The Importance of Character-Level Information in an Event Detection Model*

Emanuela Boros[1][0000−0001−6299−9452], Romaric Besançon[2][0000−0003−1331−5768], Olivier Ferret[2][0000−0003−0755−2361], and Brigitte Grau[3]

[1] University of La Rochelle, L3i, F-17000, La Rochelle, France
emanuela.boros@univ-lr.fr
[2] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
{romaric.besancon,olivier.ferret}@cea.fr
[3] Université Paris-Saclay, CNRS, LIMSI, ENSIIE, F-91405, Orsay, France
brigitte.grau@limsi.fr

Abstract. This paper tackles the task of event detection that aims at identifying and categorizing event mentions in texts. One of the difficulties of this task is the problem of event mentions corresponding to misspelled, custom, or out-of-vocabulary words. To analyze the impact of character-level features, we propose to integrate character embeddings, that can capture morphological and shape information about words, to a convolutional model for event detection. More precisely, we evaluate two strategies for performing such integration and show that a late fusion approach outperforms both an early fusion approach and models integrating character or subword information such as ELMo or BERT.

Keywords: Information extraction · Events · Word embeddings.

1 Introduction

In this article, we concentrate more specifically on event detection, which implies identifying instances of specified types of events in a text. The notion of event in our work is classically defined as something that happens and covers a wide spectrum, from terrorist attacks to births or nominations. The instances of these events in texts, which are called *event mentions* or *event triggers*, are annotated as words or phrases that evoke a reference type of events. The most successful approaches developed for achieving this task are currently based on neural models, which have been intensively studied to overcome fundamental limitations, specifically the complex choice of features [2, 27, 25, 26, 9, 36, 29]. All these proposed models based on Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or even Graph Neural Networks (GNNs) rely on word embeddings, a general distributed word representation that is produced by training a deep learning model on a large unlabeled dataset. Consequently, word

* This work was partly supported by the European Union's Horizon 2020 research and innovation program under grants 770299 (NewsEye) and 825153 (Embeddia).

embeddings replace the hard matches of words in the feature-based approaches with the soft matches of continuous word vectors. Hence, compared to previous rule-based or machine learning-based approaches, neural models are supposed to be less sensitive to the problem of unseen triggers since the distributed representations of words they exploit can account for the similarity between words.

However, this capacity may vary depending on the reasons why a trigger was not seen during the training of a model. We illustrate these different cases on the ACE 2005 dataset[4], a standard corpus used for evaluating event detection. An unseen trigger may be a morphological variant of a trigger already seen in the training set. For instance, *torturing* is not present in the training data but is a variant of *torture* and can be considered as a trigger for the same type of events, namely *Life.Injure*. Moreover, *torturing* is likely to be present among general pre-trained word embeddings and if so, a neural event extraction model is likely to successfully detect this trigger. The situation may be different when a trigger is absent from the training data because it corresponds to a misspelled version of a reference trigger. For instance, *aquitted* is part of the ACE 2005 test dataset for referring to a *Justice.Sentence* event while only *acquitted*, the correct form for that word, is present in the training data. In that case, we cannot assume that the unseen word is part of general word embeddings and as a consequence, has little chance to be detected as a trigger for a *Justice.Sentence* event.

From a more general perspective, the problem of missing word embeddings when using pre-trained models in the context of event extraction is not marginal. The ACE 2005 dataset, for instance, covers the most common events of national and international news (from a variety of sources selected from broadcast news programs, newspapers, news reports, internet sources, or transcribed audio) and thus, it contains different types of discourse, professional or noisy discussions prone to the presence of mistakes in spelling and custom words. As a result, in this dataset, 14.8% of the words are not part of the pre-trained embeddings provided by Google, trained with *word2vec* on *Google News* [23], 1.5% for the *GloVe* embeddings [30], and 4.5% for the fastText embeddings [12]. Different strategies were proposed and implemented for dealing with the issue of missing words in neural language models. For static word embeddings, fastText relies on a representation of words based on n-grams of characters. For contextual models, ELMo [31] exploits a character-based representation built with a CNN while BERT [3] adopts a mixed strategy based on subwords, called wordpieces, where a word is split into subwords when it is not part of a predefined and restricted vocabulary [21, 15, 13]. However, while BERT seems to be an interesting option for a large number of tasks in Natural Language Processing, its ability to handle noisy inputs is still an open question [32] or at least requires the addition of complementary methods [24]. This limitation may result from the dependence of BERT on a vocabulary. The alternative is to rely, as ELMo for instance, on a character model in which all words, including words with abnormal character combinations and misspellings, are processed similarly.

