# A Comparison of Different Punctuation Prediction Approaches in a Translation Context

**Vincent Vandeghinste**
CCL – KU Leuven
vincent@ccl.kuleuven.be

**Lyan Verwimp**
ESAT-PSI – KU Leuven
lyan.verwimp@esat.kuleuven.be

**Joris Pelemans**
Apple Inc.
jpelemans@apple.com

**Patrick Wambacq**
ESAT-PSI – KU Leuven
patrick.wambacq@esat.kuleuven.be

## Abstract

We test a series of techniques to predict punctuation and its effect on machine translation (MT) quality. Several techniques for punctuation prediction are compared: language modeling techniques, such as $n$-grams and long short-term memories (LSTM), sequence labeling LSTMs (unidirectional and bidirectional), and monolingual phrase-based, hierarchical and neural MT. For actual translation, phrase-based, hierarchical and neural MT are investigated. We observe that for punctuation prediction, phrase-based statistical MT and neural MT reach similar results, and are best used as a preprocessing step which is followed by neural MT to perform the actual translation. Implicit punctuation insertion by a dedicated neural MT system, trained on unpunctuated source and punctuated target, yields similar results.

## 1 Introduction

In speech translation, the first step often consists of automatic speech recognition (ASR). Most ASR systems output an unsegmented stream of words, apart from some form of acoustic segmentation which splits a transcript into so-called *utterances*. Translating this stream of words, using off-the-shelf MT, results in a lower translation quality compared to translating punctuated input, as MT systems are usually trained on properly punctuated and segmented source and target text. End-to-end speech translation systems, that do not suffer from this problem, have recently achieved high-quality results too (Weiss et al., 2017), but these models require infrastructure (in terms of GPUs and training time) that is not available to everyone.

We compare several techniques and approaches for punctuation prediction in a translation context, starting from an input that already contains the correct sentence boundaries. All techniques and approaches are trained on the same dataset, allowing us to fully attribute different results to the specific techniques and approaches used. Thus, the main contribution of this paper is not introducing new methods for punctuation prediction, but a thorough comparison of methods previously used, since extensive comparisons are often lacking in related work. We compare three families of approaches for punctuation prediction: (1) language modeling, (2) sequence modeling, and (3) monolingual MT.

These approaches are combined in three different architectures resulting in translated and punctuated output: (1) *Preprocessing* adds punctuation before translating with a normal MT system, trained on punctuated source and punctuated target data; (2) *Implicit insertion* adds punctuation during MT, which is trained on unpunctuated source and punctuated target data; and (3) *Postprocessing* adds punctuation after MT, which is trained on unpunctuated source and unpunctuated target data. Figure 1 shows these different strategies, together with the *baseline* strategy, in which the unpunctuated data is translated by a regular MT system trained on punctuated source and target data.

## 2 Related work

In this section we discuss work that explicitly tries to predict punctuation marks like we do. We do not consider sentence boundary prediction.

Punctuation prediction is first described in



Pérez-Ortiz, Sánchez-Martínez, Esplà-Gomis, Popović, Rico, Martins, Van den Bogaert, Forcada (eds.)
*Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, p. 269–278
Alacant, Spain, May 2018.

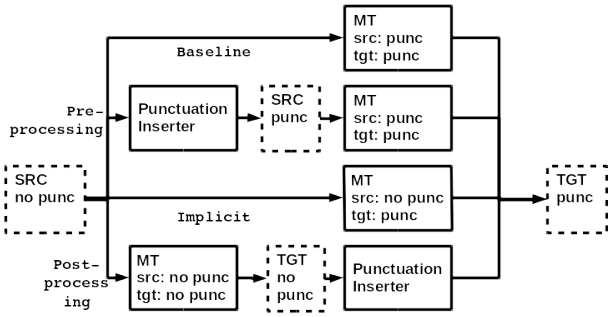


**Figure 1:** The different punctuation prediction strategies in a translation context.

(Beeferman et al., 1998), who use a lexical hidden Markov model to predict comma insertion in ASR output. Several other models have also been investigated, such as a decision tree classifiers (Kim and Woodland, 2001; Zhang et al., 2002), finite state models and multi-layer perceptrons (Christensen et al., 2001), a maximum entropy model (Huang and Zweig, 2002) and conditional random fields (Lu and Ng, 2010; Ueffing et al., 2013).

Gravano et al. (2009) use a purely text-based $n$-gram language model but do not compare with previously published methods. Several researchers use recurrent neural networks (RNNs) to tackle the problem as a sequence labeling task. Tilk and Alumäe (2015; 2016) use a two-stage LSTM (Hochreiter and Schmidhuber, 1997) to predict punctuation based on textual and prosodic features. Moró and Szaszák (2017) only use prosodic information to train a bidirectional LSTM while Gale and Parthasarathy (2017) compare several character-level convolutional and LSTM architectures, of which a simple LSTM with delay performs the best, although not consistently better than word-level bidirectional models. Pahuja et al. (2017) train a bidirectional RNN to jointly predict the correlated tasks of punctuation and capitalization.

As far as we know, only Tilk and Alumäe (2016) directly compare unidirectional and bidirectional word-level LSTMs for sequence labeling: even though their unidirectional model is smaller than their bidirectional model[1], the bidirectional one does not consistently outperform the unidirectional one. As we will see in section 4.1, we observe a similar trend.

[1]A hidden size of 100 (Tilk and Alumäe, 2015) vs. 256 (Tilk and Alumäe, 2016), while it is not clear whether both the forward and the backward have 256 units or whether each of them have 128.

