A unified approach to sentence segmentation of punctuated text in many languages

Rachel Wicks[1] and **Matt Post**[1,2]
[1]Center for Language and Speech Processing
[2]Human Language Technology Center of Excellence
Johns Hopkins University
rewicks@jhu.edu, post@cs.jhu.edu

Abstract

The sentence is a fundamental unit of text processing. Yet sentences in the wild are commonly encountered not in isolation, but unsegmented within larger paragraphs and documents. Therefore, the first step in many NLP pipelines is *sentence segmentation*. Despite its importance, this step is the subject of relatively little research. There are no standard test sets or even methods for evaluation, leaving researchers and engineers without a clear footing for evaluating and selecting models for the task. Existing tools have relatively small language coverage, and efforts to extend them to other languages are often ad hoc.

We introduce a modern context-based modeling approach that provides a solution to the problem of segmenting punctuated text in many languages, and show how it can be trained on noisily-annotated data. We also establish a new 23-language multilingual evaluation set. Our approach exceeds high baselines set by existing methods on prior English corpora (WSJ and Brown corpora), and also performs well on average on our new evaluation set. We release our tool, ERSATZ, as open source.

	Examples of Ambiguity in Punctuated Contexts
en	... in the U.S. ⊗ House of Representatives ...
	...in the U.S. ✓ Most Mexican Spanish ...
cs	... podnikanie s.r.o. ⊗ a hlavním investorem
	... a Systémy s.r.o. ✓ V roce 2017 ...
ro	... W. Pauli ș.a. ⊗ constituie direcţii ...
	... de Robles ș.a. ✓ A jucat în ...

Table 1: Examples of ambiguous `FULL STOP` punctuation in English, Czech, and Romanian from Wikipedia. ✓ denotes a sentence boundary while ⊗ denotes an ambiguous sentence-internal position.

1 Introduction

In many ways, the sentence is the fundamental unit of text in natural language processing (NLP). From the user perspective, tasks such as sentiment analysis, POS tagging, or machine translation consume sentences and emit classifications, annotations, or transductions of those inputs. Even tasks that operate at the paragraph or document level, such as coreference resolution or summarization, often make use of sentences internally. Yet at the same time, sentences in the wild rarely exist with marked sentence boundaries. For many languages, punctuation serves as a cue for these boundaries, but this punctuation is ambiguous—as we might see with acronyms or abbreviations in English. When segmented sentences are required, they must be split using a *sentence segmentation* technique that can resolve these ambiguities. Despite its importance and early position in the NLP pipeline, sentence segmentation is the subject of relatively little research. Widely-used tools such as that in Moses (Koehn et al., 2007) are implemented with ad-hoc, manually-designed, language-specific rules, leaving them vulnerable to the long tail of languages and language phenomena. The little comparative work that does exist generally focuses on techniques that work in English or other Indo-European languages (Palmer and Hearst, 1997; Gillick, 2009).

Secondly, there is not a well-understood methodology for training segmenters that do not make narrow assumptions about the features or characteristics of the languages they support. At the heart of this is the lack of labeled training data. Manually-split datasets that accompany annotation projects tend to be small, and larger datasets are typically (imperfectly) segmented by the very tools whose performance is under question. Tools such as NLTK (Bird and Loper, 2004), which packages Punkt (Kiss and Strunk, 2006), provide an unsupervised method to train a model, but it is unclear what the effect is when switching to non-Latin-script languages, or how a more supervised approach would handle such noisy data.

Finally, and perhaps most importantly, there are no standard test sets or even metrics for evaluating segmenter performance, leaving researchers and engineers with no objective way to determine which one is best.

The work described in this paper is aimed at these problems. We propose a simple window-based model and semi-supervised training paradigm for the segmentation of punctuated text (§3). We frame the task as binary classification applied to a set of candidate punctuation locations defined by a regular expression. Leveraging the similarity of the task across languages (Table 1), we show that our model is able to successfully bootstrap from multilingual data that has been imperfectly segmented. We define a common metric that works across different tools (§4), and assemble a multilingual test suite by semi-automatically splitting existing (undersegmented) test sets (§5), providing a basis for proper comparison. We release these data splits along with our tool, ERSATZ, as open source.[1]

2 Background

A sentence is a sequence of grammatically linked words that conveys a complete thought. The term can be difficult to define in a precise manner that will not admit any exceptions, and in applications like machine translation, there are many times where the basic input unit is not a sentence, but a sentence fragment, such as a headline or an item from a list. In this work, we skirt these complexities, choosing instead to focus on the most common scenario, in which we are dealing with standard written language. For this, we adopt a functional definition: a sentence is a group of words that ends with a sentence-ending punctuation mark, such as (for many languages) a period, question mark, or exclamation point. Since punctuation is often used for non-sentence-ending purposes as well, the primary challenge for sentence segmentation is resolving this ambiguity for each segmentation candidate.

Research in sentence segmentation[2] has been limited in scope. Prior work either introduces methods that work under a set of assumptions unique to Latin-script languages (the existence and importance of casing, word length, or whitespace), or tackles new languages ad hoc, making adaptation to new languages and domains difficult.

