A Neural Network Architecture for Multilingual Punctuation Generation

Miguel Ballesteros[1] **Leo Wanner**[1,2]
[1]NLP Group, Universitat Pompeu Fabra, Barcelona, Spain
[2] Catalan Institute for Research and Advanced Studies (ICREA)
miguel.ballesteros@upf.edu leo.wanner@upf.edu

Abstract

Even syntactically correct sentences are perceived as awkward if they do not contain correct punctuation. Still, the problem of automatic generation of punctuation marks has been largely neglected for a long time. We present a novel model that introduces punctuation marks into raw text material with transition-based algorithm using LSTMs. Unlike the state-of-the-art approaches, our model is language-independent and also neutral with respect to the intended use of the punctuation. Multilingual experiments show that it achieves high accuracy on the full range of punctuation marks across languages.

1 Introduction

Although omnipresent in (language learner) grammar books, punctuation received much less attention in linguistics and natural language processing (Krahn, 2014). In linguistics, punctuation is generally acknowledged to possess different functions. Its traditionally most studied function is that to encode prosody of oral speech, i.e., the prosodic *rhetorical* function; see, e.g., (Kirchhoff and Primus, 2014) and the references therein. In particular the comma is assumed to possess a strong rhetorical function (Nunberg et al., 2002). Its other functions are the grammatical function, which leads it to form a separate (along with semantics, syntax, and phonology) grammatical submodule (Nunberg, 1990), and the syntactic function (Quirk et al., 1972), which makes it reflect the syntactic structure of a sentence.

The different functions of punctuation are also reflected in different tasks in natural language processing (NLP): introduction of punctuation marks into a generated sentence that is to be read aloud, restoration of punctuation in speech transcripts, parsing under consideration of punctuation, or generation of punctuation in written discourse. Our work is centered in the last task. We present a novel punctuation generation algorithm that is based on the transition-based algorithm with long short-term memories (LSTMs) by Dyer et al. (2015) and character-based continuous-space vector embeddings of words using bidirectional LSTMs (Ling et al., 2015b; Ballesteros et al., 2015). The algorithm takes as input raw material without punctuation and effectively introduces the full range of punctuation symbols. Although intended, first of all, for use in sentence generation, the algorithm is function- and language-neutral, which makes it different, compared to most of the state-of-the-art approaches, which use function- and/or language-specific features.

2 Related Work

The most prominent punctuation-related NLP task has been so far introduction (or restoration) of punctuation in speech transcripts. Most often, classifier models are used that are trained on n-gram models (Gravano et al., 2009), on n-gram models enriched by syntactic and lexical features (Ueffing et al., 2013) and/or by acoustic features (Baron et al., 2002; Kolář and Lamel, 2012). Tilk and Alumäe (2015) use a lexical and acoustic (pause duration) feature-based LSTM model for the restoration of periods and commas in Estonian speech transcripts. The grammatical and syntactic functions of punctuation have been addressed in the context of written

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1048–1053,
Austin, Texas, November 1-5, 2016.

language. Some of the proposals focus on the grammatical function (Doran, 1998; White and Rajkumar, 2008), while others bring the grammatical and syntactic functions together and design rule-based grammatical resources for parsing (Briscoe, 1994) and surface realization (White, 1995; Guo et al., 2010). Guo et al. (2010) is one of the few works that is based on a statistical model for the generation of punctuation in the context of Chinese sentence generation, trained on a variety of syntactic features from LFG f-structures, preceding punctuation bigrams and cue words.

Our proposal is most similar to Tilk and Alumäe (2015), but our task is more complex since we generate the full range of punctuation marks. Furthermore, we do not use any acoustic features. Compared to Guo et al. (2010), we do not use any syntactic features either since our input is just raw text material.

3 Model

Our model is inspired by a number of recent works on neural architectures for structure prediction: Dyer et al. (2015)'s transition-based parsing model, Dyer et al. (2016)'s generative language model and phrase-structure parser, Ballesteros et al. (2015)'s character-based word representation for parsing, and Ling et al. (2015b)'s part-of-speech tagging .

