THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Neural Interactive Translation Prediction

Citation for published version:
Knowles, R & Koehn, P 2016, Neural Interactive Translation Prediction. in *Proceedings of AMTA 2016, vol. 1: MT Researchers' Track.* Association for Machine Translation in the Americas, AMTA, Austin, Texas, USA, pp. 107-120, Twelfth Conference of The Association for Machine Translation in the Americas, Austin, Texas, United States, 28/10/16. <https://amtaweb.org/wp-content/uploads/2016/10/AMTA2016_Research_Proceedings_v7.pdf#page=113>

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of AMTA 2016, vol. 1: MT Researchers' Track

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 27. Aug. 2022

Neural Interactive Translation Prediction

Rebecca Knowles rknowles@jhu.edu
Philipp Koehn phi@jhu.edu
Department of Computer Science, Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD, 21218, USA

Abstract

We present an interactive translation prediction method based on neural machine translation. Even with the same translation quality of the underlying machine translation systems, the neural prediction method yields much higher word prediction accuracy (61.6% vs. 43.3%) than the traditional method based on search graphs, mainly due to better recovery from errors. We also develop efficient means to enable practical deployment.

Interactive translation prediction (also called interactive machine translation) is an editing mode for translators who interact with machine translation output. In this mode, the machine translation system makes suggestions for how to complete the translation ("auto-complete"), and the translator either accepts suggested words or writes in their own translation. When the suggestion is rejected, the machine translation system recomputes its prediction for how to complete the sentence from the given prefix and presents the corrected version to the translator.

In prior work, phrase-based machine translation systems have been used for interactive translation prediction, and suggestions were made either by re-decoding constrained by the prefix (Green et al., 2014) or by searching for the prefix in the original search graph (Och et al., 2003; Barrachina et al., 2009). Recently, neural translation models have been proposed and in some cases have shown superior performance over phrase-based models (Jean et al., 2015; Sennrich et al., 2016). We propose to use such models for interactive translation prediction. Parallel to this work, Wuebker et al. (2016) also explore a similar approach to using neural MT for interactive translation prediction.

The decoding mechanism for neural models provides a natural way of doing interactive translation prediction. We show that neural translation models can provide better translation prediction quality and improved recovery from rejected suggestions. We also develop efficient methods that enable neural models to meet the speed requirements of live interactive translation prediction systems.

1 Interactive Translation Prediction

Interactive translation prediction leaves the translator in charge of writing the translation and places the machine translation system in an assisting role. Rather than having a translator post-edit machine translated output, the system actively makes suggestions as the translator writes their translation. This modality is similar to an auto-complete function. Whenever the translator diverges from the suggestion (by typing a word that differs from the model's suggestion), the system recalculates (taking the translator's input into account) and generates new suggestions. Implementations of interactive translation can be found in the CASMACAT[1] (see Figure 1) and

[1] http://www.casmacat.eu

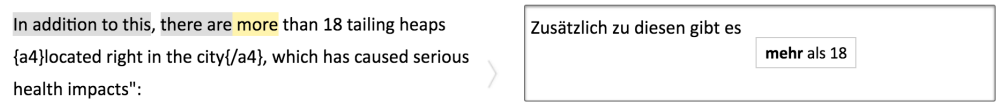


Figure 1: Interactive translation prediction in CASMACAT: The system suggests to continue the translation with the words *mehr als 18*, which the user can accept by pressing the TAB key.

Lilt[2] computer aided translation tools. This interaction mode is preferred by translators over post-editing (Koehn, 2009).

The goal of interactive translation prediction is to offer suggestions that the translator will accept. Existing approaches with statistical machine translation use the static search graph produced by the machine translation system. The system attempts to match the partial translator input (called the *prefix*) to the search graph, using approximate string matching techniques (minimal string edit distance) when an exact match cannot be found. As a baseline, we use a statistical machine translation system for interactive translation prediction that closely follows Koehn (2009) and Koehn et al. (2014). The prefix could be matched by constraint decoding, however, at a much higher computational cost.

Initial work on interactive translation prediction can be found in the TransType and TransType2 projects (Langlais et al., 2000; Foster et al., 2002; Bender et al., 2005; Barrachina et al., 2009). Our current work focuses on how to produce suggestions; for various approaches to interaction modalities, see Sanchis-Trilles et al. (2008) (mouse actions), Alabau et al. (2011) (hand-writing) and Cubel et al. (2009) (speech).