Statistical methods use text-based features such as casing, punctuation, or length of surrounding words to make decisions around punctuation. The earliest work we found (Riley, 1989) considered all sentence boundaries and used decision trees based on these features. Gillick (2009) trained two statistical models in the form of an SVM and Naive Bayes classifier. Palmer and Hearst (1997) introduced `Satz` and shifted the approach by only focusing potential sentence boundaries being near sentence-ending punctuation, using part-of-speech distribution vectors as input to a feed-forward neural network and additionally applied their technique to German and French.

In order to work without labeled data, Kiss and Strunk (2006) used heuristics to craft scores based on likelihood values of occurrences of tokens, punctuation, casing and token length, and then manually tune a threshold of score to indicate a sentence boundary. This work expanded the most multilingually, considering 10 Indo-European languages as well as Estonian and Turkish.

Other work has focused on specific non-English languages. Xue and Yang (2011) study Chinese and dissect the theoretical reasons behind segmenting Chinese sentences to match their English equivalents. To segment Thai, which lacks punctuation, Zhou et al. (2016) use POS-taggers. Some work has tackled the problem of domains. Sanchez (2019) approaches the problem of legal text, which has a set structure without punctuation; other approaches (Wang et al., 2019; Rehbein et al., 2020) have investigated speech, which lacks both punctuation and written textual structure.

A popular splitter is packaged in the Moses toolkit (Koehn et al., 2007),[3] which works by splitting on all sentence-final punctuation unless the preceding context is a "non-breaking prefix"—a hand-built, language-specific list of acronyms and abbreviations. This approach cannot resolve the ambiguity where punctuation legitimately exists at the end of a sentence and is indifferent to novel

[1]`https://github.com/rewicks/ersatz` or `pip install ersatz`.

[2]Alternately called *sentence boundary detection*.

[3]We use the repackaged Python module at `https://pypi.org/project/sentence-splitter/`.

abbreviations at inference time. It produces a conservative segmenter that is high precision (unlikely to oversegment) but low recall (prone to undersegmenting). This raises the question of what effect reliance on this tool has had on construction of recent massive bitexts, such as CCMatrix (Schwenk et al., 2019b, §4.3). Gillick (2009) credit a 0.75% increase in accuracy to reduction of summarization error by a factor of four. Errors in segmentation may therefore affect the top matches for a sentence when doing bitext construction. Another popular splitter is SpaCy, which has not been described or evaluated anywhere, as far as we could tell.

With sentence splitting being a crucial piece of modern corpus creation for machine translation and other tasks, the lack of approaches and rigorous comparisons between tools limits the field. Additionally, the research field moving towards (often massively) multilingual settings, the need to build multilingual tools compare them in a proper scientific framework is both important and evident.

3 Approach

Our general approach is to treat sentence segmentation as a binary classification problem, predicting sentence-internal (⊗) or sentence-ending (✓) positions. The input to the model (§3.1), shown in Figure 1, is the concatenated left and right token contexts, as depicted in Table 1. Predictions for both training and inference are done only at predefined *candidate sites*, which are determined by a regular expression (§3.2). We then train in a semi-supervised setting where many of the labels may be missing (§3.3).

3.1 Models

Our basic model is depicted in Figure 1. The encoder is a two-layer Transformer (Vaswani et al., 2017). Our hyperparameter search incorporates vocabulary size (V), embedding size (e), and left and right context sizes (l and r). We also experiment with simpler architectures (§8.4), including single blocks of fully-connected linear layers with a TanH activation.[4] These simpler models typically traded increased throughput for slight degradations in F1. Our training objective is binary cross-entropy loss.

[4]We initially experimented with various functions and layers (Sigmoid, ReLU, Pooling layers, etc) but found that TanH performs best.

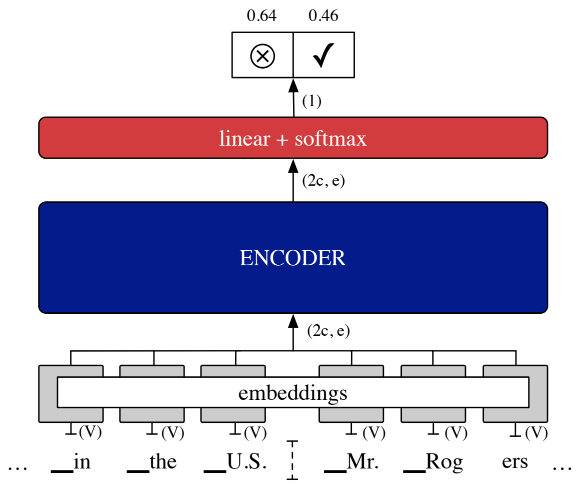

Figure 1: Model architecture. A binary predictor is constructed from token embeddings from the left and right context. Arrows denote output dimensions: V is the vocabulary, l and r the left and right context window sizes, and e the model/embedding size.

3.2 Candidate sites

Our model works with segmentation *candidate sites* for both training and inference. This can be done in a fairly general, language-agnostic way. Let P be the set of all punctuation, and $P_e \subset P$ be the set of sentence-ending punctuation. For a given input, we examine every character boundary and match based on two regular expressions for the left and right context, respectively:

- $(.*P_eP*)$: The left context ends with sentence-final punctuation, optionally followed by any amount of punctuation; and
- `([^0-9].*)` : The right context does not start with a number.

Raw text examples can be found in Table 1 and tokenized examples with fixed context sizes are shown in Table 2.