[4] https://catalog.ldc.upenn.edu/ldc2006t06.

Some researchers studied the application of CNNs to characters. For anti-spam filtering, the use of character-level n-grams was already experimented out of the context of deep learning models [14]. In [5], character-level embeddings were automatically learned and joined with pre-trained word embeddings in a CNN-based model for Part-of-Speech tagging. This architecture was also used for improving the performance of a Named Entity Recognition (NER) system in [4]. While character models have been used with success in several contexts for tackling the absence of pre-trained embeddings for all words, the use of CNNs to learn directly from characters was also investigated, without the need for any pre-trained embeddings [37]. Notably, the authors use a relatively deep network and apply it to sentiment analysis and text classification tasks. The application of character-level convolutions to language modeling was explored in [15] by using the output of a character-level CNN as the input to a Long Short Term Memory (LSTM) network at each time step. The same model is easily applied to various languages. However, the choice of CNN-based or LSTM-based character-level word embeddings did not affect the performance significantly [16, 22].

Our contributions in this article are more particularly focused on the integration of character-level features in event detection models for addressing the issue of unknown words. More specifically, we show that an event detection model exploiting a character-based representation is complementary to a word-oriented model and that their combination according to a late fusion approach outperforms an early fusion strategy.

2 Related Work

The current state-of-the-art systems for event extraction involve neural network models to improve event extraction. [27] and [2] deal with the event detection problem with models based on CNN. [28] improve the previous CNN models of [27] for event detection, slightly modifying the way CNNs are applied to sentences by taking into account the possibility to have non-consecutive n-grams as basic features instead of continuous n-grams. Both models use word embeddings for representing windows of text that are trained as the other parameters of the neural network.

The authors of [25] predict at the same time event triggers and their arguments in a joint framework with Bidirectional RNNs (Bi-RNNs) and a CNN and systematically investigate the usage of memory vectors/matrices to store the prediction information during the labeling of sentence features. Additionally, the authors augment their system with discrete local features inherited from [17].

A GNN is advocated in [29] based on dependency trees to perform event detection with a pooling method that relies on entity mentions aggregating the convolution vectors. The authors of [20] consider also that arguments provide significant clues to this task and adopt a supervised attention mechanism to exploit argument information explicitly for event detection, while also using events from FrameNet.

Table 1. Statistics about unknown words in the ACE 2005 dataset.

	All words	**Trigger words**
train	14,021	931
test	3,553	219
unknown words in test data	930 (26.2%)	66 (30.1%)
unknown words with a known similar word	825	54

Further, some researchers have proposed other hybrid neural network models with different types of pre-set word embeddings that combine different neural networks to make use of each other's abilities. A hybrid neural network (a CNN and an RNN) [9] was developed to capture both sequence and chunk information from specific contexts and use them to train an event detector for multiple languages without any handcrafted features.

Some authors went beyond sentence-level sequential modeling, considering that these methods suffer from low efficiency in capturing very long-range dependencies. An approach that goes beyond sentence level [8] was proposed by using a document representation using an RNN model that can automatically extract cross-sentence clues.

Recently, different approaches that include external resources and features at a subword representation level have been proposed. For example, Generative Adversarial Networks (GANs) framework has been applied in event extraction [36, 10]. Besides, reinforcement learning is used in [36] for creating an end-to-end entity and event extraction framework. An approach based on the BERT pre-trained model [35] attempts an automatic generation of labeled data by editing prototypes and filtering out the labeled samples through argument replacement by ranking their quality.

The problem of ambiguous indicators for particular types of events, i.e., the same word can express completely different events, such as *fired*, that can correspond to an *Attack* type of event or can express the dismissal of an employee from a job, is approached in [19] by using an RNN and cross-lingual attention to model the confidence of the features provided by other languages.

