Nightmare at test time:
How punctuation prevents parsers from generalizing

Anders Søgaard[1]   **Miryam de Lhoneux**[2]   **Isabelle Augenstein**[1]

[1]Dpt. of Computer Science
University of Copenhagen

[2] Dpt. of Linguistics and Philology
Uppsala University

Abstract

Punctuation is a strong indicator of syntactic structure, and parsers trained on text with punctuation often rely heavily on this signal. Punctuation is a diversion, however, since human language processing does not rely on punctuation to the same extent, and in informal texts, we therefore often leave out punctuation. We also use punctuation ungrammatically for emphatic or creative purposes, or simply by mistake. We show that (a) dependency parsers are sensitive to *both* absence of punctuation and to alternative uses; (b) neural parsers tend to be more sensitive than vintage parsers; (c) training neural parsers *without* punctuation outperforms all out-of-the-box parsers across all scenarios where punctuation departs from standard punctuation. Our main experiments are on synthetically corrupted data to study the effect of punctuation in isolation and avoid potential confounds, but we also show effects on out-of-domain data.

1   Introduction

We study the sensitivity of modern dependency parsers to punctuation. While punctuation was originally motivated by reading aloud, serving the purpose of "breath marks" (Baldwin and Coady, 1978), many modern-day punctuation systems are designed to facilitate grammatical disambiguation. This paper aims to show that for this reason, punctuation can significantly hurt the generalization ability of state-of-the-art syntactic parsers. In other words, syntactic parsers become too reliant on punctuation and therefore suffer from the absence or creative uses of punctuation. Such uses are abundant; see Table 1 for examples from Twitter. Such situations, where highly predictive features are absent or distorted at test time, were referred to in Globerson and Roweis (2006) as *nightmare at test time*. Human reading is very robust to variation in punctuation (Baldwin and Coady,

	No punctuation
(1)	i have so many questions i dont know where to start
	Creative punctuation
(2)	What. The. Fuck. Ever. Dot. Com
(3)	. . . and then , , , , i start to feel ~lonely~
	Both
(4)	I feel like ... idk ... idk ... idk man. Nvm I'm good.

Table 1: Examples of uses of punctuation

1978); so creative use of punctuation does not hurt human reading performance. In effect, sensitivity to punctuation is a major obstacle that prevents our syntactic parser from achieving human-level robustness.

The generalization ability of a dependency parser is usually measured by evaluating its accuracy on held-out data, our yardstick to prevent over-fitting, i.e. we define the degree to which a parser has over-fitted to the training data as the difference between performance on training data and performance on the held-out data. This practice is poor when data is not i.i.d., since the held-out data cannot be assumed to be representative; in such cases, little or no over-fitting does not guarantee our parsers have learned important linguistic generalizations: Rather, the parsers may have over-fitted to superficial cues that are present in *both* the training and test datasets (Jo and Bengio, 2017). We argue that punctuation signs are superficial cues preventing modern parsers from learning appropriately high-level abstractions from our datasets.

Contributions   We evaluate three neural dependency parsers for English, as well as two older alternatives, on a standard benchmark, before and after stripping punctuation, as well as after injecting more punctuation signs in the benchmark.
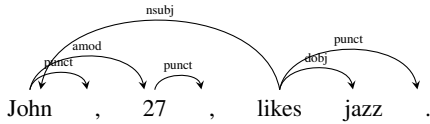
Figure 1: Punctuation in Stanford dependencies

We show that (a) projective parsers are, unsurprisingly, more sensitive to punctuation injection than non-projective ones, since punctuation injection may introduce crossing edges, and (b) neural parsers are more sensitive than vintage parsers. The latter is our main contribution, but we also show that training a neural parser *without* punctuation outperforms all parsers trained in a regular fashion across all punctuation scenarios. Our experiments are on semi-synthetic data to control for confounds, but we also show the parser trained without punctuation is superior on real data with non-standard punctuation.

2 Punctuation in Stanford dependencies

Dependency annotation Dependency annotation refers to the manual assignment of syntactic structures to sentences, following one of several sets of available annotation guidelines. This paper focuses exclusively on the Stanford dependencies annotation scheme (de Marneffe and Manning, 2008). This scheme restricts the set of possible syntactic structures to single-rooted, ordered, possibly non-projective trees whose edges are uniquely labeled by a single dependency label.

