infty)$. This is, training with the most similar 18 million sentences of the original data set.

For each of the corpora, OpenSubtitles and Paracrawl, Table 4 shows BLEU scores of the neural MT system trained over the entire data set (`All`), and over the 18 (and 15) million sentences with highest similarity score as predicted by `semb` and `wemb` classifiers. REPHRASE

Data	Size (M)	Test (BLEU)
OpenSubtitles English-French		
`All`	27.2	42.18
`semb`	18.0	
`wemb`	18.0	43.19
Paracrawl English-German		
`All`	100	12.56
`base`	22.0	19.27
`semb`	15.0	
`wemb`	15.0	

Table 4: .

In the case of the Paracrawl corpus, we applied first a `base` filtering that reduced the amount of sentences to 22 millions. It consists of limiting the sentence size to 100 words. We also limit the length ratio between source/target sentences to 6, and we employed an in-house language Id classifier that helped us to detect sentence pairs not being English-German.

As it can be seen FINISH

5.2 Parallel Sentences Identification Task

Model	$N = 1$	10	50	100	500	$1,000$
`MUSE`						
`semb`						
`wemb`	ssaling					

Table 5: Number of parallel sentences found on the N-best cosine similarity filter (FAISS) according to the sentence embeddings produced by different networks.

The Cartesian product is costly for semb a filtering preprocess is necessary. Not for our model since we use faiss.

Model	λ	Precision	Recall	F-measure
`MUSE`				
`wemb`				
`semb`				

Table 6: Precission, Recall and F-measure for the three evaluated models on the BUCC task.

6 Conclusions

We presented an unsupervised method based on deep neural networks for detecting translation divergencies in parallel sentence pairs. Our model also predicts misaligned words that can then be filtered out allowing for reusing some divergent sentences. We evaluated our model on a neural machine translation task showing that it outperforms a baseline system trained on the entire (unfiltered) data set. The method can be used on any parallel corpus without any manual annotation.

7 Further Work

We plan to further evaluate divergence classification under larger data size conditions and different noise levels and on additional language pairs. We also plan to study the use of a neural network to predict divergent words, as a replacement for the algorithm detailed in Section 3.2.

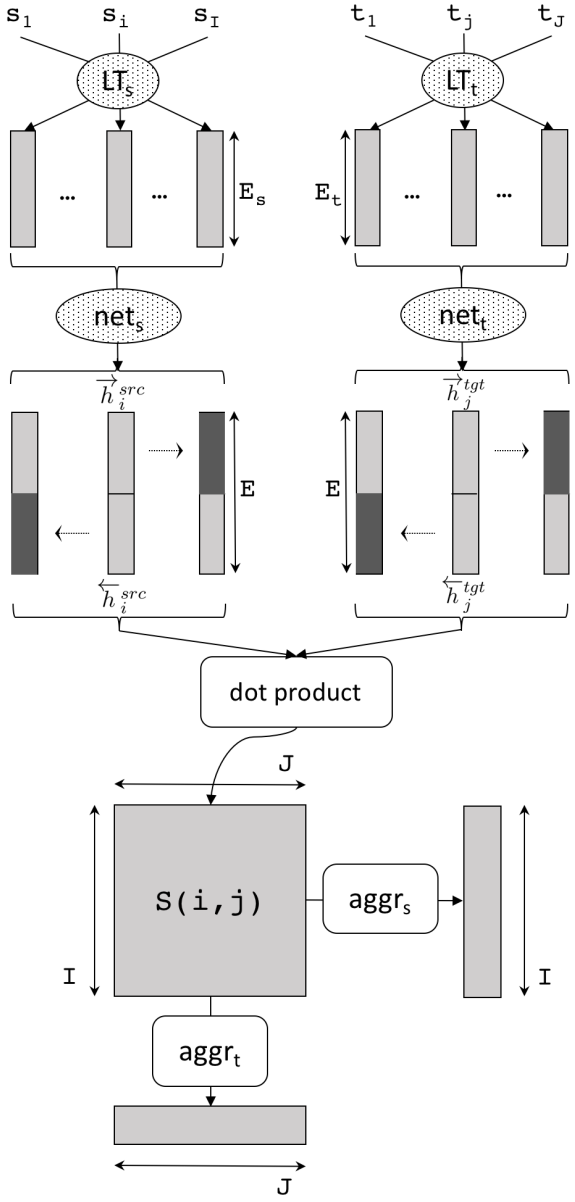

Figure 1: Illustration of the model. The network is composed of source and target word embedding lookup tables (LT_s and LT_t) and two identical subnetworks (net_s and net_t) that compute in context representations of source (s_i) and target words (t_j).

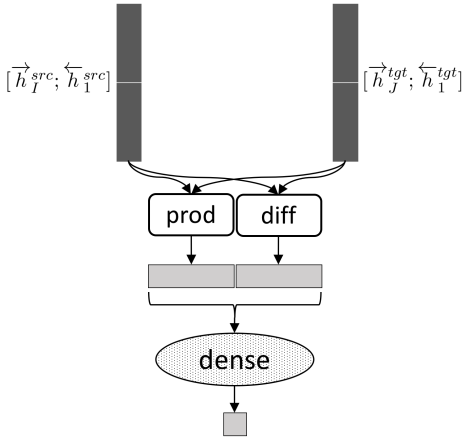

Figure 2: Illustration of the model presented in (**?**). Sentence embeddings (vectors in dark grey) are computed by the same Bi-LSTM layers shown in Figure 1.

Figure 3: Example of neural alignment matrices and divergence scores obtained with our model when trained over pair and unpair examples (right), and over all types of examples (left) as detailed in Section 3.1. Aggregation scores of Equation 2 are shown next to source and target words. Matrices contain word alignment scores. Sentence similarity scores computed by Equation 1 are shown below each matrix.