In the context of MT, Matusov et al. (2006) and Peitz et al. (2011) present the three strategies for punctuation prediction we also use (as shown in figure 1). Lee and Roukos (2006) use a preprocessing approach, and Hassan et al. (2007) present a postprocessing apprach. Peitz et al. (2011) combine the outputs of the different strategies and find that "the translation-based punctuation prediction outperformed the LM based approach as well as implicit method in terms of BLEU and TER on the IWSLT 2011 SLT task". Combining outputs from different approaches through system combination yields even better results (Matusov et al., 2006b).

If we examine the comparisons with previously published methods in related work, we see that some do no compare their approach at all (Beeferman et al., 1998; Huang and Zweig, 2002; Hassan et al., 2007; Moró and Szaszák, 2017), others compare with either $n$-gram LMs (Kim and Woodland, 2001; Zhang et al., 2002; Lu and Ng, 2010; Peitz et al., 2011; Ueffing et al., 2013; Tilk and Alumäe, 2015; Tilk and Alumäe, 2016), CRF (Gale and Parthasarathy, 2017) or CRF and LSTM sequence labeling (Pahuja et al., 2017). We are not aware of a systematic comparison of MT approaches, $n$-gram LMs, LSTM LMs and LSTM sequence labeling. Especially a direct comparison of two of the most promising approaches, LSTM sequence labeling and monolingual MT, is lacking.

## 3 Methodology

We test several methods, keeping the data for training, tuning, and testing constant. Section 3.1 describes the data, section 3.2 discusses the models for punctuation prediction and section 3.3 the bilingual translation models. Finally section 3.4 explains how the quality of the punctuation prediction and translation is measured.

### 3.1 Data

As training data, we use the Dutch (source) and English (target) components of the Europarl corpus, version 7 (Koehn, 2005). The training data contains 55M words or 2M sentences (per language). As development set and test set we use the data of Vandeghinste et al. (2013). The development set consists of 574 sentences with one reference translation, randomly selected from actual translations made by a language service provider. As test set, we use 500 sentences with three reference translations, made by three different transla-

tors.[2]

All the data are tokenized, truecased, and depending on the experimental condition, cleaned with the Moses toolkit (Koehn et al., 2007). We compare the full dataset with a dataset in which all sentences longer than 80 words have been removed.[3]

We predict the following punctuation symbols: dot (`.`), comma (`,`), question mark (`?`), exclamation mark (`!`), colon (`:`), semicolon (`;`), opening and closing brackets (`(` `)`), slash (`/`) and dash (−). Note that our punctuation set is much larger than most previous work that we are aware of: for example Gale and Parthasarathy (2017) and Pahuja et al. (2017) only focus on predicting periods, commas and question marks.

### 3.2 Punctuation prediction

We apply punctuation prediction in the preprocessing as well as in the postprocessing punctuation strategy, i.e. in Dutch and in English.

#### 3.2.1 Punctuation prediction using language modeling

We train two types of LMs: $n$-gram models and LSTMs. The models for preprocessing are trained on Dutch and those for postprocessing on English. The $n$-gram models are 4-gram LMs (5-grams did not improve the performance) with interpolated modified Kneser-Ney smoothing (Chen and Goodman, 1999), trained with the SRILM toolkit (Stolcke, 2002). We compare the results for using the left context only (forward *fw*) with those for using both the left and the right context (forward + backward *fw*+*bw*), where both the preceding 3 words and the following 3 words can be used.

The LSTM LMs are trained with TensorFlow (Abadi et al., 2015) and consist of 1 layer of 512 cells, initialized randomly with a uniform distribution between -0.05 and 0.05. They are optimized with Adagrad (Duchi et al., 2011) with a learning rate of 0.1, early stopping is applied if the validation perplexity has not improved 3 times. Otherwise, the maximum number of epochs is 39. We train on batches of size 20 and unroll the network for maximum 35 time steps during backpropagation through time. With respect to regularization, the norms of the gradients are clipped at 5 and we apply 50% dropout (Srivastava et al., 2014) during training. We use sampled softmax (Jean et al., 2014) to speed up training. Due to a lack of resources, we did not apply an exhaustive hyperparameter optimization, but started from settings that have proven to work well for similar datasets.[4]

For punctuation prediction with LMs, we proceed as follows: we train the LMs on punctuated data and test on unpunctuated data. Given a non-punctuated input sentence, we determine the most probable token after every word. If a punctuation symbol is predicted, it is inserted at the current position in the input sentence and the updated sentence is used during the rest of the prediction. We continue the prediction until the end of the sentence is reached, including the position after the last token.

The full vocabulary of the training set consists of approximately 280k words for Dutch and 130k words for English (Dutch has much more compounding than English). Since models with that vocabulary size do not fit on our GPUs and since the large vocabulary also considerably slows down training of the LSTMs, we limit the vocabulary size to 50k. For fair comparison, we report results for $n$-grams models with the same vocabulary, but also for $n$-gram models with the full vocabulary in order to investigate the effect of the vocabulary size on the performance. All words not in the vocabulary are mapped to an *unknown-word*-class.

#### 3.2.2 Punctuation prediction using sequence labeling

Besides LSTM LMs, we investigate LSTM sequence labeling ('LSTM seq'): we train an LSTM that takes as input a word and the previous state and predicts in the output whether the word is followed by a punctuation symbol or not (⟨*nopunct*⟩-class). There are several advantages to this approach compared to language modeling: firstly, we train the LSTM on unpunctuated text and test it on unpunctuated text, so there is no mismatch in training and test conditions. Secondly, the models are directly optimized for punctuation prediction and they are easier to train since we do not have the large output weight matrix of an LM and we only

[2]These test sets are freely available upon request.