2 Neural Machine Translation

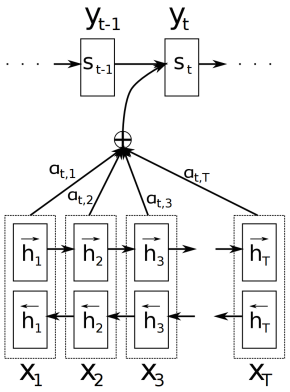
The use of neural network methods in machine translation has followed their recent success in computer vision and automatic speech recognition. Motivations for their use include better generalization of the statistical evidence (such as the use of word embeddings that have similar representations for related words), and more powerful non-linear inference.

The current state-of-the-art neural machine translation approach (Bahdanau et al., 2015) consists of:

- an **encoder** stage where the input sentence is processed by two recurrent neural networks, one running left-to-right, the other right-to-left, resulting in hidden states for each word that encode it with its left and right context,
- a **decoder** stage where the output sentence is produced left-to-right, by conditioning on previous output words in the form of a hidden state (roughly corresponding to a language model in traditional statistical machine translation) and on the input encoding (roughly corresponding to a translation model), and
- an **attention mechanism** that conditions the prediction of each output word on a distribution over input words (roughly corresponding to an alignment function).

We describe a fairly general neural machine translation approach in order to motivate its use in the interactive translation prediction setting, illustrated in Figure 2. For more details, see Bahdanau et al. (2015), whose notation this section follows, or Edinburgh's WMT 2016 submission (Sennrich et al., 2016), whose system we use in our experiments.

[2] https://lilt.com/



Target words y_t are emitted from hidden states s_t.

The hidden state s_t is informed by the input sequence, weighted by an attention mechanism $\alpha_{t,1}, ..., \alpha_{t,T}$.

The source language sequence $x_1, ..., x_T$ is encoded as the hidden states h of two recurrent neural networks.

Figure 2: Neural MT model used in this paper (figure from Bahdanau et al. (2015)).

At each time step t, the standard decoder computes the conditional probability of generating a word y_t given the input sentence $\vec{x}$. This is defined to be:

$$p(y_t|\{\hat{y}_1, \cdots, \hat{y}_{t-1}\}, \vec{x}) = g(\hat{y}_{t-1}, c_t, s_t) \tag{1}$$

where g is a non-linearity, $\hat{y}_{t-1}$ is the word produced by the previous decoding step, c_t is a context vector, and s_t is the hidden state for time t.

During encoding, annotations h_t were produced for each word x_t in the input sentence $\vec{x} = (x_1, \cdots, x_T)$. These h_t were produced by concatenating the forward and backward hidden states produced for each word by the forward and backward RNNs, respectively. The context vector c_t in Equation 1 is a weighted average of the annotations. First, weights $\alpha_{tj} = \exp(e_{tj}) / \sum_{k=1}^{T} \exp(e_{tk})$ are computed, where $e_{tj} = a(s_{t-1}, h_j)$ can be thought of as an alignment model (parameterized as a neural network and jointly trained with the rest of the system). The weight α_{tj} can be interpreted as the probability that y_t is aligned to x_j, resulting in soft alignments used by the system's attention mechanism to weight the focus of the context vector. The context vector is then computed as $c_t = \sum_{j=1}^{T} \alpha_{tj} h_j$.

As indicated above, decoding in this attention-based neural machine translation approach proceeds word by word. At each step of the decoding process, a probability distribution over possible next words is computed. This is conditioned on the previous word, the context vector, and the hidden state. The highest scoring word is selected and used in the conditioning context for the next step. Alternatively, similar to beam search in traditional statistical machine translation decoding, the top n next words may be considered and competing hypotheses with different output words maintained. Each of the hypotheses (consisting of a word sequence and a hidden state, and ranked by the combined word translation probabilities) is extended at the next decoding step.

There are various choices for the exact design of the recurrent neural networks used in the encoder and decoder. Going beyond basic approaches that use a simple hidden layer, more complex designs such as long short term memory (LSTM) cells or gated recurrent units (GRU) may be employed.

3 Neural Interactive Translation Prediction

The decoding process for neural translation models points to a straightforward implementation of interactive translation prediction. Instead of using the model's predictions in the conditioning context for the next step, the words in the prefix provided by the translator can be used. Hence,

the next word prediction is conditioned on the choice of the translator, rather than the prediction of the model.

During decoding for translation (as described above), the model's predictions $\{\hat{y}_1, \cdots, \hat{y}_{t-1}\}$ are fed back into the model to produce the next predicted word. In order to do interactive prediction, we instead feed the true prefix $\{y^*_1, \cdots, y^*_{t-1}\}$ produced by the translator back into the model. Thus we redefine the conditional probability of generating a word y_t to be:

$$p(y_t|\{y^*_1, \cdots, y^*_{t-1}\}, \vec{x}) = g(y^*_{t-1}, c_t, s_t) \tag{2}$$

In this work, we present two variations on neural interactive translation prediction:

- The **no beam search** method produces the single best hypothesis for each new word, given the prefix provided by the translator, which is fed into the model during decoding (as described above).