Input to the model is in the form of documents. A linear pass over the data identifies all candidates sites and assembles them into a batch, with their associated left and right contexts. At training time, instances are extracted with their labels: ⊗ for line-internal sites, and ✓ for sites that occur between input lines. At inference time, the trained classifier is applied, and newlines are inserted where ✓ is predicted.

This general definition carries benefits and risks. On the positive side, it allows us to work with many languages without having to develop language-

Label	Left context	Right context
⊗	_the _ P . K .	_ S h t
⊗	s o on er ? "	_ h e _
✓	B . A . T .	_ " W e
✓	n er s . ")	_ I _ st

Table 2: Candidate site examples with their labels. Left context-size (6) and right context-size (4) occurs after subword tokenization. In text, '_' is subword beginning-of-word character.

specific rules. It also speeds up training and inference, boosting both training speed and inference performance. On the downside, this loose definition can permit oversegmentation, since it permits, for example, word-internal segmentation in English and other languages. The criteria for identifying candidate sites can be easily altered to be more constrained or more general depending upon use case, and the list of punctuation to support more languages, if necessary. Our default list covers many languages.[5]

3.3 Training data

As noted in our motivation, sentences in the wild are often not segmented but are part of paragraphs and documents. It is therefore unsurprising to find many segmentation errors in existing corpora. A particular problem one can observe is that of undersegmentation, perhaps resulting from application of conservative segmentation tools. This means the raw training data may contain many false negatives (✓ sites mistakenly labeled as ⊗). Training a sentence segmentation model therefore presents a chicken-and-egg problem. We aim to train directly on existing data created for MT purposes, despite its having been either segmented by imperfect segmenters, or never segmented.

While some data is undersegmented, the vast majority of the end-of-line contexts should be correct, since they are either (a) natural existing boundaries at the end of a paragraph or document or (b) the result of applying a conservative segmenter. We therefore hope to train classifiers even despite this noise. Because we are considering a binary classification problem (and using the associated binary cross entropy loss), we additionally consider adding a weighted λ value to the ✓ class in order to give more credence to these contexts.[6]

For punctuation at the end of a line, the right-context is taken from the tokens at the beginning of the next sentence. In Section §7.3, we look into whether it matters if this right context is the true document context, or whether a random sentence will serve.

4 Evaluation: Metric

For evaluation, we begin by removing sentences that do not end in punctuation, since none of the tools are able to segment these. We then concatenate the test set into a single line, joining sentences with a space.

Evaluation among different tools contains subtle complexities. First, some tools normalize or tokenize the input text, complicating alignment between the input and the output. Second, different tools may attempt to segment at different subsets of input string locations, which might unfairly bias the results in favor of conservative tools. Finally, if we permit segmentation at *any* point of the input, there is a large class imbalance between ⊗ and ✓.

The class imbalance advocates for F1 as a natural metric. The use of F1 also addresses the second issue, since only the gold positive class (✓) factors into the score. The first two issues also require that we align a segmenter's output with the gold standard segmented text. Since the texts are largely similar, we can do this efficiently using a modified Levenshtein distance[7] that only considers a fixed maximum distance between any two characters. Once the text is aligned, we compute F1 against the set of ✓ symbols in the gold text. An example is depicted in Figure 2.

5 Evaluation: Data

We have noted the difficulty with making use of imperfect training data, and how we hope to work around it (§3.3). Unfortunately, this workaround cannot be used for evaluation, where we need gold-standard data.

We construct test sets from the WMT News Translation test sets (Barrault et al., 2020), which

[5]Our punctuation set (by unicode name): Full Stop, Question Mark, Exclamation Mark, Ellipsis, Ideographic Full Stop, Devanagari Danda, Arabic Question Mark, Arabic Full Stop, Khmer Sign Khan

[6]Generally, we find no weight ($\lambda = 1.0$) is sufficient in punctuated English, but increasing the weight ($\lambda = 20$) improved performance in some languages and the multilingual setting where the data is noisier.

[7]While the distance itself can also be considered in comparing tools, we do not report these distances, and instead use the technique to align text within the window.

		P	R	F1
text	… him. He added: “Mr. Rogers” …			
gold	**h i m . ✓ H e a d d e d : “ M r . R o g e r s**	–	–	–
sys1	h i m . ✓ H e a d d e d : " M r . ✓ R o g e r s	0.5	1.0	0.67
sys2	h i m . ✓ H e a d d e d : ✓ ' ' M r . ✓ R o g e r s	0.3	1.0	0.5

Figure 2: Input text formatted as gold-standard data with two system outputs. Gold positive labels are marked with ✓. For scoring, system outputs are independently aligned to the gold text, which accounts for text transformations made by some tools and allows precision and recall to be computed.

provides for decent-size test sets in many languages. We manually corrected all sentence segmentations. While some sets were already well-segmented, some more recent years were *extremely* under-segmented. In Table 5, we show the test sets' line counts before and after manual correction.[8] Additionally, we report the % of candidate sites with a true ✓ label, which provides a measure of the ambiguity of the punctuation. Many ⊗ positions occur in acronyms, such as “U.S.A.", embedded quotes, ellipsis, or in company names such as “Yahoo!".

6 Experimental Setup

We consider three language settings: (i) monolingual English, (ii) a multilingual setting that includes the set of recent WMT languages plus Arabic, and (iii) a much larger multilingual setting that includes the previous languages plus all languages with at least $10k$ lines in the WikiMatrix (Schwenk et al., 2019a) dataset.