[3]For the development and test set, cleaning does not make any difference, as they are hand-made and are clean to begin with.

[4]We do not use bidirectional LSTM LMs for this task, since during training, the backward LSTM will have seen punctuation symbols following the current token and the model will learn to make use of those symbols. However, for applications such as speech translation, the input for the punctuation prediction model will have no punctuation at all, and hence the model that has learned to make use of subsequent symbols will not be optimal.

have to compute the softmax function over a small number of output classes. Finally, we can train bidirectional LSTMs without causing a mismatch between training input and testing input (see footnote 4). A disadvantage of these models is that the input is not punctuated, and hence the model cannot exploit punctuation in other parts of the sentence that is previously predicted. Note that, as opposed to Tilk and Alumäe (2016), we do not insert end-of-sentence symbols for the LSTM sequence labeling, because this would be an (unfair) advantage for bidirectional models – the probability of seeing an end-of-sentence punctuation mark right before an end-of-sentence symbol is naturally very high.

The hyperparameters of these models are the same as for the LSTM LMs, except that we use a full softmax in the output layer since we do not have to deal with a large vocabulary anymore. The bidirectional LSTMs consist of one forward LSTM of 256 cells and one backward LSTM of 256 cells, in total giving the same amount of LSTM cells and parameters as for the unidirectional LSTM (512).

#### 3.2.3 Punctuation prediction using machine translation

We can model the punctuation prediction as an MT problem, treating the non-punctuated version of our text as source language, and the punctuated version as target language: we build such monolingual MT systems for Dutch (preprocessing) and English (postprocessing).

The phrase-based statistical MT (*PBSMT*) condition uses the Moses decoder (Koehn et al., 2007) in its phrase-based mode, with a 5-gram LM, and *grow-diag-final-and* as phrase alignment criterion. For other parameters we use the default settings. The data is word-aligned using GIZA++ (Och and Ney, 2003). We do not allow reordering, setting the *distortion-limit* to 0. The *PBSMT clean* condition is equal to PBSMT, but removing all sentences longer than 80 words from the training data.

The *Hiero* condition uses Moses in hierarchical mode (Chiang, 2007), with a glue grammar and a maximum phrase length of 5. The other parameters are the same as for the PBSMT condition. All Moses systems are tuned using Minimum Error Rate Training (Och, 2003), maximizing on BLEU.

For the Neural MT (NMT) models, we use the OpenNMT framework (Klein et al., 2017), trained with the default settings, i.e. 500 LSTM cells, *seq2seq* model type, a vocabulary of 50k for both source and target language, a general global attention model (Luong et al., 2015), 13 epochs, a batch size of 64, and optimization through stochastic gradient descent (SGD). The initial learning rate is 1, except for the English model trained with SGD: since the training got stuck in a local minimum, we use 0.9 instead. The learning rate is decreased with a decay factor of 0.7, a beam size of 5, and replacements of *unknowns*, based on the highest attention weight.[5] We also try a variant with optimization Adam (Kingma and Ba, 2014) and a learning rate of 0.0002.

Variants of the systems trained with byte pair encoding have not been included in this study as initial tests only showed worse results than without byte pair encoding.

### 3.3 Translation Methods

We use the same MT systems as described in section 3.2.3, but now trained on the bilingual version of Europarl. Different from section 3.2.3 is that we now do allow phrase reordering for the phrase-based model, setting the *distortion limit* to 6.

### 3.4 Evaluation

We measure the quality of *punctuation prediction* with precision, recall and F1-score. The precision over all punctuation symbols is calculated as follows:

$$precision_{all} = \sum_{i \in P} \frac{TP_i}{TP_i + FP_i} \qquad (1)$$

with $P$ the class of all punctuation symbols, $TP_i$ the number of true positives for a certain punctuation symbol and $FP_i$ the number of false positives. Recall is calculated analogously. If a certain punctuation symbol has been predicted but the target is another punctuation symbol, we count this as a false negative.

Additionally, we use three common MT evaluation metrics, i.e. BLEU (Papineni et al., 2002), TER (Snover et al., 2006) and METEOR (Denkowski and Lavie, 2014) with synonyms, comparing the test set with predicted punctuation with the reference text (original text including punctuation). These metrics give us information on the quality of the entire output (and not only the punctuation prediction), which can be an

[5] Replacing the *unknowns* by their most probable aligned source language word.

issue in MT models that allow reordering, such as Hiero and NMT.

We measure the *translation quality* with the same three MT evaluation metrics. Note that, as described in section 3.1, we use an evaluation set with three references, ensuring a higher correlation of BLEU with human judgment, than when only one reference is used.[6]

We perform significance testing by bootstrap resampling for BLEU scores (Koehn, 2004) and F1 scores.[7]

## 4 Results

Section 4.1 describes the results of punctuation prediction and section 4.2 describes the results of MT of unpunctuated input.

### 4.1 Punctuation Prediction

#### 4.1.1 Dutch

Table 1 shows the results of the punctuation prediction experiments for Dutch. All MT approaches score significantly better on F1 and BLEU scores ($p < .001$) than the LM approaches. They also score significantly better on F1-score than the LSTM seq approaches ($p < .001$), but only *PBSMT*, *PBSMT clean* and *Hiero* score better on BLEU score ($p < .001$) than any of the LSTM seq approaches. *PBSMT* scores significantly better ($p < .05$) than the other MT approaches on BLEU, but on F1-score it scores only significantly better than *PBSMT clean* ($p < .001$). This difference between BLEU and F1 score can be explained by the fact that the non-PBSMT approaches can reorder the words and perform unwanted transformations other than inserting punctuation (mainly affecting BLEU). This is why we consider the PBSMT approach to punctuation insertion the best approach for this experimental setup.

