Raccoons at SemEval-2022 Task 11: Leveraging Concatenated Word embeddings for Named Entity Recognition

Atharvan Dogra
Thapar University, Patiala, India
atharvandogra007@gmail.com

Prabsimran Kaur
Thapar University, Patiala, India
pkaur_be18@thapar.edu

Guneet Singh Kohli
Thapar University, Patiala, India
guneetsk99@gmail.com

Jatin Bedi
Thapar University, Patiala, India
jatin.bedi@thapar.edu

Abstract

Named Entity Recognition (NER), an essential subtask in NLP that identifies text belonging to predefined semantics such as a person, location, organization, drug, time, clinical procedure, biological protein, etc. NER plays a vital role in various fields such as information extraction, question answering, and machine translation. This paper describes our participating system run to the Named entity recognition and classification shared task SemEval-2022. The task is motivated towards detecting semantically ambiguous and complex entities in short and low-context settings. Our team focused on improving entity recognition by improving the word embeddings. We concatenated the word representations from State-of-the-art language models and passed them to find the best representation through a reinforcement trainer. Our results highlight the improvements achieved by various embedding concatenations.

1 Introduction

Named entity recognition and classification is an essential subtask in natural language processing (NLP). The task of identifying named entities like a person, location, organization, drug, time, clinical procedure, biological protein, etc. in a text document is key to many applications, including information extraction and retrieval, machine translation(Al-Onaizan and Knight, 2002; Steinberger et al., 2013), topic modeling(Newman et al., 2006), text summarization(Schiffman et al., 2002), and question answering. It can also be used in domain-specific entity types such as disease, symptoms, and treatments. Recently many researchers have identified the name identification issue in a variety of languages and have made efforts to apply named entity recognition.

Complex and ambiguous Named entities, like the title of creative works such as books, songs, and movies, are not simple nouns and are thus harder to recognize(Ashwini and Choi, 2014). Unlike the traditional NEs these works take the form of linguistic constituents such as "Remember me when we are parted," which is an imperative clause and does not look like the traditional NEs (Locations, Person name, organizations). This ambiguity in syntax makes it difficult to identify them based on their context. There can also occur instances where these titles are semantically ambiguous; for example, "the girl on the train" can be a preposition or the name of a book. Such entities are growing at a faster rate as compared to the traditional categories. Therefore, processing these NEs has always been a challenging task in NLP in practical and open-domain settings. However, it has received enough engagement from the research community.

Despite the high score produced by Neural models on benchmark datasets like CoNLL03/OntoNotes, it has been noticed (Augenstein et al., 2017) that these models perform significantly low on complex or unseen data. This happens because the scores were driven by the presence of easy entities, well-formed text, and memorization due to entity overlap between train/test sets. Various researchers have noted that the majority of the errors in their downstream tasks have occurred due to the failure of NER systems to recognize complex entities. Various researchers using NER on downstream tasks have noted that a significant proportion of their errors are due to NER systems failing to recognize complex entities.

This paper presents our system for Shared Task on "MultiCoNER Multilingual Complex Named Entity Recognition @ SemEval 2022" (Malmasi et al., 2022b), (Meng et al., 2021), (Fetahu et al., 2021). The task focused on detecting complex and ambiguous entities in short and low-context settings. Our team focused on improving entity recognition by improving the word representations, following (Wang et al., 2021; Yamada et al., 2020). We concatenated the word embeddings from State-of-the-art language models and passed them to find

Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pages 1576 - 1582

the best representation through a reinforcement trainer.

2 Related Work

Name entity recognition has always been a challenging task that requires massive prior knowledge resources for good performance (Ratinov and Roth, 2009; Nadeau and Sekine, 2007). Research in various domains using a variety of approaches has been done. Early methods included heuristics and handcrafted rules, lexicons, orthographic features, and ontologies. These systems involved recognizing entities based on pattern matching with input documents(Rau, 1991; Collins and Singer, 1999). A higher degree of accuracy may be achieved using the rule-based system. However, it is time-consuming and challenging to port the developed rules from one domain to another.

Later machine learning methods were used (Borthwick, 1999; McCallum and Li, 2003; Takeuchi and Collier, 2002) . This learning technique involved supervised and unsupervised approaches. The supervised being a dominant technique for named entity recognition (Nadeau and Sekine, 2007). However, this method required massive high-quality annotated data. There also exist some hybrid models that make use of different rule-based and/or learning methods for enhanced performance (Srihari, 2000; Rocktäschel et al., 2012). These systems make the best use of good features and methods for improved performance.