Punctuation Punctuation should be distinguished from diacritics and logographs. The two most frequently used punctuation signs are periods and commas. Periods ("."), however, are potentially ambiguous with other uses of dots, typically indicating omissions or pauses. When dots are used emphatically and creatively it is hard to maintain this distinction, and we will simply refer to *dots* and *commas* in this paper. We ignore other punctuation signs, including dashes, question and exclamation marks, and colons and semicolons.

Punctuation is, among other things, used to mark boundaries between constituents of written language. Space characters, for example, separate words, albeit sometimes inconsistently. Spacing is a fairly recent innovation in writing; classical Latin and Greek did not leave spaces between

words, and many Asian languages, e.g., Thai and Lao, still do not. A period is typically used to mark the end of a grammatical sentence, and commas are often used to separate clauses. Therefore, punctuation also correlates strongly with properties of syntactic structures and is therefore very predictive of dependency structures.

Variation in punctuation is often observed in informal texts, but variation may also be the result of errors. Punctuation errors are by far the most frequent error type in scientific writing, for example (Remse et al., 2016). Modern parsers should be robust to such variation, just like humans are (Baldwin and Coady, 1978).

Punctuation in Stanford dependencies In the Stanford dependencies (de Marneffe and Manning, 2008), periods attach to root tokens, and commas attach to their left neighbor or to root tokens; see Figure 1.

3 Experiments

This section describes how we remove and inject punctuation (our perturbation maps), and details of the parsers used in our experiments.

Perturbation maps Since dots consistently attach to the root token of the sentence, and commas attach to their left neighbour or to the root token, we can remove and inject additional punctuation in a sentence without affecting the rest of its syntactic structure and without violating the wellformedness of dependency trees. Note, however, that injecting a root-dominated dot or comma may lead to crossing edges, i.e., turn a projective dependency tree into a non-projective one. This may lead to cascading errors for projective dependency parsers (Ng and Curran, 2015). In our experiments, arc-eager MALTPARSER and STANFORD are the only projective parsers. We therefore propose two *perturbation maps* (Jo and Bengio, 2017): (a) simply removing punctuation, and (b) a simple injection scheme with two parameters χ and δ. Let a dependency structure be an ordered tree with n nodes decorated with words $w_1, \ldots, w_n$. At any node $1 \leq i \leq n$, we (a) inject a comma at position i with probability χ and move nodes $i \leq j \leq n$ to positions $j + 1$, increasing the size of the graph by 1; and (b) inject a dot at position $i+1$ with probability δ and move nodes $i < j \leq n$ to positions $j + 1$, increasing the size of the graph by 1. If we follow standard methodology

Parser	Neural	Trans.-based	Projective
UUPARSER	✓	✓	
KGRAPHS	✓		
MALTPARSER		✓	✓
TURBOPARSER			
STANFORD	✓	✓	✓

Table 2: Our dependency parsers

and ignore punctuation when evaluating parsers, we can compare evaluations before and after applying the injection scheme. It is equally straightforward to remove punctuation without affecting the rest of the dependency tree. Each element w_i to the right of punctuation nodes w_j ($i > j$) moves to the left ($j - 1$) for every punctuation item, decreasing the length of the sentence by 1 each time.

Note that both removing punctuation and our injection scheme can be seen as perturbation maps (Jo and Bengio, 2017) of our dataset, with the following important properties: (a) grammatical structure recognizability, i.e., human ability to correctly process sentences, is preserved (Baldwin and Coady, 1978), (b) surface statistical regularities are qualitatively different, and (c) there exists a non-trivial generalization map between the original dataset and the perturbed version. These properties mean we can use our punctuation injection scheme to evaluate the sensitivity of neural dependency parsers to the surface statistical regularities involving dots and commas (Jo and Bengio, 2017). Since human reading is largely unaffected by erroneous punctuation, we may expect parsers to be robust to absence of punctuation and punctuation injection, as well. Our results clearly show this is not the case; in fact, recently proposed neural dependency parsers are *very* sensitive to differences in punctuation.