Of the LM approaches, *n-gram fw+bw* scores significantly better than the other approaches on BLEU and F1 ($p < .001$). Increasing the vocabulary size has only a minor influence on the results: it decreases precision but increases recall, and has no significant effects on BLEU nor on F1. These

[6]The original BLEU paper (Papineni et al., 2002) also uses multiple references.

[7]We adapted the perl implementation by Mark Fishel for BLEU bootstrap resampling, which is available at https://github.com/moses-smt/mosesdecoder/blob/master/scripts/analysis/bootstrap-hypothesis-difference-significance.pl to work for F-scores on punctuation insertion.

**Table 1:** Results of punctuation prediction in Dutch

| Method | Prec. | Recall | F1 | BLEU | TER | MET. |
|---|---|---|---|---|---|---|
| $n$-gram LMs | | | | | | |
| fw 50k | 22.63 | 27.89 | 24.98 | 68.69 | 14.10 | 87.10 |
| fw full | 22.08 | 28.45 | 24.86 | 70.56 | 13.33 | 87.69 |
| fw+bw 50k | 54.49 | 78.59 | 64.36 | 79.63 | 8.32 | 92.88 |
| fw+bw full | 53.57 | 79.15 | 63.89 | 80.86 | 7.78 | 93.28 |
| LSTM LM fw | 44.75 | 31.83 | 37.20 | 83.90 | 7.97 | 92.42 |
| LSTM seq | | | | | | |
| fw | 72.03 | 11.97 | 20.53 | 86.11 | 8.42 | 91.12 |
| fw opt | 43.70 | 32.25 | 37.11 | 83.45 | 9.31 | 90.86 |
| fw+bw | 50.23 | 15.07 | 23.18 | 86.84 | 9.17 | 90.39 |
| fw+bw opt | 41.28 | 16.34 | 23.41 | 86.13 | 9.74 | 89.94 |
| PBSMT | 92.36 | 74.93 | **82.74** | **94.20** | **2.85** | **97.14** |
| clean | **93.88** | 71.27 | 81.02 | 93.76 | 3.06 | 96.88 |
| Hiero | 83.16 | **80.70** | 81.92 | 93.54 | 3.11 | 97.16 |
| NMT SGD | 84.53 | 79.30 | 81.83 | 85.71 | 6.88 | 91.79 |
| Adam | 82.43 | 79.30 | 80.83 | 85.04 | 7.12 | 91.61 |

**Table 2:** Results of punctuation prediction in English

| Method | Prec. | Recall | F1 | BLEU | TER | MET. |
|---|---|---|---|---|---|---|
| $n$-gram LM | | | | | | |
| fw 50k | 12.51 | 28.31 | 17.35 | 50.27 | 23.04 | 48.24 |
| fw full | 23.69 | 30.37 | 26.61 | 71.96 | 13.64 | 54.93 |
| fw+bw 50k | 42.62 | 73.60 | 53.96 | 69.21 | 10.41 | 53.30 |
| fw+bw full | 51.30 | 79.53 | 62.35 | 79.78 | 8.21 | 58.87 |
| LSTM fw | 35.23 | 25.10 | 29.31 | 80.76 | 10.37 | 57.21 |
| LSTM seq | | | | | | |
| fw | 69.88 | 7.50 | 13.54 | 90.19 | 8.86 | 59.75 |
| fw opt | 32.88 | 32.44 | 32.66 | 78.79 | 11.48 | 56.66 |
| fw+bw | 41.53 | 13.31 | 20.20 | 86.34 | 10.02 | 58.31 |
| fw+bw opt | 39.10 | 13.83 | 20.42 | 85.21 | 9.93 | 58.36 |
| PBSMT | 86.09 | 77.15 | 81.37 | **94.76** | **3.12** | **66.12** |
| clean | 83.46 | 76.77 | 79.77 | 93.77 | 3.45 | 65.36 |
| Hiero | 76.48 | 79.87 | 77.97 | 92.18 | 3.91 | 64.32 |
| NMT SGD | **91.41** | 82.59 | **86.76** | 93.53 | 3.44 | 64.97 |
| Adam | 90.62 | **82.63** | 86.43 | 93.78 | 3.35 | 65.19 |

models tend to overgenerate punctuation, which can be seen from their low precision.

LSTM sequence labeling (*LSTM seq*) does not score better than the LM approaches, mainly because of the low recall. The bidirectional LSTM has a lower precision but a slightly higher recall than the unidirectional LSTM. The *$n$-gram fw+bw 50k* and *$n$-gram fw-bw full* methods result in a significantly better F1 score ($p < .001$) than any of the *LSTM seq* methods. In BLEU scores, all *LSTM seq* methods are significantly better than all $n$-gram LM approaches. This reflects the fact that BLEU is a precision metric. Only the difference between *LSTM fw* and *LSTM seq fw opt* is not significant.

