Pre-trained Contextualized Character Embeddings Lead to Major Improvements in Time Normalization: a Detailed Analysis

Dongfang Xu **Egoitz Laparra** **Steven Bethard**
School of Information
University of Arizona
Tucson, AZ
{dongfangxu9,laparra,bethard}@email.arizona.edu

Abstract

Recent studies have shown that pre-trained contextual word embeddings, which assign the same word different vectors in different contexts, improve performance in many tasks. But while contextual embeddings can also be trained at the character level, the effectiveness of such embeddings has not been studied. We derive character-level contextual embeddings from Flair (Akbik et al., 2018), and apply them to a time normalization task, yielding major performance improvements over the previous state-of-the-art: 51% error reduction in news and 33% in clinical notes. We analyze the sources of these improvements, and find that pre-trained contextual character embeddings are more robust to term variations, infrequent terms, and cross-domain changes. We also quantify the size of context that pre-trained contextual character embeddings take advantage of, and show that such embeddings capture features like part-of-speech and capitalization.

1 Introduction

Pre-trained language models (LMs) such as ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), OpenAI GPT (Radford et al., 2018), Flair (Akbik et al., 2018) and Bert (Devlin et al., 2018) have shown great improvements in NLP tasks ranging from sentiment analysis to named entity recognition to question answering. These models are trained on huge collections of unlabeled data and produce contextualized word embeddings, i.e., each word receives a different vector representation in each context, rather than a single common vector representation regardless of context as in word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014).

Research is ongoing to study these models and determine where their benefits are coming from (Peters et al., 2018; Radford et al., 2018; Khandelwal et al., 2018; Qi et al., 2018; Zhang and Bowman, 2018). The analyses have focused on word-level models, yet character-level models have been shown to outperform word-level models in some NLP tasks, such as text classification (Zhang et al., 2015), named entity recognition (Kuru et al., 2016), and time normalization (Laparra et al., 2018a). Thus, there is a need to study pre-trained contextualized *character* embeddings, to see if they also yield improvements, and if so, to analyze where those benefits are coming from.

All of the pre-trained word-level contextual embedding models include some character or subword components in their architecture. For example, Flair is a forward-backward LM trained over characters using recurrent neural networks (RNNs), that generates pre-trained contextual word embeddings by concatenating the forward LM's hidden state for the word's last character and the backward LM's hidden state for the word's first character. Flair achieves state-of-the-art or competitive results on part-of-speech tagging and named entity tagging (Akbik et al., 2018). Though they do not pre-train a LM, Bohnet et al. (2018) similarly apply a bidirectional long short term memory network (LSTM) layer on all characters of a sentence and generate contextual word embeddings by concatenating the forward and backward LSTM hidden states of the first and last character in each word. Together with other techniques, they achieve state-of-the-art performance on part-of-speech and morphological tagging. However, both Akbik et al. (2018) and Bohnet et al. (2018) discard all other contextual character embeddings, and no analyses of the models are performed at the character-level.

In the current paper, we derive pre-trained contextual character embeddings from Flair's forward-backward LM trained on a 1-billion word corpus of

*Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 68–74
Minneapolis, June 6–7, 2019.

English (Chelba et al., 2014), and observe if these embeddings yield the same large improvements for character-level tasks as yielded by pre-trained contextual word embeddings for word-level tasks. We aim to analyze where improvements come from (e.g., term variations, low frequency words) and what they depend on (e.g., embedding size, context size). We focus on the task of parsing time normalizations (Laparra et al., 2018b) , where large gains of character-level models over word-level models have been observed (Laparra et al., 2018a). This task involves finding and composing pieces of a time expression to infer time intervals, so for example, the expression *3 days ago* could be normalized to the interval *[2019-03-01, 2019-03-02)*.

We first take a state-of-the-art neural network for parsing time normalizations (Laparra et al., 2018a) and replace its randomly initialized character embeddings with pre-trained contextual character embeddings. After showing that this yields major performance improvements, we analyze the improvements to understand why pre-trained contextual character embeddings are so useful. Our contributions are:

- We derive pre-trained contextual character embeddings from Flair (Akbik et al., 2018), apply them to a state-of-the art time normalizer (Laparra et al., 2018a), and obtain major performance improvements over the previous state-of-the-art: 51% error reduction in news and 33% error reduction in clinical notes.
- We demonstrate that pre-trained contextual character embeddings are more robust to term variations, infrequent terms, and cross-domain changes.
- We quantify the amount of context leveraged by pre-trained contextual character embeddings.
- We show that pre-trained contextual character embeddings remove the need for features like part-of-speech and capitalization.