Our dependency parsers We use five parsers in our experiments: the Uppsala parser (UU-PARSER) (de Lhoneux et al., 2017a,b), the graph-based parser proposed in (Kiperwasser and Goldberg, 2016)(KGRAPHS) , the arc-eager MALT-PARSER (Nivre et al., 2007), the TURBOPARSER (Fernández-González and Martins, 2015), and the STANFORD parser (Chen and Manning, 2014). UUPARSER is a neural transition-based dependency parser, while KGRAPHS is a neural graph-based parser. MALTPARSER is a more traditional transition-based parser, and TURBOPARSER is a more traditional graph-based parser. Fi-

nally, the STANFORD parser is a projective, neural transition-based dependency parser. All parsers rely on predicted part-of-speech tags, except UU-PARSER (which does not rely on part-of-speech information at all). We use the TURBOTAGGER to obtain those. See Table 2 for an overview of our parsers.

Finally, we also evaluate three non-standard versions of the UUPARSER, namely, a parser trained with the same parameters as the off-the-shelf parser (de Lhoneux et al., 2017b), but which simply *ignores* dots and commas completely (NOPUNCT), and two heavily regularised versions of the parser trained in the standard fashion: (a) a version trained with the drop-out parameter set to 0.8 (zeros out 80% of activations); (b) a version with the gradient clipping parameter set to 0.075. We do so to answer the question of whether more heavily regularized dependency parsers are less sensitive to punctuation (they are not).

4 Results and analysis

We discuss the sensitivity of off-the-shelf dependency parsers to our perturbation maps, comparing to a parser trained after removing punctuation in the training data, as well as to heavily regularised versions of the same parser.

No punctuation We first test our parsers on a version of the validation set where we strip away all punctuation. The data thus consists of newswire (WSJ 22) with punctuation removed. This is similar to Example (1) in Table 1, but in-domain. The results are in the second results column in Table 3, with the relative increases in error listed in the third results column. The drop induced by removing punctuation is quite dramatic: The UUPARSER, for example, suffers an absolute drop of 5.4% LAS or an error increase of 67%. For every three mistakes, UUPARSER does, stripping away punctuation makes it introduce another two. Note that, generally, the relative increase in error is much higher for the three neural parsers, and that the regularisation strategies (drop-out and gradient clipping) do not seem to help much.

Comma and dot injection At medium injection rates, all parsers are sensitive to punctuation injection. With $\delta = 0.05, \gamma = 0.05$, for example, all parsers perform worse than in the absence of punctuation. Our main observation is, again, that neural parsers suffer higher relative increases in errors

	ENGLISH PENN TREEBANK (CORRUPTED)									OUT-OF-DOMAIN			
	δ=0 χ=0	NO PUNCT	Rel.err. incr.	δ=0.01 χ=0.01	δ=0.01 χ=0.05	δ=0.05 χ=0.01	δ=0.05 χ=0.05	δ=0.1 χ=0.1	Rel.err. incr.	GWEB ANSW	REV	FOSTER FOOTBALL	TWITTER
UUPARSER	**0.918**	0.869	0.598	**0.901**	0.867	0.886	0.851	0.794	1.512	**0.676**	0.662	0.770	0.699
KGRAPHS	0.910	0.865	0.500	0.894	0.861	0.876	0.841	0.779	1.456	0.645	0.609	0.774	0.715
MALTPARSER	0.858	0.805	0.373	0.836	0.791	0.804	0.757	0.675	1.289	0.605	0.566	0.721	0.642
TURBOPARSER	0.894	0.852	0.396	0.883	0.858	0.875	0.851	0.802	0.868	0.640	0.595	0.766	**0.722**
STANFORD	0.870	0.816	0.415	0.845	0.808	0.806	0.772	0.688	1.400	0.640	0.608	0.735	0.689
NO PUNCT	0.898	**0.898**	0.000	0.898		0.898			0.000	0.670	**0.669**	**0.792**	0.701
DROPOUT α=0.8	0.904	0.847	0.594	0.884	0.845	0.858	0.820	0.748	1.625	0.661	0.652	0.761	0.682
CLIP t=0.075	0.917	0.871	0.554	0.900	0.864	0.887	0.851	0.793	1.494	0.672	0.657	**0.792**	0.676

Table 3: Labeled attachment scores with punctuation removed. All parsers suffer from absence of or additional punctuation. The relative increase in error ($\frac{1\text{-}BL}{1\text{-}SYS} - 1$; with BL performance on original text; SYS performance under NO PUNCT and $\delta = 0.1$, $\kappa = 0.1$, resp.) for neural parsers is higher than for non-neural parsers. GWEB and FOSTER scores are on development sentences (of at least five words) with no punctuation.

than vintage parsers. Note that the MALTPARSER is a projective parser and therefore has a higher relative increase in error; but TURBOPARSER is much more robust than the other parsers. That said, it still does much worse than the UUPARSER trained without punctuation.