Recently neural network systems (Collobert et al., 2011) have also become a popular method. These systems with minimal feature engineering have become appealing because they do not require sources such as lexicons or ontologies, thus making them domain independent. Most of the neural architectures proposed are based on recurrent neural networks (RNN).

3 Dataset Description

The English dataset provided by SemEval 2022 (Malmasi et al., 2022a) comprises 233,917 instances of data. The training sample included 15300 annotated statements, the dev set comprised 800 annotated sentences, and the test set included 217817 test data (as shown in the table). The sentences in the data are allocated a respective id. Each sentence is further divided into tokens which are separated by a newline. Each token is assigned a label according to the standard BIO tagging followed by the named entity tag in the specified format. *<token>_<BIO_tag>-<NE>*

In the BIO tagging, B and I tags are followed by <NE> tag, while O tags have no following tag. As presented in the fig, an instance of the dataset, statements with the 'O' tag represents a non-named entity. The statements with 'B' tags represent the beginning of the named entity, and 'I' tags represent the continuation of the same-named entity. The tokens are assigned one of the six named entities Person (abbreviated as PER), Location (abbreviated as LOC), Group (abbreviated as GRP), Corporation (abbreviated as CORP), Product (abbreviated as PROD), Creative Work (abbreviated as CW).

4 Methodology

The method used focuses on automating the process of finding better concatenation of embeddings for improved performance. In this approach, a task model and controller module repeatedly interact. To achieve high accuracy, the controller searches for a better embedding concatenation from the given set of pre-trained embeddings. The task model, on the other hand, produces a task output. The architecture works on a reward basis. The controller is rewarded every time the task model is trained over the task data after an embedding is generated. The controller receives a reward for updating its parameter and sampling new embedding concatenations. The general architecture of our approach is shown in Figure 1.

4.1 Design of Task Model

For the task model, a sequence-structure approach is used. Considering the input sentence to be *m* and the structured output to be *n*, the probability distribution *P(n|m)* can be calculated as:

$$P(n|m) = \frac{exp(Score(m,n))}{\sum_{n' \in N(m)} exp(Score(m,n'))} \quad (1)$$

where *N(m)* represents all possible output structures given the input sentence *n*. The BiLSTM-CRF model (Ma and Hovy, 2016; Lample et al., 2016) was used for sequence-structured outputs.

$$P^{seq}(n|m) = BiLSTM - CRF(E,n) \quad (2)$$

where $E = [e_1;\ e_2;\ ...\ e_n], E \in \mathbb{R}^{d \times w}$ is matrix of word representation for the input sentence *m* comprising of *w* words, The hidden size of the concatenation of embeddings is represented by *d*.

E_i is the word representation of the i^{th} word, which is a concatenation of L types of word embeddings:

$$e_i^l = embed_i^l(m); \;\; e_i = [e_i^1; \; e_i^2; \; ... \; e_i^L] \quad (3)$$

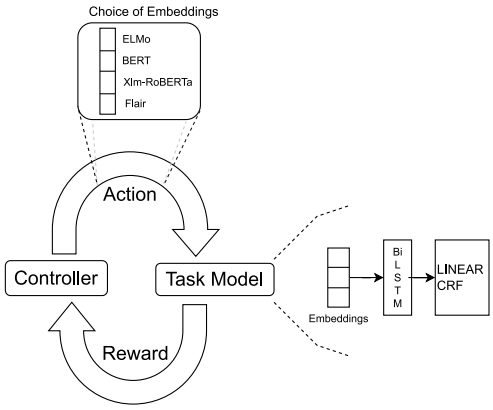

Figure 1: Architecture of our approach. It shows the task model, returning of reward function and the embeddings being concatenated

4.2 Design of Search Space

A set of neural networks can be used to represent neural architecture search space (Elsken et al., 2017). In this method, we have represented the embedding candidates as nodes. The sentence n is passed as the input to the nodes, and the resulting output is the embeddings e^l. Since the embeddings are concatenated as word representations, there is no connection between the nodes in search space, resulting in a significantly reduced search space.

A variety of embeddings are available for the extraction of word representation, for instance, finetuned XLM-RoBERTa-large embeddings.However, due to the restriction on the use of a multilingual model for a single language track, we finetuned the BERT embeddings on the provided training data and concatenated the last four layers as word features applying mean pooling operation over them (Devlin et al., 2019).