We tested two methods to improve recall for sequence labeling: thresholding for the probability distribution and weighted cross-entropy. Thresholding means that if ⟨*nopunct*⟩ is predicted but the ratio of the probability of the second most probable output over the probability of ⟨*nopunct*⟩ is higher than a certain threshold, we assign the second most probable token as prediction. This method indeed improves the recall of the model but lowers the precision: we report the result after optimizing the threshold for F1 score ('opt' in the table). We also

observe that optimizing for F1 does not result in better quality according to the MT metrics. The optimal threshold for the unidirectional model was 0.3 and for the bidirectional model 0.6. Trading off precision and recall had a much smaller effect on the bidirectional model than on the unidirectional one. Training with weighted cross-entropy, where more weight is given to the punctuation symbols since they are much less frequent than the ⟨*nopunct*⟩-class, has similar effects but has the disadvantage of having to re-train the model and optimize the weights per output class, while the threshold can be optimized during testing.

#### 4.1.2 English

Table 2 shows the results of punctuation prediction for English. As we had three reference sets in the original test set, we present averaged results over punctuation prediction on each of these three sets (we calculate the result for each set separately and average over the three datasets). For BLEU scores we used all three references.

All MT approaches score significantly better than the LM approaches ($p < .001$) They also score significantly better than the *LSTM seq* methods (at least $p < .005$). Similar to punctuation insertion for Dutch, *PBSMT* reaches the best BLEU scores, although not significantly better than *PBSMT clean*, but significantly better than *NMT SGD* and *NMT Adam* (both $p < .05$). With respect to the F1-score, we see that there is no significant difference between *NMT SGD* and *NMT Adam*, but *NMT SGD* scores significantly better ($p < .001$) than the other MT methods. *NMT Adam* scores better than *PBSMT* ($p < 0.05$) and *PBSMT clean* ($p < 0.001$ for two of the three test sets, not significant for the third one), and *Hiero* ($p < .001$).

For LM and sequence labeling, we see similar results as for Dutch, with the exception that limiting the vocabulary to only 50k words decreased the performance much more for English than for Dutch. This might seem surprising given that the Dutch dataset has a much larger vocabulary, but it has many more words that occur only once or a few times (ca. 200k types have a frequency of 5 or less in Dutch, as opposed to ca. 80k in English).

The $n$-gram LM approaches score much better on F1 score, but they overgenerate, as can be seen from the low precision and lower BLEU scores, when compared to LSTM seq approaches, which seem to undergenerate.

To conclude, we observe that for both Dutch and English the MT approaches work best for punctuation prediction as an isolated task. Since we are mainly interested in punctuation prediction in the context of speech translation, the phrase-based approach is the most promising since it does not cause any reordering of the words, giving the best results according to the MT metrics. We will now examine which approach achieves the best (bilingual) translation quality.

### 4.2 Translation of unpunctuated input

Table 3 shows the different experimental conditions that are evaluated and will be further explained in the next subsections. The best scores per punctuation strategy are marked in bold, the best scores per translation system are underlined.

#### 4.2.1 Baselines

In the baseline conditions, we train the MT systems on normal, punctuated, tokenized, and truecased source and target text, and tune them on the normal, punctuated, tokenized and truecased development set. We remove all the punctuation from the test set, and let the MT systems translate it. It hence constitutes the *lower bound*.

*NMT SGD* gets the highest BLEU score, but not significantly better than *PBSMT* and *PBSMT clean*. *Hiero* and *NMT Adam* score significantly worse than the other three conditions ($p < .001$).

#### 4.2.2 Upper Bounds

In the upper bounds conditions, we use the same MT systems as in the baselines, and evaluate them on the normal, punctuated, tokenized and truecased test set, to see how well the MT systems would do with "perfect" input.

Each of the upper bound scores is significantly better ($p < 0.001$) than the same approach in the baseline condition, so using MT without any form of punctuation insertion results in a significant loss in translation quality.

Comparing the different MT systems, *NMT SGD* is significantly better than *PBSMT* ($p < .01$), *PBSMT clean* and *NMT Adam* (both $p < .001$). There is no significant difference between *PBSMT*, *PBSMT clean*, and *NMT Adam*, but all score significantly better than *Hiero* ($p < .001$). Remarkable is the higher METEOR score for *PBSMT*.