3 Motivation

Learning word representations from a corpus (word embeddings) allows us to derive a flexible similarity between words that takes into account a form of synonymy or relatedness between the words into the model. A drawback of this kind of representation is that unknown words (i.e. words unseen in the training corpus) are not well represented: they are generally associated with a random embedding even if these words are morphologically close to known words. Existing embeddings trained on very large collections of text, such as *word2vec* embeddings, which have proven their efficiency as initial embeddings for event extraction, do not take into account these morphological similarities: no lemmatization or stemming or even case normalization is performed.

We present in Table 1 some statistics about unknown words on the dataset we will use for training and testing our proposed approach, the ACE 2005 dataset,

Table 2. Examples of unknown words focused on triggers.

Event Type	**Unknown/Closest Trigger Words**
Start-Org	*creating/creation, opening/open, forging/forming, formed/form*
End-Org	*crumbled/crumbling, dismantling/dismantle, dissolved/dissolving*
Transport	*fleeing/flying, deployment/deployed, evacuating/evacuated*
Attack	*intifada/Intifada, smash/smashed, hacked/attacked, wiped/wipe*
End-Position	*retirement/retire, steps/step, previously/previous, formerly/former*

using the standard training/validation/test split [11]. We report the size of the vocabulary for the whole dataset, the size of the vocabulary for the trigger words, the number of words in the test dataset not seen in the training dataset, and among those, the number of words for which a similar word (measured by a Levenshtein ratio of less than 0.3) can be found.

We see that there is an important number of words, even among the trigger words, that cannot be exploited by the models because they are not seen in the training corpus. Also, most of these words (more than 88%) have similar words in the training corpus which could be used to approximate their representation, as they are likely to be semantically close words. To illustrate this, we show in Table 2 examples of unknown words focused on triggers. The examples are pairs of words used as triggers in the test set associated with their closest trigger words (for an event of the same type) as seen in the training set (with a distance of Levenshtein ratio less than 0.3). We can see in this table that most of the pairs correspond to derivational morphology links. These semantic links are lost with the standard embedding models.

The integration of a character-based embedding model should be able to help in dealing with such cases by allowing to bridge the gap between the unknown words and representations of known words used for training the system. The same problem occurs with infrequent words, that could be better represented if they are processed at a character level.

4 Approach

Our approach lies in the standard supervised framework of event detection where the task is modeled as a word classification task: considering a sentence, we want to predict for each word of the sentence if it is an event trigger and associate it with its event type. The input of the system is therefore a target word in the context of a sentence and the output an event type or NONE for non-trigger words. To study the influence of character-based features, we rely on the CNN model proposed by [27] as a core model. This core architecture is used in the two components of our overall system: the Word model and the Character model. These two components are combined using either an early fusion approach or a late fusion approach, as illustrated by Figure 1.

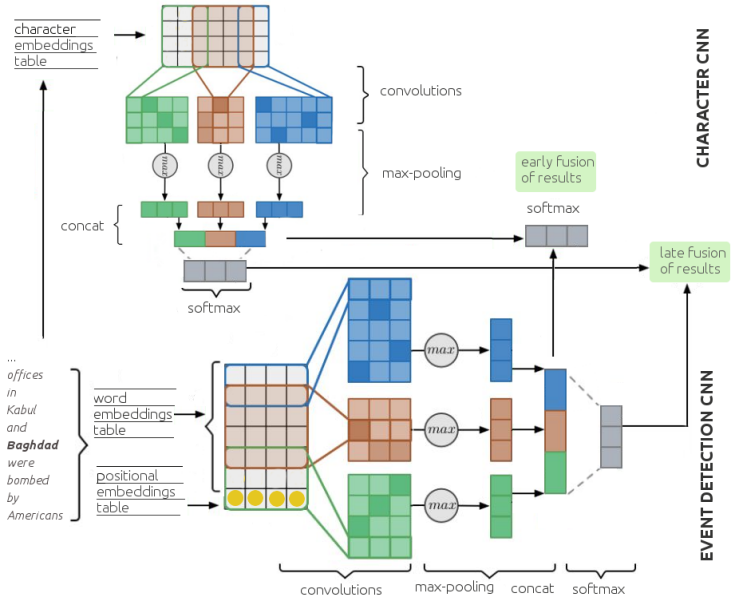

Fig. 1. Word + Character CNN.