- The **beam search** method conducts beam search and selects the most probable full translation of the sentence. If and when the translator diverges from this full translation, a new beam search is conducted from the translator-generated prefix through to the end of the sentence. We show results for beam size 12, but note that a beam size of 2 provides most of the improvement (a similar observation was made by Sutskever et al. (2014) with respect to standard MT evaluation).

While beam search is known to produce better BLEU scores than models without beam search, it is also more computationally expensive. We demonstrate that it performs well on the interactive translation prediction task, but note that full beam search is too slow for use in a live system.

Passing the translator prefix into the system (when the translator diverges from the predicted sequence) may produce subsequent errors in the translation, for instance by causing the attention mechanism to be misfocused. We show that the system is often able to recover from these errors, but that it occasionally results in incoherent sequences of suggestions. However, we show that the sequences of rejected suggestions produced by the neural systems tend to be shorter than those produced by the traditional search graph based systems.

4 Experimental Setup

Since a user study that collects sufficient data for the various settings of our methods would be too costly, we resort to a simulation where a preexisting human translation is used in place of the translator's input to the interactive translation system. We do this by treating the preexisting human translation as though it is being typed live, one word (or letter) at a time, by a translator interacting with our prediction system. While we expect that in practical use, the human translator may match the machine translation's suggestions more closely (for example, by accepting synonyms which we score as "wrong" as they are not exact matches), we can nevertheless compare methods based on their prediction accuracy against the human translation relative to one another.

Data Our experiment is carried out on the German–English data sets[3] made available for the shared news translation task of the Conference for Machine Translation (WMT). The data consists of a 115 million word parallel corpus (Europarl, News Commentary, CommonCrawl),

[3] http://www.statmt.org/wmt16/

System	Configuration	BLEU
Neural	**no beam search**	**34.5**
	beam size 12	36.2
	+ ensemble	37.5
	+ r2l reranking	38.6
Phrase-based		**34.5**

Table 1: Quality measured by BLEU scores (case-sensitive) of the systems used in this paper on the WMT 2016 news test set (German-English).

and about 75 billion words of additional English monolingual data (LDC Gigaword, monolingual news, monolingual CommonCrawl). We use the official 2999 sentence test set (average sentence length 23 tokens) to measure the accuracy of our methods.

Neural Translation Model The neural machine translation model uses Nematus[4], a fork of the DL4MT toolkit[5] by Cho (2015). It was trained on all the available parallel data and a similar amount of synthetic parallel data that was generated by translating part of the monolingual news data into German (Sennrich et al., 2015a). It uses byte pair encoding (Sennrich et al., 2015b) for a vocabulary of 90,000 words. Training this model on GPU takes about three weeks. We use the publicly available model[6] that matches the training settings of Edinburgh's WMT submission (Sennrich et al., 2016).

Phrase-Based Model The phrase based model, which we use as a baseline to produce search graph based predictions, uses all available parallel and monolingual training data. The system matches Johns Hopkins University's submission to the WMT shared task (Ding et al., 2016). This version does not use the byte pair encodings used by the neural models.

System Quality Since we are concerned with the translation speed, we consider a few simplifications of the neural translation model. We do not use ensemble decoding ("ensemble") or a reranking stage ("r2l reranking")[7]. Each of these simplifications makes decoding several times faster at a cost to quality of 1-2 BLEU points. See Table 1 for a comparison of quality scores for the different settings. Without beam search, the neural system used has the same BLEU score as the phrase-based system. This has the nice advantage that we are comparing methods based on systems of similar quality. It also shows the potential for improvement if computational concerns are removed. For a longer discussion of the speed of the methods, see Section 6.

5 Results

See Figure 3 for an example of the neural interactive translation prediction model's output for one sentence. The figure displays the correct word choices (taken from the reference translation), the model's prediction (using the prefix of the reference translation as conditioning context), and the most probable word choices according to the model's probability distribution.