3.1 Algorithm

We define a transition-based algorithm that introduces punctuation marks into sentences that do not contain any punctuation. In the context of NLG, the input sentence would be the result of the surface realization task (Belz et al., 2011). As in transition-based parsing (Nivre, 2004), we use two data structures: Nivre's *queue* is in our case the *input buffer* and his *stack* is in our case the *output buffer*. The algorithm starts with an input buffer full of words and an empty output buffer. The two basic actions of the algorithm are SHIFT, which moves the first word from the input buffer to the output buffer, and GENERATE, which introduces a punctuation mark after the first word in the output buffer. Figure 1 shows an example of the application of the two actions.

At each stage t of the application of the algorithm, the state, which is defined by the contents of the output and input buffers, is encoded in terms of a vector $\mathbf{s}_t$; see Section 3.3 for different alternatives of state representation. As Dyer et al. (2015), we use $\mathbf{s}_t$ to compute the probability of the action at time t as:

Transition	Output	Input
	[ ]	[No it was not]
SHIFT	[No]	[it was not]
GENERATE(",")	[No ,]	[it was not]
SHIFT	[No , it]	[was not]
SHIFT	[No , it was]	[not]
SHIFT	[No , it was not]	[ ]
GENERATE(".")	[No, it was not .]	[ ]

Figure 1: Transition sequence for the input sequence *No it was not* – with the output *No, it was not.*

$$p(z_t \mid \mathbf{s}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^\top \mathbf{s}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}} \exp\left(\mathbf{g}_{z'}^\top \mathbf{s}_t + q_{z'}\right)} \quad (1)$$

where $\mathbf{g}_z$ is a vector representing the embedding of the action z, and q_z is a bias term for action z. The set $\mathcal{A}$ represents the actions (either SHIFT or GENERATE(p)).[1] $\mathbf{s}_t$ encodes information about previous actions (since it may include the history with the actions taken and the generated punctuation symbols are introduced in the output buffer, see Section 3.3), thus the probability of a sequence of actions $\boldsymbol{z}$ given the input sequence is:

$$p(\boldsymbol{z} \mid \boldsymbol{w}) = \prod_{t=1}^{|\boldsymbol{z}|} p(z_t \mid \mathbf{s}_t). \quad (2)$$

As in (Dyer et al., 2015), the model greedily chooses the best action to take given the state with no backtracking.[2]

3.2 Word Embeddings

Following the tagging model of Ling et al. (2015b) and the parsing model of Ballesteros et al. (2015), we compute character-based continuous-space vector embeddings of words using bidirectional LSTMs (Graves and Schmidhuber, 2005) to learn similar representation for words that are similar from an orthographic/morphological point of view.

[1] Note that GENERATE(p) includes all possible punctuations that the language in question has, and thus the number of classes the classifier predicts in each time step is #punctuations + 1.

[2] For further optimization, the model could be extended, for instance, by beam-search.

The character-based representations may be also concatenated with a fixed vector representation from a neural language model. The resulting vector is passed through a component-wise rectifier linear unit (ReLU). We experiment with and without pretrained word embeddings. To pretrain the fixed vector representations, we use the skip n-gram model introduced by Ling et al. (2015a).

3.3 Representing the State

We work with two possible representations of the input and output buffers (i.e, the state $\mathbf{s}_t$): (i) a look-ahead model that takes into account the immediate context (two embeddings for the input and two embeddings for the output), which we use as a baseline, and (ii) the LSTM model, which encodes the entire input sequence and the output sentence with LSTMs.

3.3.1 Baseline: Look-ahead Model

The look-ahead model can be interpreted as a 4-gram model in which two words belong to the input and two belong to the output. The representation takes the average of the two first embeddings of the output and the two first embeddings at the front of the input. The word embeddings contain all the richness provided by the character-based LSTMs and the pretrained skip n-gram model embeddings (if used). The resulting vector is passed through a component-wise ReLU and a softmax transformation to obtain the probability distribution over the possible actions given the state $\mathbf{s}_t$; see Section 3.1.