4.1 Word and Character CNN Models

In the Word CNN model, the context of a target word is formed by its surrounding words in the sentence, which constitutes the input of the convolution layer. To consider a limited-sized context, longer sentences are trimmed and shorter ones are zero-padded. We consider a context window for every trigger candidate where each token is associated with a word embedding and a relative position to the trigger candidate embedding. The word and position embeddings are concatenated and passed through the convolution layer. The concatenated output of convolutional filter maps forms, after a max-over-time pooling operation on each one, the representation of the input (target token and context) that finally goes through a softmax classification layer. The Character CNN model is very close to the Word CNN model, with two main differences: words are replaced by characters and there is no position embedding associated with each character.

4.2 Integration of Word and Character Models

Early Fusion The first type of integration is the early fusion model, in which the two representations of the input sequence produced by the Word and Character CNNs (i.e., the concatenation of the output vectors of their filters) are concatenated before the fully-connected softmax classification layer. Using this type of integration allows joint learning of the parameters of the two models in the training phase.

Late Fusion The late fusion integration of the Word and Character CNNs relies on the combination of the decisions of the two models, which are trained separately and therefore learn different characteristics of the candidate trigger. Indeed, the word-level CNN combines word and position embeddings that can capture syntactic and semantic information, and of course, the relative positions of words to the candidate trigger. The character-level CNN learns more local features from character n-grams and can capture morphological information. The late fusion focuses on the individual strength of these two models by applying the following rule: we always keep the Character CNN label, except if a trigger was detected by the Word CNN but not by the Character CNN. This strategy is motivated by the fact that the Word CNN model has good coverage whereas the Character model is more focused on precision.

5 Experiments and Results

5.1 Dataset

The evaluation is conducted on the annotated ACE 2005 corpus. We use the same split as previous studies with this dataset [27, 25]: 40 news articles (672 sentences) for the test set, 30 other documents (863 sentences) for the development set, and the remaining 529 documents (14,849 sentences) for the training set. Following the same line of work, we consider that a trigger is correct if its event type, subtype, and offsets match those of a reference trigger. We use Precision (P), Recall (R), and F-measure (F1) to evaluate the overall performance.

5.2 Hyperparameters

For the Word CNN, we consider a sliding window with a maximal size of 31 words. Hence, sentences are padded at the beginning and the end with a vector of 15 zeros (a common practice for the padding special character). The window sizes for the convolutions are in the set $\{1, 2, 3\}$ and 300 feature maps are used for each window size. After each convolutional layer with orthogonal weights initialization, a *ReLu* non-linear layer is applied. We employ dropout with a probability of 0.5 after the embedded window of text since they contain most of the parameters and as a consequence, the possibility of being responsible for overfitting. A dropout of 0.3 is also applied after the concatenation of the convolutions. The size of the position embeddings is equal to 50, similarly to [27]. We use the Google News word embeddings pre-trained with *word2vec* (size = 300).

For the Character CNN, we consider a maximum length of 1,024 for a sequence of characters: longer sequences are trimmed and shorter ones are padded with zeros. The window sizes for the convolutions are in the set $\{2, \ldots, 10\}$, with 300 feature maps. The convolutional layer non-linearity and initialization are the same as for the Word CNN. The size of the character embeddings is 300. These embeddings are initialized based on a normal distribution and trained on the

event detection task. A dropout of 0.5 is applied after the embedded characters. When jointly trained, in the early fusion model, the features obtained after convolutions from both models are concatenated and, similarly to the Word CNN, a dropout of 0.3 is applied afterward, before the softmax layer. We encode all the characters except space.

We train both networks (Word and Character CNNs) with Adam optimizer. During the training, we optimize the embedding tables (i.e., word, position, and character embeddings) to achieve the optimal states. Finally, for training, we use a batch size of 256 for the Word CNN and 128 for the Character CNN. When they are trained jointly in the early fusion model, we use a batch size of 128. All these hyperparameters were optimized by a grid search on the development set.