Starting with the English setting, we investigate the performance of a basic model and vary parameters such as context size, embedding size, and vocabulary size. After finding an optimal setting, we expand to the first multilingual setting and repeat. We train a single multilingual model that is agnostic of language and does not need language specification as input. Similar to the monolingual setting, we vary the aforementioned parameters, and compare the best model to baselines (§6.3). In order to test expandability, we then train with the same parameters on the largest set of languages (using the additional WikiMatrix data), and compare to the previous model's performance.

While we do not widely experiment with additional monolingual settings, we train monolingual models in each language to compare against the multilingual models' performance. We report the comparison of these three settings to baselines in Table 5.

6.1 Datasets

We train our English model on a subset of the WSJ[9] and the English News Commentary datasets provided by WMT.[10]

To expand to a multilingual setting, we consider the set of all WMT Task languages and Arabic (23 in total) allowing us to leverage the various monolingual datasets (Joanis et al., 2020) released as part of the WMT workshops—often using News Commentary datasets, as well as WikiMatrix (Schwenk et al., 2019a), CCMatrix (Schwenk et al., 2019b), and Global Voices (Nguyen and Daumé III, 2019). For validation data, we use WMT test sets when available, and IWSLT (Cettolo et al., 2017) for Arabic.

We experimented with (i) balancing the data so each language has equal amounts of data, (ii) normalizing the amount of data per language based on the relative ambiguity (measured by percent of candidate sites labeled as true ✓), and (iii) using all available data. We find that the third method performs the best and thus report under this setting.

In the larger multilingual setting, we consider all WikiMatrix languages with more than $10k$ unique lines (64 additional languages) and do not expand the validation set. For a complete list of datasets, please see Table 7 in Appendix A.

6.2 Training

For each vocabulary size, we train a SentencePiece (Kudo and Richardson, 2018) model over the training data.

We use a binary cross-entropy loss over the labels, Adam optimizer with a learning rate of 0.0001, and a λ of 1.0 (English) and 20.0 (multilingual)

[8] `iu` was left uncorrected due to the fact that available bitext often aligned “sentences" with singular or compound sentences in English and a lack of automatic translation corresponding to sentences.

[9] Sections 1-2, 7-23 for training; section 24 for validation, and sections 03-06 for test in order to mirror the splits in Bird and Loper (2004)

[10] `http://data.statmt.org/news-commentary/v15/`

on the ✓ class (with the exception of the experiments in §7.4). We use a batch size of 25k instances, and compute F1 over the validation data every 500 batches, saving the model with the highest inference-time F1 score. This is the collective F1 score across all languages in the multilingual settings. If the model has not improved in 15 validations, training terminates.

The models were trained on a Tesla V100 GPU. The monolingual models took approximately 2 hours to train while the multilingual models took approximately 10-15 hours.

6.3 Baselines

We use the following existing tools as baselines:

Always split on every candidate site. This serves as a lower-bound for our precision metric.

Splitta (Gillick, 2009) ships with both SVM and Naive Bayes models. It targets English texts. We found similar performance and only report the Naive Bayes scores.

NLTK Punkt Kiss and Strunk (2006) introduce an unsupervised training method for this task which uses frequency of occurences of input features such as casing, punctuation, and length in order to segment. Pretrained models for 18 languages (labeled as PUNKT in Table 5) are packaged with NLTK. NLTK additionally provides the framework to train a new model. We use this to train an additional model on all data (to simulate a multilingual model) and report the results in Table 5 as PUNKT$_{\text{ML}}$. PUNKT (and thus PUNKT$_{\text{ML}}$) does not segment around non-Latin punctuation.

Moses Sentence Splitter uses a list of predefined acronyms and abbreviations for each language. If left token is in this list, it does not split. This circumvents the whole point behind the ambiguity "in the U.S."

SpaCy Sentencizer is a "rule-based system" without specific details and varies from language to language.

7 Monolingual Experiments

We first explore common questions and concerns while focusing on English data and results. We have three main parameters to study: context size, embedding size, and vocabulary size. We additionally consider how the training data affects results–both in relative noise in class labels in addition to

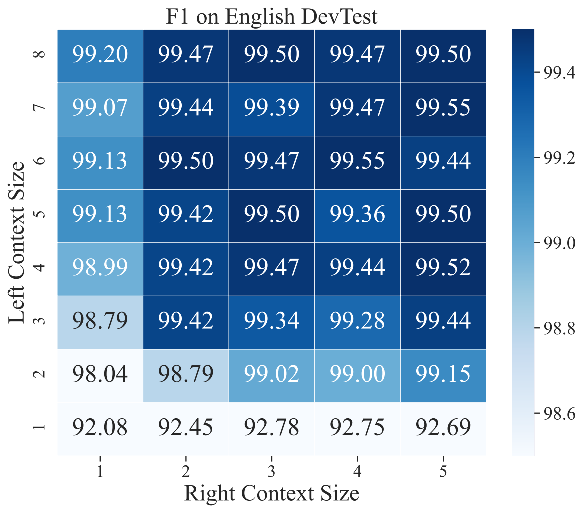

Figure 3: Heat map showing the change in F1 with respect to context size in a linear model. Embedding size and vocabulary size are kept constant at 32 and 125 respectively.

training on shuffled sentences instead of documents. In general, we find our technique creates a monolingual English model (Table 3) that outperforms the baselines.