2 Framework

The parsing time normalizations task is based on the Semantically Compositional Annotation of Time Expressions (SCATE) schema (Bethard and Parker, 2016), in which times are annotated as compositional time entities. Laparra et al. (2018a) decomposes the Parsing Time Normalizations task into two subtasks: a) time entity identification using a character-level sequence tagger which detects

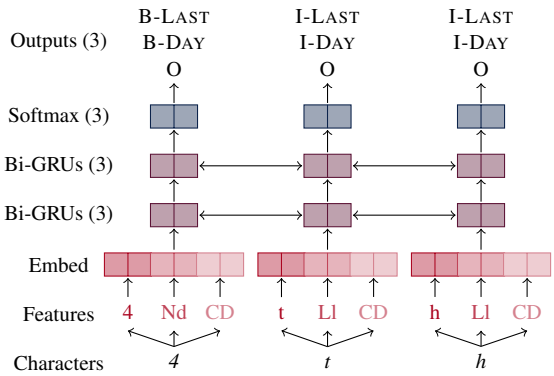

Figure 1: Architecture of Laparra et al. (2018a)'s time identification system. The input is *the 4th of May* (truncated for space). *4th* is a DAY-OF-MONTH, with an implicit LAST over the same span. At the feature layer, *4* is a digit (Nd), *t* and *h* are lowercase letters (Ll), and *4th* has the cardinal number (CD) part-of-speech tag.

the spans of characters that belong to each time expression and labels them with their corresponding time entity; and b) time entity composition using a simple set of rules that links relevant entities together while respecting the entity type constraints imposed by the SCATE schema. These two tasks are run sequentially using the predicted output of the sequence tagger as input to the rule-based time entity composition system. In this paper, We focus on the character-level time entity identifier that is the foundation of Laparra et al. (2018a)'s model.

The sequence tagger is a multi-output RNN with three different input features, shown in Figure 1. Features are mapped through an embedding layer, then fed into stacked bidirectional Gated Recurrent Units (bi-GRUs), and followed by a softmax layer. There are three types of outputs per Laparra et al. (2018a)'s encoding of the SCATE schema, so there is a separate stack of bi-GRUs and a softmax for each output type. We keep the original neural architecture and parameter settings in Laparra et al. (2018a), and experiment with the following embedding layers:

Rand(128): the original setting of Laparra et al. (2018a), where 128-dimensional character embeddings are randomly initialized.

Rand(4096): 4096-dimensional character embeddings are randomly initialized, matching the dimensionality of the Flair forward-backward LM hidden states, i.e., matching the dimensionality of Cont(4096).

Cont(4096): 4096-dimensional pre-trained contextual character embeddings are derived by run-

Model	Domain	Ident.	Parsing	Interv.
Rand(128)-ori	News	61.5	51.2	76.4
Rand(128)	News	59.4	50.5	64.6
Rand(4096)	News	64.8	54.1	68.2
Cont(4096)	News	**80.3**	**66.8**	**81.5**
Rand(128)-ori	Clinical	84.7	57.9	72.1
Rand(128)	Clinical	92.8	65.3	82.1
Rand(4096)	Clinical	93.2	65.3	83.8
Cont(4096)	Clinical	**95.2**	**67.3**	**85.8**

Table 1: Results on Identification (Ident.), Parsing and Interval extraction (Interv.) of time expressions for News and Clinical domain. Rand(128)-ori refers to the original implementation, and Rand(128) and Cont(4096) refer to our re-implementation[1].

ning Flair forward-backward character-level LM Flair's forward and backward character-level language models over the text, and concatenating the hidden states from forward and backward character-level LMs for each character .

We evaluate in the clinical and news domains, the former being more than 9 times larger and the latter having a more diverse set of labels. Three different evaluation metrics are used for parsing time normalization tasks: identification of time entities, which evaluates the predicted span (offsets) and the SCATE type for each entity; parsing of time entities, which evaluates the span, the SCATE type, and properties for each time entity; interval extraction, which interprets parsed annotations as intervals along the timeline and measures the fraction of the correctly parsed intervals. The SemeEval task description paper (Laparra et al., 2018b) has more details on dataset statistics and evaluation metrics.