5.3 Results

We compare our model with several neural-based models proposed for the same task that do not use external resources, namely: a set of CNN-based models including a CNN model without any additional features [27], the dynamic multi-pooling CNN model of [2], the non-consecutive CNN of [26], and the Graph CNN proposed by [29]; a set of RNN-based models, represented by the bidirectional joint RNN model of [25], the DLRNN model of [8] and the DEEB-RNN model of [38] that both rely on a document representation, and the work of [19], based on a Gated Cross-Lingual Attention mechanism. Our reference models also include the hybrid model proposed by [9], the model exploiting arguments through an attention mechanism of [20], and the GAIL-ELMo model of [36], based on GANs. We do not consider models that are using other external resources such as [1], [18], or [35], since we only rely on the input text in our model. We also compare this model with four baselines based on the BERT language model, applied in a similar way to [3] for the NER task, with the recommended hyperparameters: a learning rate of 2e-5 and the split of sentences into chunks of 128 tokens.

The best performance (75.8 F1 on the test set) is achieved by combining word and position embeddings with the character-level features using a late fusion strategy. This performance relates to improvements that have been reported on other tasks when concatenating word embeddings with the output from a character-level CNN for Part-of-Speech tagging [6] and NER [4]. From Table 3, we can also outline that adding character embeddings in a late fusion strategy outperforms all the word-based models, including complex architectures such as the graph CNN and the models based on the BERT language model. Among BERT models, it is worth noticing that the `cased` models perform better than the `uncased` ones, which confirms that the character morphology is important for the task, maybe because capitalization is connected to the recognition of named entities, which are usually considered important to detect event mentions.

However, we can see that the character embeddings are not sufficient on their own: using only the Character CNN leads to the smallest recall among all the considered approaches. However, its precision is high (71.7), which makes this model fairly reliable about the triggers it retrieves. Given this observation, we can compare the two integration strategies, early and late fusions. In the case of

Table 3. Evaluation of our models and comparison with state-of-the-art systems for event detection on the blind test data. †beyond sentence level, +with gold arguments.

Approaches	**Precision**	**Recall**	**F1**
Word CNN [27] (without entities)	71.9	63.8	67.6
Dynamic multi-pooling CNN [2]	75.6	63.6	69.1
Joint RNN [25]	66.0	73.0	69.3
RNN with document context† [8]	77.2	64.9	70.5
Non-Consecutive CNN [26]	na	na	71.3
Attention-based+ [20]	78.0	66.3	71.7
GAIL-ELMo [36]	74.8	69.4	72.0
Gated Cross-Lingual Attention [19]	78.9	66.9	72.4
Graph CNN [29]	77.9	68.8	73.1
Hybrid NN [9]	84.6	64.9	73.4
DEEB-RNN3† [38]	72.3	75.8	74.0
BERT-base-uncased + LSTM [33]	na	na	68.9
BERT-base-uncased [33]	na	na	69.7
BERT-base-uncased [7]	67.2	73.2	70.0
BERT-QA [7]	71.1	73.7	72.4
DMBERT [34]	77.6	71.8	74.6
DMBERT+Boot [34]	77.9	72.5	75.1
BERT-base-uncased	71.7	68.5	70.0
BERT-base-cased	71.3	72.0	71.7
BERT-large-uncased	72.1	72.9	72.5
BERT-large-cased	69.3	**77.2**	73.1
Word CNN (replicated)	71.4	65.9	68.5
Character CNN	71.7	41.2	52.3
Word + Character CNN - early fusion	**88.6**	61.9	72.9
Word + Character CNN - late fusion	87.2	67.1	**75.8**

early fusion, where the two models are trained jointly, we notice that the precision is the highest among all the compared models. We assume that in the joint approach, the power of representation of morphological properties provided by the characters is overtaking the influence of the word and positions embedding, and the combination reproduces the imbalance between precision and recall observed for the Character CNN, the recall being the lowest among all the models except the *Character CNN*. In the case of the late fusion, since we have more control over the combination and we can give priority to the Character CNN to establish the labels on the trigger candidates retrieved by the Word CNN, the method takes advantage of the high precision of the Character CNN, allowing an increase of the precision from 71.7 to 87.2, while still having a high recall, also increasing the recall of the *Word CNN* model from 65.9 to 67.1. The late fusion integration is therefore able to take into account the complementarities of the two models.