	F1	Precision	Recall
Always	86.9	76.9	**100.0**
Splitta	99.3	99.6	99.1
Punkt	98.6	98.8	98.4
Moses	98.8	99.7	98.0
SpaCy	88.0	86.3	89.7
Our Tool	**99.8**	**99.8**	99.8

Table 3: Scores on English WSJ 03-06 Test Data. The candidate set is determined the original English punctuation contexts as described in 3.2

7.1 Exploring context size

Starting with a minimal model with an embedding size of 32, and a vocabulary size of 125, we investigate whether such a small model can solve this problem. Our method is rooted in a contextual encoding of the subword tokens inside its context windows, and may benefit from increasing the size of these windows. At the operating point with a very small embedding and vocabulary size, the window size is the determining factor on performance. The results on English in Figure 3 show that a minimal amount of left and right context is necessary; however, left context is more beneficial than right context.

7.2 How large of a model is necessary?

We consider whether increasing the size of the model by doubling the embedding size and quadrupling the vocabulary size can produce better results. While varying the context windows (as seen in Figure 3) can result in increasingly higher scores, varying embedding size and vocabulary size did not produce the same effect. Keeping a fixed context window, we find that any given change in embedding size or vocabulary size increases F1 score by no more than 0.6%. While necessary to find the optimal model, it is clear that the context size is more important to experimentation. We note that a vocabulary size of 2000 tends to perform worse than smaller sizes while vocabulary sizes of 125 and 500 perform equally well when paired with any embedding size. Each of our monolingual models reported in Table 5 is the result of a grid search over various vocab sizes, and lambda weight (§3.3). We keep context sizes of left (6) and right (4) and embedding size (128) constant.

7.3 Is document context necessary?

Because released monolingual data is often cleaned with sentences being removed and shuffled, it is unreasonable to assume that a set of consecutive sentences will always be available for training.

In order to justify using this data, we repeat a subset of the previous English experiment—testing context and embedding sizes by training the model on the same data that has been shuffled. We test on the same validation data that has *not* been shuffled and retain its document order. In Table 4, we show that shuffling the training data has little impact on performance and document context is unnecessary in this punctuated setting.

	F1	Precision	Recall
Original	99.8	99.7	99.9
Shuffled	99.6	99.6	99.6
Undersegmented	97.5	95.2	99.8

Table 4: Scores on English WSJ 03-06 Test Data. Original is the best model trained on original English monolingual News Commentary data. Shuffled is trained on shuffled data, described in §7.3. Undersegmented is trained on raw Wikipedia, described in §7.4.

7.4 Can we train on undersegmented data?

Uncleaned, unfiltered Wikipedia dumps do not have sentence boundaries in them. The smallest unit is the paragraph. Data scraped from internet sites is likely to have a similar form and much of our monolingual data is not guaranteed to be segmented. In order to justify that this approach works without *already* having segmented data, we show that we can achieve similar results as our previous English results in this setting. We train on one million randomly-selected paragraphs from an English Wikipedia dump. While many $\otimes$ labels are now incorrect due to paragraphs being unsegmented, we assume the $\checkmark$ class is relatively noise-free.

Because we already established that shuffling the data does not affect performance in this setting, the random selection is sufficient. While maintaining previously chosen hyper-parameters—such as context sizes, learning rate, and dropout—we search among potential λ values to use as a weight for the $\checkmark$ label. We find that increasing the λ value to 200.0 achieves the highest F1 of 97.5. An unweighted model performs poorly. While still distant from the cleanly-trained models, it performs significantly better than the poorer baselines. Comparison to our other English models can be seen in Table 4.

8 Multilingual Experiments

After outperforming current baselines in a monolingual English setting, we generalized our approach to work multilingually. The multilingual model can segment text irrespective of input language.

In parallel to the monolingual conditions, we train two-layer transformer models with 6 tokens of left context, and 4 tokens of right context with 128 embedding size. While we did experiment with scaling these for the multilingual model, we found little effect. We additionally scale the vocabulary size to 12,000 to accommodate the larger character sets in Chinese and Japanese. Because more of the additional languages have undersegmented data, we searched over potential lambda weights for the $\checkmark$ class and report the best configuration ($\lambda = 20.0$) in Table 5.

8.1 Discussion

Results of ERSATZ and baselines can be found in Table 5. In all cases, ERSATZ is at least competitive with baselines, if not outperforming them. Although most differences are small it outperforms