3 Results

Table 1 shows that the model using pre-trained contextual character embeddings, Cont(4096), outperforms the model of Laparra et al. (2018a) on all three metrics: identification of time entities, parsing, and interval extraction. For identification, our primary focus as we are only modifying the identification portion of Laparra et al. (2018a), Cont(4096) reduces error by 51% (59.4 to 80.3 F_1) on news, and by 33% (92.8 to 95.2 F_1) on clinical notes. For the following experiments, we only use the identification metric to evaluate the performance.

[1]We upgraded Keras from 1.2 to 2.1 and fixed a code bug that allowed predictions to be made on padding tokens.

	Domain	Dev	Test
Rand(128)	News	76.5	59.4
Rand(4096)	News	82.7	64.8
Cont(4096)	News	**87.4**	**80.3**
Rand(128)	Clinical	92.9	92.8
Rand(4096)	Clinical	92.6	93.2
Cont(4096)	Clinical	**94.7**	**95.2**

Table 2: Performance (F_1) of time entity identification.

		News		Clinical	
		Dev	Test	Dev	Test
Variation	+var	**+8.4**	**+15.0**	+1.2	+1.3
	-var	+1.6	+8.7	+1.2	**+1.4**
Frequency	$\leq$10	**+8.1**	**+17.6**	**+2.0**	**+4.2**
	>10	+2.4	+5.0	+1.1	+1.1

Table 3: Effect of term variations and frequency: improvement in F_1 of Cont(4096) over Rand(4096).

4 Where the improvements come from

4.1 Larger character embeddings

Table 2 compares different embedding sizes. Moving from random 128-dimensional to random 4096-dimensional embeddings improves the model: Rand(4096) statistically outperforms[2] Rand(128) on news dev ($p = 0.0001$), news test ($p = 0.0291$), and clinical test ($p = 0.0301$), though it is not statistically different on clinical dev ($p = 0.2524$). Pre-trained contextual embeddings provide additional benefits: Cont(4096) significantly outperforms Rand(4096) on all datasets ($p < 0.001$ in all cases). We conclude that pre-trained contextual character embeddings provide more than just greater model capacity.

4.2 Robustness to variants and frequency

Table 3 shows how pre-trained contextual character embeddings improve performance on both **term variations** and **low frequency words**.

We define **term variations** as time entities that appear in the training data in the following patterns: both upper-case and lower-case, e.g., *DAY*, *Day*, and *day*; abbreviation with and without punctuation, e.g., *AM* and *A.M.*; or same stem, e.g., *Month* and *Months*, *previously* and *previous*. In the dev and test sets, 30.4-35.6% of entities are term variations. The first 2 rows of table 3 show the performance improvements in F_1 of Cont(4096)

[2]We used a paired bootstrap resampling significance test.

	Train	Target	Dev	Test
Rand(128)	Clinical	News	63.4	65.5
Rand(4096)	Clinical	News	62.6	66.9
Cont(4096)	Clinical	News	**68.3**	**78.5**
Rand(128)	News	Clinical	45.3	46.3
Rand(4096)	News	Clinical	43.8	44.3
Cont(4096)	News	Clinical	**57.1**	**59.5**

Table 4: Effect of domain change on performance: (F_1) on News and Clinical datasets.

over Rand(4096) on time entities with (+var) and without (-var) term variations. Cont(4096) is always better than Rand(4096) so all differences are positive, but the improvements in +var are much larger than those of -var in the news domain (+8.4 vs. +1.6 and +15.0 vs. +8.7). In the clinical domain, where 9 times more training data is available, both +var and -var yield similar improvements. We conclude that pre-trained contextual character embeddings are mostly helpful with term variations in low data scenarios.

We define **infrequent terms** as time entities that occur in the training set 10 or fewer times. In the dev and test sets, 73.9-86.9% of terms are infrequent, with about one third of infrequent terms being numerical[3]. The bottom two rows of table 3 show the improvements in F_1 of the Cont(4096) over Rand(4096) on frequent ($>$10) and infrequent ($\leq$10) terms. Cont(4096) is always better than Rand(4096), and in both domains the improvements on low frequency terms are always greater than those on high frequency terms (+8.1 vs. +2.4 in news dev, +17.6 vs. +5.0 in news test, etc.). We conclude that pre-trained contextual character embeddings improve the representations of low frequency words in both low and high data settings.