Finally, for more qualitative analysis, we examine the new triggers correctly detected by the Word + Character CNN (late fusion), in comparison with the Word CNN. We observe that among the 37 new correctly found triggers, some are

Table 4. Examples of new triggers found with the Word+Character CNN (late fusion).

Event Type	**New triggers correctly found**	**Trigger words in training data**
End-Position	*steps*	*step*
Extradite	*extradited*	*extradition*
Attack	*wiped*	*wipe*
Start-Org	*creating*	*create*
Attack	*smash*	*smashed*
End-Position	*retirement*	*retire*

indeed derivational or inflectional variants of known words in the training data, such as illustrated in Table 4. This seems to confirm that the character-based model can capture some semantic information associated with morphological characteristics of the words and manage to detect new correct event mentions that correspond to inflections of known event triggers (i.e., existing in the training data). Also, the fact that the convolution windows in the Character CNN range from 2 to 10 means that character n-grams in the same range are included in the model and contribute to the model's ability to handle different word variations.

6 Conclusion and Perspectives

We have proposed in this article a study of the integration of character embeddings in an event detection neural-based model using a simple CNN model as core architecture and testing early and late fusion strategies to integrate the character-based features. The best results are achieved by combining the word-based features with the character-based features in a late fusion strategy that gives priority to the Character CNN for deciding the event type. This method outperforms more complex approaches such as Graph CNN or adversarial networks and BERT-based models. Our results demonstrate that a convolutional approach for learning character-level features can be successfully applied to event detection and that these features allow overcoming some issues concerning unseen or misspelled words in the test data.

We do not integrate the character information at the embedding level as it is usually done in models considering smaller units such as ELMO with characters, FastText with character n-grams, or BERT with subwords. In a certain way, they implement another kind of early fusion than ours. However, our late fusion approach is complementary and as a perspective, we consider implementing this late fusion framework using more complex models as core models. Another way to deal with the problem of unseen words would be to exploit data augmentation strategies that would focus on increasing the variability about derivational and inflectional variants of event mentions in the training data.

References

1. Bronstein, O., Dagan, I., Li, Q., Ji, H., Frank, A.: Seed-Based Event Trigger Labeling: How far can event descriptions get us? In: ACL-IJCNLP. pp. 372–376 (2015)
2. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J.: Event extraction via dynamic multi-pooling convolutional neural networks. In: ACL-IJCNLP 2015. pp. 167–176 (2015)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: NAACL-HLT 2019. pp. 4171–4186 (2019)
4. Dos Santos, C., Guimarães, V.: Boosting Named Entity Recognition with Neural Character Embeddings. In: Fifth Named Entity Workshop. pp. 25–33 (2015)
5. Dos Santos, C., Zadrozny, B.: Learning character-level representations for part-of-speech tagging. In: 31st International Conference on Machine Learning (ICML-14). pp. 1818–1826 (2014)
6. Dos Santos, C.N., Gatti, M.: Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In: COLING. pp. 69–78 (2014)
7. Du, X., Cardie, C.: Event Extraction by Answering (Almost) Natural Questions. In: 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 671–683 (2020)
8. Duan, S., He, R., Zhao, W.: Exploiting Document Level Information to Improve Event Detection via Recurrent Neural Networks. In: Eighth International Joint Conference on Natural Language Processing (IJCNLP 2017). pp. 352–361 (2017)
9. Feng, X., Huang, L., Tang, D., Ji, H., Qin, B., Liu, T.: A language-independent neural network for event detection. In: 54th Annual Meeting of the Association for Computational Linguistics. pp. 66–71 (2016)
10. Hong, Y., Zhou, W., Zhang, J., Zhou, G., Zhu, Q.: Self-regulation: Employing a generative adversarial network to improve event detection. In: 56th Annual Meeting of the Association for Computational Linguistics. pp. 515–526 (2018)
11. Ji, H., Grishman, R., et al.: Refining Event Extraction through Cross-Document Inference. In: ACL. pp. 254–262 (2008)
12. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification. In: 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017). pp. 427–431 (2017)
13. Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., Wu, Y.: Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410 (2016)
14. Kanaris, I., Kanaris, K., Houvardas, I., Stamatatos, E.: Words versus character n-grams for anti-spam filtering. International Journal on Artificial Intelligence Tools **16**(06), 1047–1067 (2007)
15. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-Aware Neural Language Models. In: Thirtieth AAAI Conference on Artificial Intelligence. pp. 2741–2749 (2016)
16. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural Architectures for Named Entity Recognition. In: NAACL-HLT 2016. pp. 260–270 (2016)
17. Li, Q., Ji, H., Huang, L.: Joint Event Extraction via Structured Prediction with Global Features. In: ACL. pp. 73–82 (2013)
18. Li, W., Cheng, D., He, L., Wang, Y., Jin, X.: Joint event extraction based on hierarchical event schemas from FrameNet. IEEE Access **7**, 25001–25015 (2019)
19. Liu, J., Chen, Y., Liu, K., Zhao, J.: Event Detection via Gated Multilingual Attention Mechanism. In: Thirty-second AAAI Conference on Artificial Intelligence (AAAI-18) (2018)