	# Orig.	# Corr.	% ✓	PUNKT	PUNKT$_{ML}$	ALWAYS	SPACY	MOSES	ERSATZ$_{M}$	ERSATZ	ERSATZ$_{WM}$
ar	1460	1504	84.9	-	92.7	93.5	90.3	-	**98.2**	98.0	98.0
cs	664	1726	80.1	**99.8**	99.6	96.3	85.3	99.7	**99.8**	**99.8**	**99.8**
de	785	1965	90.2	99.7	99.5	97.9	91.4	**99.9**	**99.9**	99.8	99.8
en	7706	7706	48.6	98.6	87.7	77.0	88.0	98.8	**99.8**	98.7	99.1
es	3000	3064	86.5	**99.1**	98.9	96.5	83.6	98.7	98.8	98.6	98.6
et	2000	2017	78.2	99.3	99.4	90.6	84.0	99.5	**99.8**	99.7	**99.8**
fi	1996	1996	95.0	99.7	99.7	98.9	97.9	99.8	**99.9**	**99.9**	**99.9**
fr	1619	1655	95.0	99.5	99.6	98.2	90.4	**99.7**	**99.7**	99.6	99.4
gu	1016	1018	92.3	-	**100.0**	97.9	3.8	99.7	99.8	**100.0**	**100.0**
hi	2507	2521	68.6	-	14.4	83.7	90.6	15.1	98.5	**99.1**	98.6
iu	2971	2971	59.1	-	91.3	63.9	-	-	86.1	**93.7**	93.6
ja	993	1072	89.4	-	0.2	98.1	93.7	-	**99.9**	**99.9**	**99.9**
kk	1000	1002	92.2	-	99.6	97.1	-	-	99.7	99.8	**99.9**
km	2320	2361	96.3	-	2.0	99.1	-	-	**99.7**	**99.7**	**99.7**
lt	1000	1000	59.2	-	94.7	85.5	76.6	98.6	98.8	98.8	**98.9**
lv	2001	2017	76.4	-	99.4	90.3	88.6	99.6	**99.7**	99.5	99.6
pl	1001	1005	70.7	98.3	94.8	90.1	78.9	92.8	93.4	99.1	**99.2**
ps	2719	2726	96.4	-	**99.4**	99.1	-	-	99.3	99.3	99.3
ro	1999	2000	89.1	-	98.7	97.0	90.9	98.5	**99.3**	**99.3**	99.2
ru	991	991	88.4	98.8	98.1	96.4	91.3	99.4	99.3	99.4	**99.5**
ta	997	1005	66.1	-	92.3	89.6	89.3	93.8	**98.1**	96.9	96.6
tr	3000	3009	67.5	-	95.8	85.2	85.1	99.5	**99.6**	99.5	99.5
zh	2000	2003	85.1	-	-	99.2	96.6	-	**100.0**	**100.0**	**100.0**
all	45k	48k	73.3	-	87.6	89.0	-	-	-	**98.9**	**98.9**

Table 5: Test set statistics (left block) and F1 scores (right block) on our test data. % ✓ denotes the number of candidate sites with a true ✓ label. PUNKT$_{ML}$ denotes PUNKT model trained on our data. Lack of a score means the model was not available for that language. ERSATZ$_{M}$ denotes monolingual models, ERSATZ the WMT-languages multilingual model, and ERSATZ$_{WM}$ the model trained with additional WikiMatrix languages.

SpaCy in all languages and often outperforms both Punkt and Moses.

The Moses splitter is an interesting case. It identifies split points via a mix of general and language-specific regular expressions, which are then filtered against a curated list of "non-breaking prefixes". This results in a conservative segmenter that will not (for example) allow a sentence to end with the token *U.S.*. As such, its high performance is notable. However, the comparison is likely unfair, since it was likely built and refined against the news datasets that constitute our WMT test sets. This approach is therefore effective in this domain, but may not generalize. Our single multilingual model, trained on noisy data, performs nearly identically.

8.2 Performance across languages

Sentence segmentation is not equally difficult in all languages or with respect to all punctuation. The '.' is by far the most ambiguous form of punctuation and is frequently used as an abbreviation marker. Other scripts using their own punctuation, such as Hindi, have specified a particular marker (the Devanagari Danda) as a sentence-ending punctuation that is rarely used sentence-internally. In these cases, ambiguity is introduced when alternative punctuation (such as '.' or '...') is used. Additionally, even languages with the same scripts may not have the same level of ambiguity. French has the smallest number of punctuated contexts occurring sentence-internally within our test set, while English has the most.

We note that the multilinguality of our model hurts the near-perfect performance that we see in the monolingual English models. We additionally note that some monolingual models perform worse than the multilingual model (see `pl` in Table 5). We hypothesize that this may be due to a lack of data, and the additional languages contain similar contexts, so the model may learn more about casing, punctuation, and length with additional data.

8.3 Scaling to more languages

While we note that it is difficult to evaluate many of the world's languages due to a lack of gold standard test data, we test for scalability by including additional languages (as described in §6) during training and noting any changes in performance on the evaluable languages. We include 64 additional languages (see Table 7 in the Appendix for comprehensive list) to bring us to a total of 87 languages. Table 5 also includes scores from a larger multilingual model (ERSATZ$_{\mathrm{WM}}$) that was built with these 64 additional languages. Overall, we find very little change between these two settings. With `en`, we actually see some improvement in performance from the smaller multilingual model. Generally, there is not significant degradation of scores, implying this technique can generalize to additional languages.

8.4 How does size affect the speed?

With our context construction method, we benefit from batching to decrease runtime, since the decision at each candidate point is dependent only on its immediate window. We benchmark our models as well as the baselines (Table 6). While our models are slower than some baselines, we find that increasing the size of the model does not dramatically increase the runtime. Additionally, the rate (in tokens per second) is roughly constant.