4.3 Robustness to domain differences

To illustrate the ability of pre-trained contextual character embeddings to handle unseen data, we train the models in one domain and evaluate in the other, as shown in Table 4. We find that Rand(128) and Rand(4096) achieve similar cross-domain performance, e.g., Rand(128) achieves 63.4% of F_1 on news dev and Rand(4096) achieves 62.6% F_1. But Cont(4096) achieves much better cross-domain performance than Rand(128) or Rand(4096): 78.5% vs. 65.5% or 66.9% F_1 on news test, 59.5% vs. 46.3% or 44.3% on clinical test, etc. All these improvements are significant ($p < 0.001$). We conclude that pre-trained contextual character embeddings generalize better across domains.

4.4 Greater reliance on nearby context

Inspired by Khandelwal et al. (2018)'s analysis of the effective context size of a word-based language model, we present an ablation study to measure performance when contextual information is removed. Specifically, when evaluating models, we retain only the characters in a small window around each time entity in the dev and test sets, and replace all other characters with padding characters.

Figures 2a and 2b evaluate the Cont(4096), Rand(4096) and Rand(128) models across different context window sizes on the news dev and test set, respectively. Rand(128) performs similarly across all context sizes, suggesting that it makes little use of context information. Both Rand(4096) and Cont(4096) depend heavily of context: without any context information (context size 0), they perform worse than Rand(128). Cont(4096) is sensitive to the nearby context, with a $\sim$10 point gain on news dev and $\sim$15 point gain on news test from just the first 10 characters of context, putting it easily above Rand(128). Rand(4096) doesn't exceed the performance of Rand(128) until at least 50 characters of context.

Figures 2c and 2d shows similar trends in the clinical domain, except that the Rand(128) model now shows a small dependence on context, with a $\sim$5 point drop on clinical dev and a $\sim$3 drop on clinical test in the no-context setting. Cont(4096) again makes large improvements in just the first 10 characters, and Rand(4096) now takes close to 100 characters of context to reach the performance of Rand(128). We conclude that pre-trained contextual character embeddings make better use of local context, especially within the first 10 characters.

4.5 Encoding word categories

We perform a feature ablation to see if pre-trained contextual character embeddings capture basic syntax (e.g., part-of-speech) like pre-trained contextual word embeddings do (Peters et al., 2018; Akbik et al., 2018). Table 5 shows that removing both part-of-speech and unicode category features from Cont(4096) does not significantly change performance: news dev ($p = 0.8813$), news test ($p = 0.1672$), clinical dev ($p = 0.5367$), clinical test ($p = 0.8537$). But ablating part-of-speech tags and unicode character categories does decrease per-

[3]Numbers are common in time expressions.