Some of the failures are near-synonyms (*numbers* instead of *levels*, or *increasing* instead of *rising*) that we might expect would be accepted by real human users of the system. For the purposes of our evaluation, we count even these near-synonyms as incorrect (as they are not exact string matches). However, it is worth noting the prevalence of these near-synonyms: in the neural versions, we find that 21.0% of incorrect predictions (22.4% with beam search) are

[4] `https://github.com/rsennrich/nematus/`

[5] `https://github.com/nyu-dl/dl4mt-tutorial/`

[6] `https://github.com/rsennrich/wmt16-scripts/`

[7] For more on these methods, please see Sennrich et al. (2016)

Input: *Das Unternehmen sagte, dass es in diesem Monat mit Bewerbungsgesprächen beginnen wird und die Mitarbeiterzahl von Oktober bis Dezember steigt.*

	Correct	Prediction	Prediction probability distribution
✓	the	the	**the (99.2)**
✓	company	company	**company (90.9)**, firm (7.6)
✓	said	said	**said (98.9)**
✓	it	it	**it (42.6)**, this (14.0), that (13.1), job (2.0), the (1.7), ...
✓	will	will	**will (77.5)**, is (4.5), started (2.5), 's (2.0), starts (1.8), ...
✓	start	start	**start (49.6)**, begin (46.7)
	inter@@	job	job (16.1), application (6.1), en@@ (5.2), out (4.8), ...
✘	viewing	state	state (32.4), related (5.8), **viewing (3.4)**, min@@ (2.0), ...
✘	applicants	talks	talks (61.6), interviews (6.4), discussions (6.2), ...
✓	this	this	**this (88.1)**, so (1.9), later (1.8), that (1.1)
✓	month	month	**month (99.4)**
✘	,	and	and (90.8), **, (7.7)**
✘	with	and	and (42.6), increasing (24.5), rising (6.3), **with (5.1)**, ...
✓	staff	staff	**staff (22.8)**, the (19.5), employees (6.3), employee (5.0), ...
✘	levels	numbers	numbers (69.0), **levels (3.3)**, increasing (3.2), ...
✘	rising	increasing	increasing (40.1), **rising (35.3)**, climbing (4.4), rise (3.4), ...
✓	from	from	**from (97.4)**
✓	October	October	**October (81.3)**, Oc@@ (12.8), oc@@ (2.9), Oct (1.2)
✘	through	to	to (73.2), **through (15.6)**, until (8.7)
✓	December	December	**December (85.6)**, Dec (8.0), to (5.1)
✓	.	.	**. (97.5)**

Figure 3: Example with about average prediction accuracy. Note the good recovery from failure and that several of the correct choices rank highly in the probability distribution of predicted words (values in parentheses indicate percent of probability mass assigned to words; only words with $\geq$1% shown). Tokens containing @@ are an artifact of byte pair encoding.

synonyms[8] of the correct answer (in the phrase-based system, this drops to 17.7%). Were these to be accepted by a real translator, overall system accuracy scores would improve.

The neural method copes well with failure, and typically resumes with plausible predictions. One exception is the prediction of *talks* after having seen ... *will start interviewing*. This may be due to the attention mechanism being thrown off after a sequence of low-probability prefix words.

5.1 Word Prediction Accuracy

Figure 3 also illustrates the evaluation metric we use to measure the quality of the prediction methods. It measures how many words are predicted correctly (see the first column in the figure). Note that we measure on the level of tokens, so we score the split word *inter@@ viewing* (an artifact of byte pair encoding) as a single token, rather than two tokens.

Table 2 shows the prediction accuracy for the three methods. The neural systems clearly outperform the method based on the search graph of the phrase-based model (with over 60% prediction accuracy for the neural systems and just 43.3% for the phrase-based). We discuss more reasons for this improvement in Section 5.3.

[8]Here we define words to be synonyms if their Wu-Palmer similarity (Wu and Palmer, 1994) in WordNet (Fellbaum, 1998) is equal to 1.

System	Configuration	Word Prediction Accuracy
Neural	no beam search	61.6%
	beam size 12	63.6%
Phrase-based	-	43.3%

Table 2: **Word prediction accuracy:** Ratio of words predicted by the interactive translation prediction system that matched the human reference translation exactly.

System	Configuration	Letter Prediction Accuracy
Neural	no beam search	86.8%
	beam size 12	87.4%
Phrase-based	-	72.8%

Table 3: **Letter prediction accuracy:** Ratio of letters predicted correctly.

5.2 Letter Prediction Accuracy

A useful feature of interactive translation prediction is the ability to react to single key strokes of the user. Consider an example from Figure 3. The system has a preference for *numbers* over *levels*. The latter is preferred by the human translator, hence the system fails to make the right prediction. If the user types the letter *l*, the system should quickly update its prediction to *levels*, the most likely word (from the probability distribution that generated the original hypothesis) that starts with this letter. In general, when the user types the initial letters of a word, the system should predict the most probable word with this letter prefix. In the beam search setting, the system first runs through the first word of each of the hypotheses in the beam (from most to least probable) to see if any match the the translator's letter prefix, before falling back to the probability distribution over the full vocabulary. With a beam size of 12, the correct word appears in the beam (but not as the predicted word) 25.2% of the time.