Layer (# layers)	# params	Time (s)	F1
Linear (x1)	1.7M	33	97.5
Linear (x2)	1.7M	35	98.0
Transformer (x1)	2.3M	74	98.7
Transformer (x2)	2.9M	172	98.7
Spacy	-	13.8	88.0
Moses	-	1164	98.8
Punkt	-	3.2	98.6

Table 6: Time in seconds for 1 million English tokens in input file. F1 is score on English Test Set. We show various size encoders for our method. Linear is a linear layer with TanH activation.

9 Summary

As one of the earliest steps in NLP pipelines, sentence segmentation is an important task. However, it has not to this date received proper experimental attention, relying instead on ad hoc methods. It is a good time to correct this oversight, as NLP moves to the use of larger and larger corpora covering more and more languages. Even as the field moves towards processing text at the paragraph or document level directly, it is likely that sentence processing will be with us for some time.

We show here that a simple context-based model can produce state-of-the-art results with a modest hyperparameter search, trained on noisy annotations from imperfectly-segmented data. Together with a straightforward multilingual approach to identifying candidate split points and training on noisy segmented data, our single model performs well across a range of languages. More fundamentally, we have defined an experimental framework for benchmarking and future comparative work.

Missing from our paper is an evaluation of the effect of these tools on downstream tasks. An obvious candidate for future work is to conduct this evaluation. It is possible that some tasks will not be affected by small differences among the best performing models, but this work at least sheds light on those differences. Another obvious direction is to look at approaches that would work for *unpunctuated* text (e.g., Wang et al. (2019)). This would expand the functionality of segmenters into other important areas, such as speech translation, and to languages, like Thai, that do not mark ends of sentences.

Acknowledgments

The authors wish to thank Elizabeth Salesky, Carlos Aguirre, Jacob Bremerman and the anonymous reviewers for helpful technical discussions and feedback.

References

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of* *the Fifth Conference on Machine Translation*, pages 1–55, Online. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuitho Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the iwslt 2017 evaluation campaign. In *14th International Workshop on Spoken Language Translation*, pages 2–14, Tokyo, Japan.

Dan Gillick. 2009. Sentence boundary detection and the problem with the U.S. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 241–244, Boulder, Colorado. Association for Computational Linguistics.

Eric Joanis, Rebecca Knowles, Roland Kuhn, Samuel Larkin, Patrick Littell, Chi-kiu Lo, Darlene Stewart, and Jeffrey Micher. 2020. The Nunavut hansard Inuktitut–English parallel corpus 3.0 with preliminary machine translation results. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2562–2572, Marseille, France. European Language Resources Association.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Philipp Koehn, 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. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Khanh Nguyen and Hal Daumé III. 2019. Global Voices: Crossing borders in automatic news summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 90–97, Hong Kong, China. Association for Computational Linguistics.

David D. Palmer and Marti A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 23(2):241–267.

Ines Rehbein, Josef Ruppenhofer, and Thomas Schmidt. 2020. Improving sentence boundary detection for spoken language transcripts. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 7102–7111, Marseille, France. European Language Resources Association.

Michael D. Riley. 1989. Some applications of tree-based modelling to speech and language. In *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, October 15-18, 1989*.

George Sanchez. 2019. Sentence boundary detection in legal text. In *Proceedings of the Natural Legal Language Processing Workshop 2019*, pages 31–38, Minneapolis, Minnesota. Association for Computational Linguistics.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2019a. Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. *CoRR*, abs/1907.05791.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, and Armand Joulin. 2019b. Ccmatrix: Mining billions of high-quality parallel sentences on the WEB. *CoRR*, abs/1911.04944.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Xiaolin Wang, Masao Utiyama, and Eiichiro Sumita. 2019. Online sentence segmentation for simultaneous interpretation using multi-shifted recurrent neural network. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 1–11, Dublin, Ireland. European Association for Machine Translation.

Nianwen Xue and Yaqin Yang. 2011. Chinese sentence segmentation as comma classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 631–635, Portland, Oregon, USA. Association for Computational Linguistics.

Nina Zhou, AiTi Aw, Nattadaporn Lertcheva, and Xuancong Wang. 2016. A word labeling approach to Thai sentence boundary detection and POS tagging. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 319–327, Osaka, Japan. The COLING 2016 Organizing Committee.

A Appendix

	Dataset	# Lines	# Tokens		Dataset	# Lines	# Tokens
ar	News Comm. v15	181k	10M	**kk**	News Comm. v15	16.4k	280k
	WikiMatrix	774k	16M		News Crawl	1.1M	14M
cs	News Comm. v15	277k	5.2M	**km**	JW Corpus	107k	4.6M
	WikiMatrix	429k	7.3M		Common Crawl	343k	2.0M
de	News Comm. v15	422k	8.9M	**lt**	News Crawl	2.5M	37M
	WikiMatrix	1M	19M		WikiMatrix	84.8k	1.1M
en	News Comm. v15	609k	13M	**lv**	News Crawl	1.8M	29M
	WSJ (sec 00-02;07-23)	40k	819k				
es	News Comm. v15	465k	12M	**pl**	Global Voices	58k	890k
					Wikipedia	405k	6.6M
					News Crawl	3.0M	44M
et	News Crawl	1.6M	22M	**ps**	News Crawl	64.0k	1.8M
	WikiMatrix	152k	5.1M		SADA	132k	4.1M
					SYSTRAN	196k	5.1M
					TRANSTAC	75k	1.2M
fi	News Crawl	4.7M	50M	**ro**	Global Voices	4043	76k
	WikiMatrix	207k	2.6M		WikiMatrix	223k	4.7M
					News Crawl	6.9M	140M
fr	News Comm. v15	415k	10M	**ru**	News Comm. v15	377k	7.3M
	WikiMatrix	2.2M	50M		WikiMatrix	2.2M	37M
gu	News Crawl	283k	3.8M	**ta**	News Crawl	501k	5.3M
	Common Crawl	164k	1.3M		WikiMatrix	61.0k	532k
hi	News Comm. v15	7815	213k	**tr**	Global Voices	6529	80k
	WikiMatrix	1.1M	20M		WikiMatrix	304k	4.5M
	News Crawl	135k	3.0M		News Crawl	7.9M	108M
iu	N.H.I 3.0	1.3M	8.0M	**zh**	News Comm. v15	445k	772k
					WikiMatrix	492k	890k
ja	News Comm.	2983	4390				
	News Crawl	3.4M	6.9M				