3.3.2 LSTM Model

The baseline look-ahead model considers only the immediate context for the input and output sequences. In the proposed model, we apply recurrent neural networks (RNNs) that encode the entire input and output sequences in the form of LSTMs. LSTMs are a variant of RNNs designed to deal with the vanishing gradient problem inherent in RNNs (Hochreiter and Schmidhuber, 1997; Graves, 2013). RNNs read a vector $\mathbf{x}_t$ at each time step and compute a new (hidden) state $\mathbf{h}_t$ by applying a linear map to the concatenation of the previous time step's state $\mathbf{h}_{t-1}$ and the input, passing then the outcome through a logistic sigmoid non-linearity.

We use a simplified version of the stack LSTM model of Dyer et al. (2015). The input buffer is encoded as a stack LSTM, into which we PUSH the entire sequence at the beginning and POP words from it at each time step. The output buffer is a sequence, encoded by an LSTM, into which we PUSH the final output sequence. As in (Dyer et al., 2015), we include a third sequence with the history of actions taken, which is encoded by another LSTM. As already mentioned above, the three resulting vectors are passed through a component-wise ReLU and a softmax transformation to obtain the probability distribution over the possible actions that can be taken (either to shift or to generate a punctuation mark), given the current state $\mathbf{s}_t$; see Section 3.1.

4 Experiments

To test our models, we carried experiments on five languages: Czech, English, French, German, and Spanish. English, French and Spanish are generally assumed to be characterized by prosodic punctuation, while for German the syntactic punctuation is more dominant (Kirchhoff and Primus, 2014). Czech punctuation also leans towards syntactic punctuation (Kolář et al., 2004), but due to its rather free word order we expect it to reflect prosodic punctuation as well.

The punctuation marks that the models attempt to predict (and that also occur in the training sets) for each language are listed in Table 1.[3] Commas represent around 55% and periods around 30% of the total number of marks in the datasets.

Czech	‘.’, ‘,’, ‘–’, ‘(’, ‘)’, ‘:’, ‘/’, ‘?’, ‘%’, ‘*’, ‘=’, ‘|’, ‘”’, ‘+’, ‘;’, ‘!’, ‘o’, ‘“’, ‘&’, ‘[’, ‘]’, ‘§’
English	‘–’, ‘(’, ‘)’, ‘,’, ‘ ” ’, ‘.’, ‘…’, ‘:’, ‘;’, ‘?’, ‘ “ ’, ‘}’, ‘{’
French	‘ ” ’, ‘,’, ‘–’, ‘:’, ‘?’, ‘(’, ‘)’, ‘.’, ‘!’, ‘…’
German	‘ ” ’, ‘(’, ‘)’, ‘,’, ‘.’, ‘/’, ‘:’, ‘–’, ‘…’, ‘?’, ‘ “ ’
Spanish	‘ ” ’, ‘(’, ‘)’, ‘,’, ‘–’, ‘.’, ‘:’, ‘?’, ‘¿’, ‘!’, ‘¡’

Table 1: Punctuation marks covered in our experiments.

4.1 Setup

The stack LSTM model uses two layers, each of dimension 100 for each input sequence. For both the

[3]The consideration of some of the symbols listed in Table 1 as punctuation marks may be questioned (see, e.g., ‘+’ or ‘§’ for Czech). However, all of them are labeled as punctuation marks in the corresponding tag sets, such that we include them.