To measure the accuracy of system predictions for word completion, we count the number of incorrectly-predicted characters. To give a more complex example, suppose that the human translator wants to use the word *increased*. The system first predicts *rising*, but after seeing the letter *i*, it updates its prediction to *increasing*. It predicts all letters correctly until it comes to the final *e*. When the user enters *increase*, the system updates its prediction to *increased*. We count this as two wrongly predicted letters: **i***ncreas***e***d*. Table 3 shows the scores for our methods. Again, the neural methods clearly outperform the phrase-based method.

Note that this measure is not as clearly tied to user actions as word prediction accuracy. In the user interface shown in Figure 1, correctly predicted words are accepted by pressing TAB, while incorrectly predicted words have to typed in completely (assuming no word completion). So, the percentage of correctly predicted words reflects the ratio of words that do not have to be typed in. The effort savings for word completion are less clear, since there are various ways the user could interact with the system. In our example, when the user sees the prediction *increasing* but wants to produce *increase*, there are several choices even within the CASMACAT user interface. The user could accept the system's suggestion, and then delete the suffix *ing* and type in *ed*. Or, she could type in the entire prefix *increase* until the system makes the correct prediction — which in this example does not yield any savings at all: the user may accept the prediction with TAB or type in *d* herself.

System	Configuration	1	2	3	4	5
Neural	no beam search	55.9%	61.8%	61.3%	62.2%	61.1%
	beam size 12	58.0%	62.9%	62.8%	64.0%	61.5%
Phrase-based	-	28.6%	45.5%	46.9%	47.4%	48.4%

Table 4: Ratio of words correct after first failure.

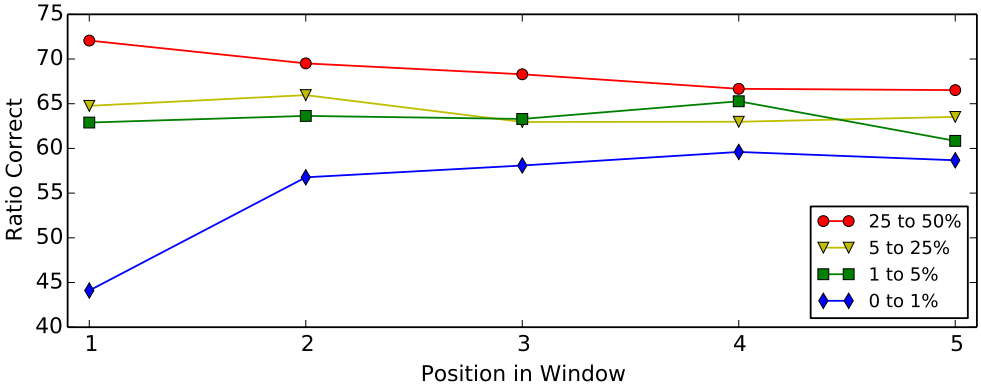


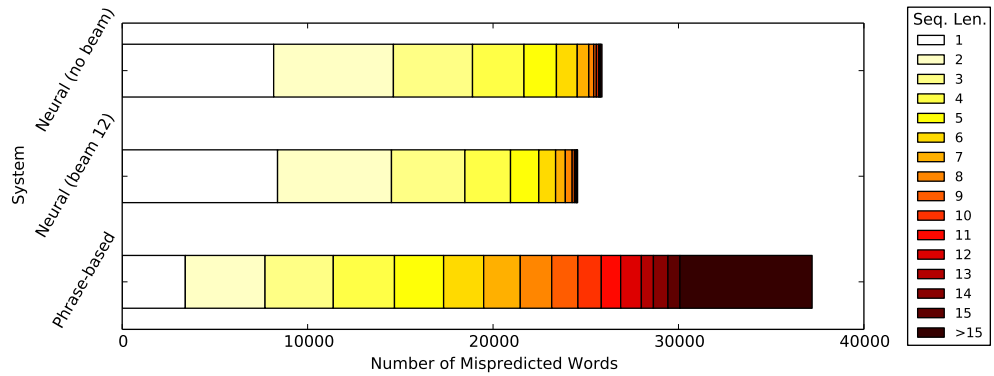
Figure 4: Neural (no beam) recovery from first failure at each position in the window of 5 words following the first failure, binned by probability assigned to correct solution (see legend).

5.3 Recovering from Failure

To get a more detailed picture of the performance of the neural prediction method, we explore how it recovers from failure. First, how well does the method predict the words following its first failure? We look at a window of up to five words following the first failure in a sentence (note that if the first failure is near the end of the sentence, the window will be truncated at the end of the sentence). See Table 4 for numbers for the methods.