20. Liu, S., Chen, Y., Liu, K., Zhao, J.: Exploiting Argument Information to Improve Event Detection via Supervised Attention Mechanisms. In: 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017). pp. 1789–1798 (2017)
21. Luong, M.T., Manning, C.D.: Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. In: 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016). pp. 1054–1063 (2016)
22. Ma, X., Hovy, E.: End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In: 54th Annual Meeting of the Association for Computational Linguistics. pp. 1064–1074 (2016)
23. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: International Conference on Learning Representations (ICLR 2013), workshop track (2013)
24. Muller, B., Sagot, B., Seddah, D.: Enhancing BERT for Lexical Normalization. In: 5th Workshop on Noisy User-generated Text (W-NUT 2019). pp. 297–306 (2019)
25. Nguyen, T.H., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In: NAACL-HLT. pp. 300–309 (2016)
26. Nguyen, T.H., Fu, L., Cho, K., Grishman, R.: A two-stage approach for extending event detection to new types via neural networks. ACL 2016 p. 158 (2016)
27. Nguyen, T.H., Grishman, R.: Event Detection and Domain Adaptation with Convolutional Neural Networks. In: ACL-IJCNLP 2015. pp. 365–371 (2015)
28. Nguyen, T.H., Grishman, R.: Modeling Skip-Grams for Event Detection with Convolutional Neural Networks. In: EMNLP (2016)
29. Nguyen, T.H., Grishman, R.: Graph Convolutional Networks With Argument-Aware Pooling for Event Detection. In: Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018) (2018)
30. Pennington, J., Socher, R., Manning, C.D.: Glove: Global Vectors for Word Representation. In: EMNLP. pp. 1532–1543 (2014)
31. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep Contextualized Word Representations. In: NAACL-HLT 2018. pp. 2227–2237 (2018)
32. Sun, L., Hashimoto, K., Yin, W., Asai, A., Li, J., Yu, P., Xiong, C.: Adv-BERT: BERT is not robust on misspellings! Generating nature adversarial samples on BERT. arXiv preprint arXiv:2003.04985 (2020)
33. Wadden, D., Wennberg, U., Luan, Y., Hajishirzi, H.: Entity, Relation, and Event Extraction with Contextualized Span Representations. In: EMNLP-IJCNLP 2019. pp. 5784–5789 (2019)
34. Wang, X., Han, X., Liu, Z., Sun, M., Li, P.: Adversarial training for weakly supervised event detection. In: NAACL-HLT 2019. pp. 998–1008 (2019)
35. Yang, S., Feng, D., Qiao, L., Kan, Z., Li, D.: Exploring Pre-trained Language Models for Event Extraction and Generation. In: 57th Annual Meeting of the Association for Computational Linguistics. pp. 5284–5294 (2019)
36. Zhang, T., Ji, H., Sil, A.: Joint entity and event extraction with generative adversarial imitation learning. Data Intelligence **1**(2), 99–120 (2019)
37. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in neural information processing systems. pp. 649–657 (2015)
38. Zhao, Y., Jin, X., Wang, Y., Cheng, X.: Document Embedding Enhanced Event Detection with Hierarchical and Supervised Attention. In: 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018). pp. 414–419 (2018)