Commas															
	Czech			English			French			German			Spanish		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
LookAhead	78.79	43.54	56.09	75.60	38.52	51.04	54.00	22.76	32.02	68.87	32.89	44.52	63.17	19.15	29.39
LookAhead + Pre	–	–	–	75.94	40.81	53.09	–	–	–	71.30	39.62	50.94	58.03	26.67	36.54
LSTM	**80.79**	**68.30**	**74.02**	78.88	70.02	74.19	**61.73**	**44.52**	**51.73**	73.78	65.45	69.37	64.01	42.73	51.25
LSTM + Pre	–	–	–	**80.83**	**74.81**	**77.70**	–	–	–	**76.56**	**69.19**	**72.69**	**65.65**	**45.33**	**53.63**
Periods															
	Czech			English			French			German			Spanish		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
LookAhead	82.62	**95.64**	88.65	88.51	97.76	92.91	71.34	94.61	81.34	77.10	97.76	86.21	73.13	99.13	84.17
LookAhead + Pre	–	–	–	87.44	97.71	92.29	–	–	–	78.26	95.93	86.20	73.16	**99.29**	84.25
LSTM	**89.39**	93.66	**91.48**	93.07	**98.31**	95.62	**76.38**	**95.47**	**84.86**	84.75	98.18	90.97	**74.70**	98.65	**85.02**
LSTM + Pre	–	–	–	**94.44**	98.06	**96.22**	–	–	–	**85.65**	**98.39**	**91.58**	74.24	98.57	84.69
Average															
	Czech			English			French			German			Spanish		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
LookAhead	80.90	58.57	67.95	82.72	52.72	64.40	60.67	32.33	42.18	75.82	52.58	62.10	67.50	33.88	45.12
LookAhead + Pre	–	–	–	81.83	53.90	64.99	–	–	–	75.75	54.57	63.65	64.80	38.58	48.36
LSTM	**82.42**	**69.11**	**75.18**	**84.89**	71.23	77.46	**65.34**	**45.52**	**53.66**	80.03	65.90	72.28	67.78	47.80	56.06
LSTM + Pre	–	–	–	83.72	**74.56**	**78.87**	–	–	–	**81.60**	**67.47**	**73.87**	**68.09**	**49.21**	**57.13**

Table 2: Results of the LSTM model and the Baseline (Look-ahead model) for precision, recall and F score for commas, periods and micro average for all punctuation symbols (including commas and periods) listed in Table 1. +Pre refers to models that include pretrained word embeddings.

look-ahead and the stack LSTM models, character-based embeddings, punctuation embeddings and pretrained embeddings (if used) also have 100 dimensions. Both models are trained to maximize the conditional log-likelihood (Eq. 2) of output sentences, given the input sequences.

For Czech, English, German, and Spanish, we use the wordforms from the treebanks of the CoNLL 2009 Shared Task (Hajič et al., 2009); the French dataset is by Candito et al. (2010). Development sets are used to optimize the model parameters; the results are reported for the held-out test sets.

4.2 Results and Discussion

Table 2 displays the outcome of the experiments for periods and commas in all five languages and summarizes the overall performance of our algorithm in terms of the micro-average figures. In order to test whether pretrained word embeddings provide further improvements, we incorporate them for English, Spanish and German.[4]

The figures show that the LSTMs that encode the entire context of a punctuation mark are better than a strong baseline that takes into account a 4-gram sliding window of tokens. They also show that character-based representations are already useful for the punctuation generation task on their own, but when concatenated with pretrained vectors, they are even more useful.

The model is capable of providing good results for all languages, being more consistent for English, Czech and German. Average sentence length may indicate why the model seems to be worse for Spanish and French, since sentences are longer in the Spanish (29.8) and French (27.0) datasets, compared to German (18.0), Czech (16.8) or English (24.0). The training set is also smaller in Spanish and French compared to the other languages. It is worth noting that the results across languages are not directly comparable since the datasets are different, and as shown in Table 1, the sets of punctuation marks that are to be predicted diverge significantly.

The figures in Table 2 cannot be directly compared with the figures reported by Tilk and Alumäe (2015) for their LSTM-model on period and comma restoration in speech transcripts: the tasks and datasets are different.

Our results prove that the state representation (through LSTMs, which have already been shown to be effective for syntax (Dyer et al., 2015; Dyer et al., 2016)) and character-based representations (which allow similar embeddings for words that are mor-

[4]Word embeddings for English, Spanish and German are trained using the AFP portion of the English Gigaword corpus (version 5), the German monolingual training data from the 2010 Machine Translation Workshop, and the Spanish Gigaword version 3 respectively.

phologically similar (Ling et al., 2015b; Ballesteros et al., 2015)) are capturing strong linguistic clues to predict punctuation.