The neural system predicts the first word in the window correctly 55.9% of the time, after it fails on a word. The second word in the window is predicted correctly 61.8% with similar numbers for the following words. So, failing on a word does impact the prediction of the word immediately following the failure, but only slightly words further down the sentence. The phrase-based method only correctly predicts 28.6% of the first words immediately after failing on a word. This suggests that the phrase-based method has a harder time recovering initially.

Interestingly, not all failures have an equal impact on the predictability of the subsequent words. Figure 4 shows the prediction accuracy for the neural method (without beam search) in more detail for the same five word window following the system's first error. We examine the way that the probability assigned to the correct word (which the model failed to predict) influences recovery from errors. When the model assigns extremely low probability (below 1%) to the correct answer, it performs very poorly on the next word, getting it correct only 44.1% of the time. On the other hand, when the model assigns relatively high probability to the correct word (25% to 50%), the probability of correctly guessing the next word rises to 72.1%. We can think of the probability assigned to the correct word as approximating how close the model was to being correct when it made the first error. When its prediction is far from correct, it has difficulty recovering, but when it is close to correct, it does not suffer a drop in performance in predicting the next words.



System	Config.	1	2	3	4	5	6	7	8	9	10+
Neural	no beam	8168	3229	1422	694	350	187	89	33	16	25
	beam 12	8378	3072	1320	615	305	151	75	46	14	15
Phrase	-	3403	2150	1227	825	530	360	282	212	157	774

Figure 5: The graph shows number of mispredicted *words*, categorized by lengths of the sequence of mispredicted words to which they belong. The table gives a breakdown of the number of *sequences* of each length.

We observe examples of this phenomenon (and its ties to near-synonyms) in Figure 3. When the model assigns low probability to the correct answer (e.g. *interviewing*), there are sequences of incorrect predictions. In the case of *rising*, the model predicts *increasing*, a near-synonym, and assigns the highest probability to *increasing*, *rising*, and *climbing* (in descending order). As we saw in Section 5.2, the neural system predicts more synonyms than the search graph system. We hypothesize that this is partly due to the neural system having additional information about the semantics of words (as represented by their embeddings), while the search graph system treats synonyms and non-synonyms alike.

5.4 Length of Sequences of Mispredicted Words

Another revealing set of statistics is the length of sequences of word prediction failures. If the method fails on one word, and predicts the next word correctly, we have a 1-word failure sequence. However, if it misses the next word also and only recovers after that, we have a 2-word failure sequence, and so on. Shorter failure sequences indicate better models (and a better user experience). Figure 5 visualizes the sequences of word prediction failures by showing how many mispredicted words can be accounted for by each failure sequence length (mispredictions in shorter sequences are represented by light colors while mispredictions in long sequences are shown in darker red).

The methods show a stark contrast. The neural methods have a much higher number of 1-word failure sequences (8168 and 8378 vs. 3403) and 2-word failure sequences (3229 and 3072 vs. 2150, comprised of 6458 and 6144 vs. 4300 mispredicted words) but comparably very few long failure sequences. For instance, only a small fraction of the neural systems' mispredicted words occur in sequences of greater than 15 errors in a row (93 or 16 words total), while 7129 of the phrase-based system's word prediction errors occur in misprediction sequences of length greater than 15 words. This is not simply a consequence of the greater word prediction accuracy of the neural systems; in particular, the phrased based model shows far more long misprediction sequences than one would expect were those errors distributed uniformly randomly (the neural systems also have more long misprediction sequences than would be expected if the errors

Length	1-4	5-9	10-14	15-19	20-24	25-29	30-34	35-39	100-104
CPU	108.6	115.7	122.7	127.0	131.3	136.1	140.7	145.2	184.4
GPU	7.0	7.2	7.4	7.4	7.4	7.4	7.6	7.6	7.6

Table 5: Decoding speed per word in milliseconds (neural model, no beam search) for different sentence lengths.

occurred randomly, but to a lesser extent).

These numbers suggest that the neural method recovers much better from failures, while the phrase-based system has more difficulty. Since the neural method considers every word in the vocabulary at any point in the sequence, it can always place the user's choice in a word prediction sequence, and does not have to resort to string edit distance to match up with the user's translation.

6 Speed Considerations

We have shown that the neural method delivers superior prediction accuracy, but is it fast enough? To be used in an interactive user interface, the method has to quickly produce alternative sentence completion predictions. A common time limit in human computer interaction is 100 milliseconds for the system's response. Any longer feels sluggish and annoying to the user.

6.1 Speed Measurements

In a basic setup, the neural machine translation decoder has to step through the user's prefix, and then produce predicted words until the end of the sentence. In other words, it has to translate the entire sentence for a response to a user interaction. Table 5 gives numbers for decoding speed, running on a multi-core CPU (32 core 3.20GHz Intel Xeon CPU E5-2667 v3, although only 2-3 cores are utilized on average) and a GPU (Tesla K80).