**Table 3:** Results of punctuation insertion + translation.

| Punctuation Insertion Method | Translation System: PBSMT BLEU | PBSMT TER | PBSMT METEOR | PBSMT clean BLEU | PBSMT clean TER | PBSMT clean METEOR | Hiero BLEU | Hiero TER | Hiero METEOR | NMT SGD BLEU | NMT SGD TER | NMT SGD METEOR | Adam BLEU | Adam TER | Adam METEOR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Baselines** | 43.22 | 44.69 | 37.27 | 42.97 | 44.37 | 37.31 | 39.81 | 46.69 | 35.70 | **44.01** | **43.80** | **36.40** | 38.91 | 46.85 | 33.02 |
| **Upper bounds** | 46.19 | 39.15 | **39.08** | 46.55 | 38.80 | 39.01 | 42.72 | 41.59 | 37.35 | **49.13** | **37.66** | 38.26 | 46.30 | 39.41 | 37.59 |
| **Preprocessing** | | | | | | | | | | | | | | | |
| $n$-gram | | | | | | | | | | | | | | | |
| fw 50K | 36.64 | 48.53 | 35.36 | 35.36 | 48.11 | 34.93 | 35.68 | 49.35 | 34.73 | 36.00 | 52.76 | 31.50 | 34.33 | 53.05 | 31.01 |
| fw full | 37.63 | 47.36 | 36.62 | 37.46 | 46.98 | 36.77 | 36.19 | 48.51 | 35.55 | 38.76 | 50.99 | 33.45 | 36.60 | 51.34 | 32.74 |
| fw+bw 50K | 38.98 | 44.91 | 36.46 | 38.25 | 44.51 | 36.14 | 37.97 | 46.05 | 35.71 | 43.51 | 43.14 | 35.78 | 40.43 | 45.30 | 34.97 |
| fw+bw full | 40.41 | 43.46 | 37.81 | 40.08 | 43.42 | 37.83 | 38.33 | 45.36 | 36.40 | 44.36 | 42.01 | 36.68 | 42.54 | 43.18 | 36.45 |
| LSTM fw | 39.10 | 46.08 | 34.48 | 38.06 | 45.52 | 34.15 | 37.11 | 47.24 | 33.64 | 38.23 | 49.25 | 32.95 | 35.91 | 50.85 | 31.37 |
| LSTM seq | | | | | | | | | | | | | | | |
| fw | 42.75 | 44.25 | 37.22 | 42.17 | 44.02 | 37.15 | 39.56 | 46.25 | 35.76 | 43.49 | 43.54 | 36.19 | 39.43 | 46.55 | 33.43 |
| fw opt | 40.44 | 45.55 | 36.99 | 40.29 | 45.01 | 37.09 | 38.88 | 45.88 | 35.93 | 41.81 | 45.97 | 35.22 | 39.21 | 46.97 | 34.00 |
| fw+bw | 42.01 | 45.35 | 37.03 | 41.57 | 45.26 | 36.98 | 39.96 | 45.57 | 35.96 | 43.78 | 43.87 | 36.15 | 39.69 | 47.11 | 33.96 |
| fw+bw opt | 41.73 | 45.83 | 36.93 | 10.99 | 45.91 | 36.87 | 40.13 | 45.40 | 36.08 | 43.13 | 44.94 | 35.95 | 39.69 | 47.11 | 33.96 |
| PBSMT | 45.61 | 40.26 | **38.59** | 45.27 | 40.26 | 38.56 | 42.11 | 42.60 | 37.01 | **47.37** | **39.18** | 37.55 | 45.49 | 40.79 | 37.10 |
| clean | 45.54 | 39.98 | 38.51 | 45.19 | 40.27 | 38.58 | 41.87 | 42.70 | 36.95 | 47.04 | 39.40 | 37.44 | 45.29 | 41.05 | 36.97 |
| Hiero | 44.94 | 40.44 | 38.57 | 42.97 | 44.37 | 37.32 | 39.81 | 46.69 | 35.70 | 46.41 | 39.72 | 37.12 | 45.39 | 40.96 | 37.08 |
| NMT SGD | 44.99 | 40.49 | 37.49 | 44.81 | 40.56 | 37.64 | 40.97 | 42.90 | 35.83 | 47.17 | 39.35 | 37.22 | 45.19 | 40.77 | 37.04 |
| Adam | 44.65 | 40.77 | 37.40 | 44.53 | 40.35 | 37.60 | 40.94 | 43.14 | 35.84 | 47.05 | 39.77 | 37.34 | 45.23 | 40.95 | 37.13 |
| **Implicit** | 44.47 | 41.65 | **38.11** | 37.37 | 44.68 | 34.77 | 41.89 | 42.37 | 36.78 | **47.12** | **38.99** | 37.46 | 44.78 | 41.24 | 36.56 |
| **Unpunctuated** | 44.81 | 42.00 | 38.55 | 44.26 | 41.58 | **38.62** | 40.27 | 45.20 | 36.77 | **46.86** | **41.08** | 37.55 | 43.71 | 43.57 | 36.52 |
| **Postprocessing** | | | | | | | | | | | | | | | |
| $n$-gram | | | | | | | | | | | | | | | |
| fw 50K | 29.62 | 55.08 | 35.28 | 29.05 | 55.35 | 35.34 | 26.75 | 58.08 | 33.84 | 31.36 | 53.62 | 34.09 | 29.23 | 55.68 | 33.33 |
| fw full | 36.77 | 48.15 | 36.47 | 36.25 | 47.71 | 36.45 | 33.71 | 50.69 | 35.02 | 38.29 | 47.02 | 35.22 | 35.35 | 49.90 | 34.32 |
| fw+bw 50K | 36.59 | 46.18 | 37.08 | 30.06 | 45.80 | 37.26 | 33.19 | 49.13 | 35.54 | 39.17 | 45.06 | 35.97 | 36.19 | 47.54 | 35.05 |
| fw+bw full | 38.54 | 44.62 | 37.34 | 38.11 | 44.74 | 37.52 | 34.86 | 47.72 | 35.82 | 41.53 | 43.47 | 36.37 | 38.07 | 46.33 | 35.40 |
| LSTM fw | 39.67 | 46.51 | 36.74 | 39.04 | 46.15 | 36.80 | 35.81 | 49.42 | 35.10 | 41.78 | 45.53 | 35.70 | 39.20 | 47.73 | 34.92 |
| LSTM seq | | | | | | | | | | | | | | | |
| fw | 41.71 | 44.90 | 36.76 | 41.00 | 44.43 | 36.72 | 37.57 | 47.64 | 35.09 | 42.69 | 44.72 | 35.68 | 39.86 | 46.69 | 34.70 |
| fw opt | 40.66 | 46.44 | 36.43 | 37.68 | 46.77 | 36.57 | 34.46 | 50.24 | 34.98 | 39.62 | 46.52 | 35.40 | 37.17 | 48.84 | 34.48 |
| fw+bw | 40.66 | 46.44 | 36.43 | 39.88 | 46.29 | 36.40 | 36.93 | 49.14 | 34.81 | 41.86 | 45.95 | 35.34 | 39.12 | 47.98 | 34.34 |
| fw+bw opt | 40.51 | 46.67 | 36.41 | 39.66 | 46.54 | 36.38 | 36.71 | 49.52 | 34.81 | 41.70 | 46.09 | 35.30 | 39.01 | 48.07 | 34.32 |
| PBSMT | 45.10 | 40.94 | 38.31 | 44.75 | 40.41 | 38.44 | 41.11 | 43.62 | 36.63 | 46.73 | 40.21 | 37.34 | 43.58 | 42.44 | 36.33 |
| clean | 45.05 | 40.83 | 38.30 | 44.73 | 40.33 | 38.43 | 41.06 | 43.58 | 36.61 | 46.64 | 40.20 | 37.30 | 43.60 | 42.45 | 36.30 |
| Hiero | 44.41 | 41.61 | 38.25 | 43.85 | 41.11 | 38.39 | 40.38 | 44.27 | 36.59 | 46.04 | 40.73 | 37.21 | 43.14 | 43.08 | 36.26 |
| NMT SGD | 44.87 | 40.62 | 38.34 | 43.95 | 41.15 | 38.40 | 40.65 | 43.45 | 36.66 | **47.00** | **39.68** | **37.48** | 44.17 | 41.65 | 36.46 |
| Adam | 44.54 | 40.78 | 38.10 | 44.04 | 40.61 | 38.25 | 40.50 | 43.30 | 36.50 | 46.89 | **39.68** | 37.38 | 44.21 | 41.57 | 36.47 |