Table 7: Multilingual Datasets Line Count and Token Count.

	Dataset	# Lines	# Tokens		Dataset	# Lines	# Tokens
ar	News Comm v15	1637	38k	**kk**	News Comm v15	3000	38k
	IWSLT 2017	**1504**	**20k**		**WMT19 Test**	**1002**	**30k**
cs	WMT18 Test	3008	47k	**km**	WMT WikiDev	2609	14k
	WMT20 Test	**1726**	**26k**		**WMT20 Test**	**2361**	**15k**
de	WMT19 Test	2009	31k	**lt**	News Comm v15	3000	44k
	WMT20 Test	**1965**	**31k**		**WMT19 Test**	**1000**	**17k**
en	News Commentary	3000	56k	**lv**	News Commentary	3000	49k
	WMT20 Test (en-cs)				**WMT17 Test**	**2017**	**33k**
	WSJ 03-06; 24	**10k**	**277k**				
es	WMT11 Test	3013	69k	**pl**	News Commentary v15	3000	16k
	WMT13 Test Set	**3064**	**62k**		**WMT20 Test Set**	**1005**	**16k**
et	News Commentary	3000	41k	**ps**	WMT Wiki Dev	3166	64k
	WMT18 Test Set	**2017**	**30k**		**WMT20 Test Set**	**2726**	**55k**
fi	WMT18 Test Set	3031	38k	**ro**	News Commentary v15	3000	60k
	WMT19 Test Set	**1996**	**21k**		**WMT16 Test Set**	**2000**	**43k**
fr	WMT15 Test Set	1502	25k	**ru**	WMT18 Test Set	3000	52k
	WMT20 Test Set (fr-de)	**1655**	**33k**		**WMT20 Test Set**	**991**	**15k**
gu	News Commentary	3000	40k	**ta**	News Commentary	3000	32k
	WMT19 Test Set	**1018**	**14k**		**WMT20 Test Set**	**1005**	**13k**
hi	News Commentary	3000	56k	**tr**	WMT16 Test Set	3011	44k
	WMT14 Test Set	**2521**	**57k**		**WMT18 Test Set**	**3009**	**46k**
iu	N.H.I 3.0 Dev	3028	27k	**zh**	WMT18 Test Set	4097	5.8k
	N.H.I 3.0 Dev	**3028**	**27k**		**WMT20 Test Set**	**2003**	**3.7k**
ja	News Commentary	3000	5.9k				
	WMT20 Test Set	**1072**	**1888**				

Table 8: Dev and Test Data. Test Data is bolded. All News and Wikipedia sets come from WMT news translation tasks except `ar`. All test sets are `lang-en` unless otherwise noted. NHI is the Nunavut Hansard Inuktitut English Parallel Corpus-3.0. When News Commentary was used, the bottom N lines were taken.

	# lines	# tokens		# lines	# tokens
an	52k	93k	la	45k	50k
arz	35k	57k	lb	43k	59k
as	16k	15k	lmo	10k	16k
az	164k	170k	mg	12k	18k
bar	40k	49k	mk	452k	672k
ba	101k	112k	ml	150k	130k
be	164k	223k	mr	216k	225k
bg	454k	523k	mwl	32k	78k
bn	360k	452k	nds-nl	14k	22k
br	43k	53k	nds	95k	145k
bs	502k	831k	ne	70k	81k
ca	459k	417k	nl	456k	348k
ceb	80k	188k	no	457k	472k
da	453k	494k	oc	171k	389k
el	454k	555k	pt	460k	367k
eo	454k	554k	sh	454k	582k
eu	305k	369k	simple	465k	666k
fa	427k	818k	si	182k	281k
fo	38k	46k	sk	453k	539k
fy	56k	84k	sl	451k	624k
gl	453k	512k	sq	262k	523k
gom	22k	19k	sr	452k	520k
he	458k	387k	sv	452k	388k
hr	455k	551k	sw	70k	118k
hu	456k	353k	te	213k	170k
hy	23k	52k	tg	17k	20k
id	456k	468k	tl	122k	237k
is	124k	160k	tt	78k	80k
it	469k	386k	uk	466k	313k
jv	27k	40k	vi	456k	646k
ka	42k	81k	wuu	46k	7k
ko	454k	395k			

Table 9: Additional WikiMatrix languages with line and token counts for training data. Language code based on Wikipedia codes.