Decoding time is spent mostly on the matrix multiplications to compute hidden and output layer vectors. The computational cost of the argmax operation to pick the best word is negligible, hence the computational cost is essentially the same for matching words in the user prefix and predicting new words.

To predict a single word, the CPU requires over 100 milliseconds, which is clearly too slow. The time it takes to translate a single word slightly increases with the length of the sentence, since the attention mechanism has to sum over a larger context of source language words.

On a GPU, the cost to predict one word drops to 7 milliseconds. For a 20-word sentence, this means (7×20) 140 milliseconds which is also beyond our 100 millisecond time limit.

6.2 Fast Recovery

However, we can employ the following optimizations:

- We can precompute the initial translation when the document is uploaded. This gives us the entire prediction sequence.
- The user will diverge from the suggested translation at some point. We do not need to match the accepted user prefix again, since we can use the cached prediction sequence up to this point.
- We can produce a limited sequence of predicted words rather than a completely new full translation. In the user interface illustrated in Figure 1, only three words are shown. We may initially respond to the user interface with a small sequence of words and deliver the remaining words later.

- To have a full sentence completion at any time, we may initially patch together a limited prediction with the original sentence completion, since later words are less likely to be influenced by the local user choice.

(1) Initial hypothesis	*A sovereign prel@@ ate of the Champions League season .*
(2) Translator prefix	A confident
(3) New prediction (3 words)	`start to the`
(4) Alignment	`start to` **`the`** → *prel@@ ate of* ***the*** *Champions*
(5) New hypothesis	A confident `start to the` *Champions League season .*

Figure 6: Example of patching a 3-word prediction into the original sentence completion.

We propose a method (without beam search) that patches together a limited (say, 3 word) new prediction with the existing sentence completion each time that the translator diverges from the predicted translation. An example of this is shown in Figure 6 (we reference the row numbers parenthetically in the following description). We begin by precomputing a full translation when the document is uploaded (row 1). If and when the translator diverges from this (row 2), we compute predictions for the next 3 words only (row 3) and attempt to patch them together with the original translation.

We find the patch position by computing the KL divergence between the probability distribution that produced the last of the 3 new words and the stored probability distributions that produced the words in a 5-word window (following the position of the the last word in the translator prefix). This results in an alignment between the last of the 3 new words and the index of some word in the existing translation (row 4). The new translation hypothesis consists of concatenating the translator prefix, the 3 newly predicted words, and any words following the position of the index in the existing translation hypothesis that minimized the KL divergence (row 5).

By patching together earlier predictions with a short sequence of predictions based on new input from the translator, we can guarantee that we can serve the translator new predictions quickly. The new prediction and patching combined takes an average of 54.3 milliseconds to compute. This approach yields a word prediction accuracy of 56.4% and a letter prediction accuracy of 84.2% (a drop from the full search neural model by 5.2% and 2.4%, respectively, but still vastly outperforming the phrasal search graph system).

In a real-life setting, we may sometimes have enough time to recompute the full sentence in the background, rather than relying on patching together different predictions, so we could expect performance closer to the performance noted earlier. Additionally, we could use beam search (or other improvements to the neural model) in order to precompute better initial sequences, which we expect would also improve performance.

6.3 Analysis

In the example in Figure 6, the new hypothesis is in fact the correct translation. If the initial error by the system is a single-token error (for example a synonym), we might expect the last of the 3 newly translated words to align to the word at the center of the window. In this case it (correctly) aligns one position to the right of this and produces the desired hypothesis.

When the alignment is close to the center of the window, this suggests that the sentence does not require much reordering. The patching heuristic is somewhat imprecise and has difficulty handling sentences with long-range reordering. In Figure 7 we compute the failure

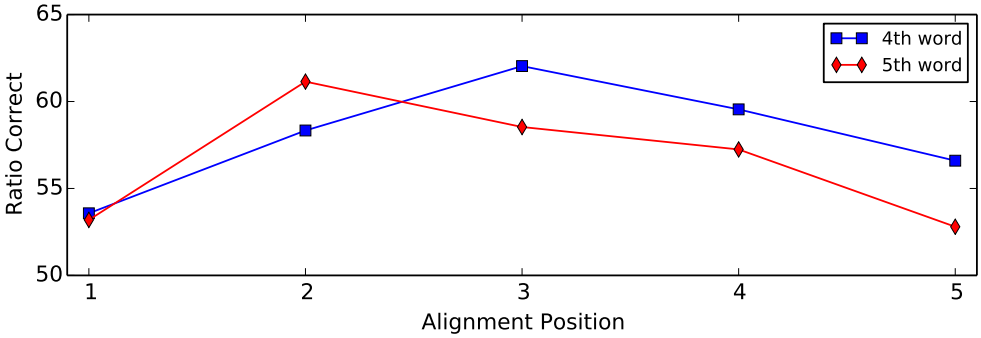


Figure 7: Ratio correct after first failure for the 4th and 5th words in the window.

recovery ratios for the 4th and 5th words in the window following the first error,[9] conditioned on the alignment position.