#### 4.2.3 Preprocessing

In the preprocessing conditions, we first insert punctuation, as described in section 4.1, before translating. The output of the punctuation insertion is then translated using a regular MT system, trained on punctuated data.

Using LM and LSTM seq as preprocessing approach never helps significantly over the baseline, only in the case of *ngram fw+bw full + NMT Adam* ($p < .001$). Using monolingual MT as preprocessing nearly always helps ($p < .005$), except when using *Hiero* as preprocessing or as translation engine. Whether *PBSMT* or *PBSMT clean* are used as preprocessors does not make a significant difference. When using *NMT SGD* as translation method, the kind of monolingual MT (apart from Hiero) does not play a significant role.

The best preprocessing results (using PBSMT as punctuation inserter) score still significantly lower than the upper bound scores when using the same translation system ($p < .05$ for *PBSMT*, *PBSMT clean* and *NMT Adam*, $p < .01$ for *Hiero* and $p < .001$ for *NMT SGD*).

#### 4.2.4 Implicit Punctuation Insertion

We remove punctuation from the source side of the parallel corpus and train the MT engines on these data, so they should be well-suited to translate source text without punctuation in target text with punctuation.

The score for implicit translation using *NMT SGD* is not significantly worse than preprocessing *PBSMT + NMT SGD*. *NMT SGD* scores significantly better ($p < .005$) than all other implicit punctuation insertion methods.

#### 4.2.5 Unpunctuated

We have tested the MT systems trained on unpunctuated data both in the source and the target, and evaluated against references from which the punctuation is also removed. As we use a different version of the references, we cannot apply significance testing. We present these results as they provide an indication about the maximum score we can expect for the postprocessing approach.

Even without punctuation inserted, it is clear that the scores are much lower than the *Upper bounds* presented earlier. The presence of punctuation thus improves the bilingual translation quality in general.

#### 4.2.6 Postprocessing

In the postprocessing approach, we translate using MT systems trained on unpunctuated data (both source and target), resulting in a translation that does not contain punctuation. The postprocessing step consists of punctuation insertion, similar to the preprocessing punctuation insertion step, but now for English.

Postprocessing with LM and LSTM seq does not yield any improvements over the baseline. With monolingual MT we reach significance in all cases where we use *PBSMT* ($p < .001$), *PBSMT clean* ($p < .001$) and *NMT SGD* ($p < .05$). *NMT Adam* also improves over the baseline ($p < .05$), except when combined with the *Hiero* system.

#### 4.2.7 General results

We note the lack of significant difference between pre- and postprocessing in the cases where punctuation insertion consists of *PBSMT*, *PBSMT clean*, *NMT SGD* or *NMT Adam* and translation consists of *PBSMT*, *PBSMT clean*, *NMT SGD* or *NMT Adam*.

When considering how much we can close the gap between *upper bound* and *baseline* using the best scoring combination of methods for each of the translation systems, we note gap closure of 80% for *PBSMT*, 64% for *PBSMT clean*, 79% for *Hiero*, 66% for *NMT SGD* and 89% for *NMT Adam*.

## 5 Conclusions and Future Work

We set out to compare different approaches to punctuation prediction in the context of translation. We test several different architectures and methods for punctuation prediction as well as for MT, all trained on the exact same data sets, and evaluate the punctuation prediction quality as a monolingual phenomenon, as well as its effect on MT quality.

While there is a clear deterioration of MT quality when working with unpunctuated input, this gap can be closed for 66% in the case of our best bilingual MT system, NMT, by applying monolingual MT as punctuation insertion, or by using a dedicated implicit insertion MT system.

Whether we use pre- or postprocessing did, in most cases, not result in a significant difference, indicating that the general punctuation prediction quality for Dutch is similar to that of English.

In future work, we would like to develop a similar experiment for *segmentation prediction*, and test the results on real speech signals in order to determine the usefulness of the results in a more realistic setting. A possible improvement would be to use NMT as punctuation prediction model, but constrain the word order with the help of the attention weights, thus combining the advantage of neural MT with the constraints on reordering of PBSMT.