5 Conclusions

We presented an LSTM-based architectured that is capable of adding punctuation marks to sequences of tokens as produced in the context of surface realization without punctuation with high quality and linear time.[5] Compared to other proposals in the field, the architecture has the advantage to operate on sequences of word forms, without any additional syntactic or acoustic features. This tool could be used for ASR (Tilk and Alumäe, 2015) and grammatical error correction (Ng et al., 2014). In the future, we plan to create cross-lingual models by applying multilingual word embeddings (Ammar et al., 2016).

Acknowledgments

This work was supported by the European Commission under the contract numbers FP7-ICT-610411 (MULTISENSOR) and H2020-RIA-645012 (KRISTINA).

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*, abs/1602.01925.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.

Don Baron, Elizabeth Shriberg, and Andreas Stolcke. 2002. Automatic punctuation and disfluency detection in multi-party meetings using prosodic and lexical cues. In *Proceedings of the International Conference on Spoken Language Processing*, pages 949–952, Denver, CO.

Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226.

Ted Briscoe. 1994. Parsing (with) punctuation. Technical report, Rank Xerox Research Centre, Grenoble, France.

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010. Statistical French dependency parsing: treebank conversion and first results. In *Proceedings of the LREC*.

Christine D. Doran. 1998. *Incorporating Punctuation into the Sentence Grammar*. Ph.D. thesis, University of Pennsylvania.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*.

Agustín Gravano, Martin Jansche, and Michiel Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. In *Proceedings of the ICASSP 2009*, pages 4741–4744.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Yuqing Guo, Haifeng Wang, and Josef van Genabith. 2010. A linguistically inspired statistical model for chinese punctuation generation. *ACM Transactions on Asian Language Information Processing*, 9(2).

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.

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

Frank Kirchhoff and Beatrice Primus. 2014. The architecture of punctuation systems. A historical case study of the comma in German. *Written Language and Literacy*, 17(2):195–224.

Jáchym Kolář and Lori Lamel. 2012. Development and Evaluation of Automatic Punctuation for French and English Speech-to-Text. In *Proceedings of the 13th Interspeech Conference*, Portland, OR.

[5] The code is available at `https://github.com/miguelballesteros/LSTM-punctuation`

Jáchym Kolář, Jan Švec, and Josef Psutka. 2004. Automatic Punctuation Annotation in Czech Broadcast News Speech. In *Proceedings of the 9th Conference Speech and Computer*, St. Petersburg, Russia.

Albert Edward Krahn. 2014. *A New Paradigm for Punctuation*. Ph.D. thesis, University of Wisconsin-Milwaukee.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015a. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015b. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June. Association for Computational Linguistics.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

Geoffrey Nunberg, Ted Briscoe, and Rodney Huddleston. 2002. Punctuation. In *The Cambridge Grammar of the English Language*, pages 1723–1764. Cambridge University Press, Cambridge.

Geoffrey Nunberg. 1990. *The Linguistics of Punctuation*. CSLI Publications, Stanford, CA.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1972. *A Grammar of Contemporary English*. Longman, London.

Ottokar Tilk and Tanel Alumäe. 2015. LSTM for Punctuation Restoration in Speech Transcripts. In *Proceedings of the 16th Interspeech Conference*, Dresden, Germany.

Nicola Ueffing, Maximilian Bisani, and Paul Vozila. 2013. Improved models for automatic punctuation prediction for spoken and written text. In *Proceedings of the 14th Interspeech Conference*, Lyon, France.

Michael White and Rajakrishnan Rajkumar. 2008. A More Precise Analysis of Punctuation for Broad-Coverage Surface-Realization with CCG. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*, pages 17–24, Manchester, UK.

Michael White. 1995. Presenting punctuation. In *Proceedings of the 5th European Workshop on Natural Language Generation*, pages 107–125, Lyon, France.