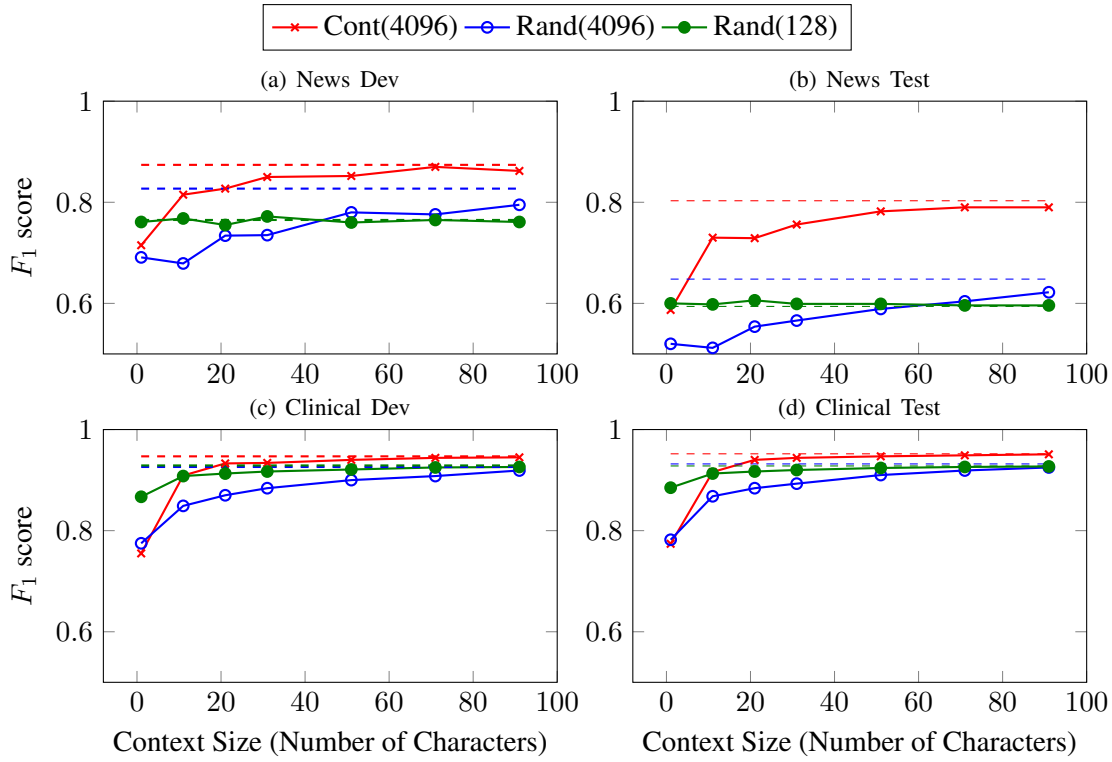

Figure 2: Effect of the context information on the performances for Cont(4096), Rand(4096) and Rand(128) on the dev and test sets. The dashed lines are the performances of models using the original context setting.

		News		Clinical	
	Set	Dev	Test	Dev	Test
Rand(128)	C	73.6	56.1	91.9	92.1
Rand(128)	CUP	76.5	59.4	92.9	92.8
Rand(4096)	C	80.5	62.4	91.7	92.2
Rand(4096)	CUP	82.7	64.8	92.6	93.2
Cont(4096)	C	87.9	78.1	94.7	95.5
Cont(4096)	CUP	87.4	80.3	94.7	95.2

Table 5: Effect of features on performance: Performance (F_1) with different feature sets, including characters (C), part-of-speech tags (P), and unicode character categories (U).

formance for both Rand(128) and Rand(4096) in all cases. For example, Rand(4096) with all features achieves 82.7 F_1 on news dev, significantly better than the 80.5 F_1 of using only characters ($p = 0.0467$). We conclude that pre-trained contextual character embeddings encode a variety of word category information such as part-of-speech, capitalization, and punctuation.

5 Conclusion

We derive pre-trained character-level contextual embeddings from Flair (Akbik et al., 2018), a word-level embedding model, inject these into a state-of-the-art time normalization system, and achieve major performance improvements: 51% error reduction in news and 33% in clinical notes. Our detailed analysis concludes that pre-trained contextual character embeddings are more robust to term variations, infrequent terms, and cross-domain changes; that they benefit most from the first 10 characters of context; and that they encode part-of-speech, capitalization, and punctuation information.

6 Acknowledgements

We thank the anonymous reviewers for helpful comments on an earlier draft of this paper. This work was supported by National Institutes of Health grants R01GM114355 from the National Institute of General Medical Sciences (NIGMS) and R01LM012918 from the National Library of Medicine (NLM). The computations were done in systems supported by the National Science Foundation under Grant No. 1228509. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health or National Science Foundation.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Steven Bethard and Jonathan Parker. 2016. A semantically compositional annotation scheme for time normalization. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294. Association for Computational Linguistics.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921.

Egoitz Laparra, Dongfang Xu, and Steven Bethard. 2018a. From characters to time intervals: New paradigms for evaluation and neural parsing of time normalizations. *Transactions of the Association of Computational Linguistics*, 6:343–356.

Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018b. Semeval 2018 task 6: Parsing time normalizations. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 88–96.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 529–535.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

A Appendices

A.1 Examples of the improvement

We analyzed a few examples where Cont(4096) makes correct predictions, but Rand(4096) does not.

Robustness to variants

"... with year-earlier profit of millions..."

In this sentence, the Cont(4096) model labeled *earlier* correctly, while the Rand(4096) model missed it. In the news training set, *earlier* occurs a few times, but none of them have "-" nearby.

Robustness to frequency

"... in the first days after President..."

In this sentence, the Cont(4096) model labeled *first* correctly, while the Rand(4096) model labeled it incorrectly. In the news training set, *first* only occurred once when followed by another time entity, but there were several similar sentences for *second* and *third* in the training set.

Robustness to word order

"... until twenty years after the first astronauts..."

"... comes barely a month after Qantas..."

"... Retaliating 13 days after the deadly..."

In each of the sentences above, the Cont(4096) model labeled *after* correctly, while Rand(4096) labeled it incorrectly. In the training set, there were a few examples where *after* occurred near a time entity, but always before the time entity (e.g., *after ten years*, *after 22 months*, *after three days*, *after a 16-hour flight*) rather than after it as in the examples above. Cont(4096) may have learned a better representation for *after* that allows it to be less dependent on exact word order.