## Acknowledgements

This research was done in the context of the SCATE project, funded by the Flemish Agency for Innovation and Entrepreneurship (IWT project 13007).

## References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow.org*.

Beeferman, Doug, Adam Berger and John Lafferty. 1998. Cyberpunc: A lightweight punctuation annotation system for speech. *IEEE Conference on Acoustics, Speech and Signal Processing.* 689–692.

Kim, Ji-Hwan and P.C. Woodland 2001. The use of prosody in a combined system for punctuation generation and speech recognition. *Proceedings of EuroSpeech.*

Christensen, Heidi, Yoshihiko Gotoh and Steve Renals. 2001. Punctuation Annotation using Statistical Prosody Models. *Proceedings of ISCA Workshop on Prosody in Speech Recognition and Understanding.*

Huang, Jing and Geoffrey Zweig. 2002. Maximum entropy model for punctuation annotation from speech. *Proceedings of ICSLP.*

Gravano, Agustín, Martin Jansche and Michiel Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. *Proceedings ICASSP.*

Chen, Stanley F. and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 17:359–394.

Chiang, David. 2007. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228.

Denkowski, Michael and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. *Proceedings of WMT.* 376–380.

Duchi, John, Elad Hazan and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Gale, William and Sarangarajan Parthasarathy. 2017. Experiments in Character-level Neural Network Models for Punctuation. *Proceedings Interspeech.* 2794–2798.

Hassan, Hany, Yanjun Ma and Andy Way. 2007. Matrex: the DCU machine translation system for IWSLT 2007. *Proceedings IWSLT.* 69–75.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation*, 9(8):1735–1780.

Jean, Sébastien, Kyunghyun Cho, Roland Memisevic and Yoshua Bengio. 2014. On Using Very Large Target Vocabulary for Neural Machine Translation. *arXiv preprint arxiv:1412.2007*.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv preprint arxiv:1701.02810*.

Kingma, Diederik P. and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arxiv:1412.6980*.

Koehn, Philip. 2004. Statistical Significance Tests for Machine Translation Evaluation. *Proceedings EMNLP.* 388–395.

Koehn, Philip. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. *Proceedings MT Summit X.* 79–86.

Koehn, Philip, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. *Proceedings of ACL Demonstration Sessions.* 177–180.

Lee, Young-suk and Salim Roukos 2006. IBM Spoken Language Translation System. *Proceedings TC-STAR Workshop on Speech-to-Speech Translation.* 13–18.

Lu, Wei and Hwee Tou Ng. 1998. Better punctuation prediction with dynamic conditional random fields. *Proceedings EMNLP.* 177–186.

Luong, Minh-Thang, Hieu Pham and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025.*

Matusov, Evgeny, Arne Mauser and Hermann Ney. 2006. Automatic sentence segmentation and punctuation prediction for spoken language translation. *Proceedings IWSLT.* 158–165.

Matusov, Evgeny, Nicolas Ueffing and Hermann Ney. 2006. Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. *Proceedings EACL.* 33–40.

Moró, Anna and György Szaszák. 2017. A phonological phrase sequence modelling approach for resource efficient and robust real-time punctuation recovery. *Proceedings Interspeech.* 558–562.

Och, Franz Josef and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Och, Franz Josef. 2003. Minimum Error Rate Training in Statistical Machine Translation. *Proceedings of ACL.* 160–167.

Pahuja, Vardaan, Anirban Laha, Shachar Mirkin, Vikas Raykar, Lili Kotlerman and Guy Lev. 2017. Joint Learning of Correlated Sequence Labeling Tasks Using Bidirectional Recurrent Neural Networks. *Proceedings Interspeech.* 548–552.

Papineni, Kishore, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings ACL.* 311–318.

Peitz, Stephan, Markus Freitag, Arne Mauser and Hermann Ney. 2011. Modeling Punctuation Prediction as Machine Translation. *Proceedings IWSLT.* 238–245.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. *Proceedings of AMTA.* 223–231.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky,Ilya Sutskever and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Stolcke, Andreas. 2002. SRILM an extensible language modeling toolkit. *Proceedings International Conference Spoken Language Processing.* 901–904.

Tilk, Ottokar and Tanel Alumäe. 2015. LSTM for Punctuation Restoration in Speech Transcripts. *Proceedings Interspeech.* 683–687.

Tilk, Ottokar and Tanel Alumäe. 2013. Bidirectional Recurrent Neural Network With Attention Mechanism for Punctuation Restoration Transcripts. *Proceedings Interspeech.* 3047–3051.

Ueffing, Nicola, Maximilian Bisani and Paul Vozila. 2013. Improved models for automatic punctuation prediction for spoken and written text. *Proceedings Interspeech.* 3097–3101.

Vandeghinste, Vincent, Scott Martens, Gideon Kotzé, Jörg Tiedemann, Joachim Van den Bogaert, Koen De Smet, Frank Van Eynde and Gertjan van Noord. 2013. Parse and Corpus-based Machine Translation. *Essential Speech and Language Technology for Dutch*, chapter 17, Peter Spyns and Jan Odijk (eds.), 305–319.

Weiss, Ron, Jan Chorowski, Navdeep Jaitly, Yonghui Wu and Zhifeng Chen. 2017. Sequence-to-Sequence Models Can Directly Translate Foreign Speech. *Proceedings Interspeech.* 2625–2629.

Zhang, Zhu, Michael Gamon, Simon Corston-Oliver and Eric Ringger. 2002. Intra-sentence punctuation insertion in natural language generation. *Microsoft Technical Report.*