An alignment position of 3 indicates that the 3rd newly translated word aligned with the 3rd word in the window (as would be expected if no reordering were needed). Similarly, an alignment position of 1 indicates that the 3rd newly translated word aligned to the 1st word following the failure, and so on. We see that when a longer-distance alignment occurs, the ratio drops, demonstrating either an error of alignment or the system's difficulty in handling long-distance reordering.

7 Conclusion

In this paper we demonstrate that neural machine translation systems can effectively be applied to interactive translation prediction, improving upon the performance of traditional methods. We show that they recover well from errors, have shorter sequences of incorrect predictions, and produce more near-synonyms of the correct answers. Finally, we demonstrate a method that allows for practical deployment given real-life time constraints.

Acknowledgments

This work was supported, in part, by the Human Language Technology Center of Excellence (HLTCOE) through the 2016 SCALE workshop CADET, as well as by a National Science Foundation Graduate Research Fellowship under Grant No. DGE-1232825 (to the first author). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

Alabau, V., Sanchis, A., and Casacuberta, F. (2011). Improving on-line handwritten recognition using translation models in multimodal interactive machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 389–394, Portland, Oregon, USA. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Barrachina, S., Bender, O., Casacuberta, F., Civera, J., Cubel, E., Khadivi, S., Lagarda, A., Ney, H.,

[9] We show only performance for the 4th and 5th words in the window; performance on the first three is identical to the no-beam-search values reported in Table 4, as the patching occurs after this sequence of 3 new predictions.

Toms, J., Vidal, E., and Vilar, J.-M. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1).

Bender, O., Hasan, S., Vilar, D., Zens, R., and Ney, H. (2005). Comparison of generation strategies for interactive machine translation. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT)*, Budapest.

Cho, K. (2015). Neural machine translation tutorial. Technical report.

Cubel, E., Khadivi, S., Lagarda, A., Ney, H., Toms, J., Vidal, E., and Vilar, J.-M. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1).

Ding, S., Duh, K., Khayrallah, H., Koehn, P., and Post, M. (2016). The JHU machine translation systems for WMT 2016. In *Proceedings of the First Conference on Machine Translation (WMT)*.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Foster, G., Langlais, P., and Lapalme, G. (2002). User-friendly text prediction for translators. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 148–155, Philadelphia. Association for Computational Linguistics.

Green, S., Wang, S. I., Chuang, J., Heer, J., Schuster, S., and Manning, C. D. (2014). Human effort and machine learnability in computer aided translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1225–1236, Doha, Qatar. Association for Computational Linguistics.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Jean, S., Firat, O., Cho, K., Memisevic, R., and Bengio, Y. (2015). Montreal neural machine translation systems for wmt15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal. Association for Computational Linguistics.

Koehn, P. (2009). A process study of computer-aided translation. *Machine Translation*, 23(4):241–263.

Koehn, P., Tsoukala, C., and Saint-Amand, H. (2014). Refinements to interactive translation prediction based on search graphs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 574–578, Baltimore, Maryland. Association for Computational Linguistics.

Langlais, P., Foster, G., and Lapalme, G. (2000). Transtype: a computer-aided translation typing system. In *Proceedings of the ANLP-NAACL 2000 Workshop on Embedded Machine Translation Systems*.

Och, F. J., Zens, R., and Ney, H. (2003). Efficient search for interactive statistical machine translation. In *Proceedings of Meeting of the European Chapter of the Association of Computational Linguistics (EACL)*.

Sanchis-Trilles, G., Ortiz-Martínez, D., Civera, J., Casacuberta, F., Vidal, E., and Hoang, H. (2008). Improving interactive machine translation via mouse actions. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 485–494, Honolulu, Hawaii. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2015a). Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709.

Sennrich, R., Haddow, B., and Birch, A. (2015b). Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Sennrich, R., Haddow, B., and Birch, A. (2016). Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation (WMT)*.

Sutskever, I., Vinyals, O., and Le, Q. V. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, pages 133–138, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wuebker, J., Green, S., DeNero, J., Hasan, S., and Luong, M.-T. (2016). Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Berlin, Germany. Association for Computational Linguistics.