Evaluation on informal text with non-standard punctuation We also evaluate the models on sentences with non-standard punctuation in the development sections in the Google Web Treebank with informal text (from Yahoo Answers and user reviews). Specifically, we evaluate the models on sentences with more than one dot. Again, we show that the neural dependency parser trained without punctuation is superior to the other parsers.

5 Related work

Punctuation in parsing Spitkovsky et al. (2011) introduced the idea of splitting sentences at punctuation and imposing parsing restrictions over the fragments and observed significant improvements in the context of unsupervised parsing. Ng and Curran (2015) aim to prevent cascading errors by enforcing correct punctuation arcs. They restrict themselves to projective dependency parsing; erroneous punctuation arcs do not lead to cascading errors in non-projective dependency parsing. Ma et al. (2014), motivated by the same observation, treat punctuation marks as properties of their neighboring words rather than as individual tokens, showing improvements on in-domain data.

Breaking NLP models Jia and Liang (2017) show how machine reading models can easily be broken with distractor sentences at test time

and propose an alternative evaluation scheme, and Belinkov and Bisk (2018) show how susceptible character-based machine translation models are to noise. Both papers are similar to ours in evaluating the performance of state-of-the-art models under corruptions of the data. There was recently a workshop dedicated to evaluation of NLP models under human adversarial example selection (Ettinger et al., 2017). Historically, NLP models were rarely evaluated on synthetic or otherwise adversarial data, but we believe this is a fruitful research direction. This is largely a philosophical question, and we believe a philosophical argument is in order. John Dewey (John Dewey, 1910), the American philosopher, distinguishes three modes of thinking: (i) common reasoning, which identifies pattern in available, historical data, (ii) empirical thinking, which collects new data to vary the experimental conditions, and (iii) experimental thinking, which actively modifies the conditions in controlled experiments to isolate the relevant variables. We believe recent work on breaking NLP models is an attempt to introduce experimental thinking into NLP, which has otherwise been limited – or *handicapped* in Dewey's words – by what data happens to be available.

6 Conclusions

We evaluate the sensitivity of five dependency parsers to variations in punctuation, showing that available neural parsers tend to be more sensitive to such variation. We also show, however, that training neural parsers without punctuation provides a robust model that is better than any off-the-shelf parsers.

References

Scott Baldwin and James Coady. 1978. Psycholinguistic approaches to a theory of punctuation. *Journal of Literacy Research*, 10(4):363–376.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and Natural Noise Both Break Neural Machine Translation.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750. Association for Computational Linguistics.

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M. Bender. 2017. Towards Linguistically Generalizable NLP Systems: A Workshop and Shared Task. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 1–10, Copenhagen, Denmark. Association for Computational Linguistics.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of ACL*, pages 1523–1533. Association for Computational Linguistics.

Amir Globerson and Sam Roweis. 2006. Nightmare at test time: robust learning by feature deletion. In *ICML*.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of EMNLP*.

Jason Jo and Yoshua Bengio. 2017. Measuring the tendency of CNNs to Learn Surface Statistical Regularities. *CoRR*, abs/1711.11561.

John Dewey. 1910. *How we think*. Dover.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. 4:313–327.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017a. From raw text to universal dependencies - look, no tags! pages 207–217, Vancouver, Canada.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-hybrid non-projective dependency parsing with a static-dynamic oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 99–104, Pisa, Italy.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014. Punctuation processing for projective dependency parsing. In *ACL*.

Marie-Catherine de Marneffe and Chris Manning. 2008. The Stanford typed dependencies representation. In *Coling Workshop on Cross-Framework and Cross-Domain Parser Evaluation*.

Dominick Ng and James Curran. 2015. Identifying cascading errors using constraints in dependency parsing. In *ACL*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Madeline Remse, Mohsen Mesgar, and Michael Strube. 2016. Feature-rich error detection in scientific writing using logistic regression. In *BEA*.

Valentin Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. 2011. Punctuation: Making a point in unsupervised dependency parsing. In *CoNLL*.