Unlike (Kondratyuk and Straka, 2019), applying a weighted sum of all twelve layers did not significantly differ the empirical results. There was no significant difference between the XLM-RoBERTa CONLL-03 English embeddings and the Finetuned BERT-Large embeddings for the submitted predictions. Thus we used regular embeddings to reduce search space. Subsequently, each embedding only has a specific operation resulting in a search space that contains 2^L-1 possible combinations of nodes.

Except for the character embeddings, all the other weights of the pre-trained embedding candidates are fixed. The parameters of task models are shared at each iteration of the search in accordance with Neural Architecture Search (NAS). All the nodes in the graph are kept in the search space. Each node performs an operation to indicate whether the embedding is masked out. This is done to avoid deciding which node is to be kept, making weight sharing difficult. For this binary task.

For this task, a binary vector $o = [o_1, \ldots, o_l, \ldots, o_L]$ is used as a mask for those embeddings which are not selected:

$$e_i = [e_i^1 o_1; \; e_i^2 o_2; \; ... \; ; e_i^l o_l; ... \; ; e_i^L o_L] \quad (4)$$

Where o_l represents a binary variable. Input E is applied to the linear layer in the BiLSTM layer hence the multiplication operation in E, *i.e.* multiplying mask with embedding, is equivalent to directly concatenating the selected embeddings:

$$W^T e_i = \sum W_l^T e_i^l o_l \quad (5)$$

Also, the unused embedding candidates and the corresponding weights in W are removed for a light task-model after finding the best concatenation.

4.3 Searching in the Space

The controller is responsible for generating the embedding mask using parameters which are generated using $\theta = [\theta_1; \theta_2; ...; \theta_L]$. The probability distribution of selecting a concatenation o is $P^{ctrl}(q; \theta) = \Pi_{l=1}^{L} P_l^{ctrl}(o_l; \theta_l)$. Element o_l of $\mathbf{o}$ is sampled using Bernoulli distribution which is given by:

$$P_l^{ctrl}(o_l; \theta_l) = \begin{cases} \sigma(\theta_l), & o_l = 1. \\ 1 - P_l^{ctrl}(o_l = 1; \theta_l), & o_l = 0. \end{cases} \quad (6)$$

where σ is the sigmoid function. Given the mask, the task model is trained until convergence and returns an accuracy R on the dev set. The reinforcement algorithm is used for optimization. The accuracy R is used as a reward signal for training the controller, whose aim is to maximize the reward $J(\theta) = \mathbb{E}_{P^{ctrl}(o;\theta)}[R]$ utilizing the policy of gradient method (Williams, 1992). The reward function that we have used:

$$r^t = \sum_{i=1}^{t-1} (R_t - R_i) \gamma^{Hamm(o^t, o^i)-1} |o^t - o^i| \quad (7)$$

The final gradient comes out as:

$$\nabla_\theta J(\theta) \approx \sum_{l=1}^{L} \nabla_\theta . \, log P_l^{ctrl}(o_l^t; \theta_l) \, r_l^t \quad (8)$$

This reward function evaluates how each embedding candidate contributes to the accuracy change. The binary vector $|o^t - o^i|$ represents the change in embeddings. o^t represents a binary vector at current iteration *t*, and o_i represents the previous time step *i*. The Hamming distance $Hamm(o^t, o^i)$ is used to measure the contribution of embedding candidates in the accuracy. As hamming distance increases, the contribution becomes less significant.

4.4 Training the controller

The corresponding validation scores and concatenations were stored in a dictionary D to train the controller. Firstly the task model was trained with all the embedding candidates concatenated. Then the concatenation o^t was sampled based on Equation (6). The task model was trained using the equation (4) after which the model is evaluated on a development set to return an accuracy R_t. Then the gradient is computed for the controller following Equation (8) using the concatenation o_t, accuracy R_t and D. Based on the computed gradient, the parameters of the controller are updated. Finally, o^t and R_t are added to D. If a concatenation o^t is in dictionary, we compare its accuracy with the value in the dictionary and keep the higher one. Selecting o^{t-1} (i.e., previous concatenation) and zero vector is avoided.

5 Experimental Setup

Data was provided in form of sentences which were broken down into individual words, or tokens, each lying in a newline and their corresponding tags in front of them. All the tokens are first passed through a function to encode them into embeddings, which are initial representations for the tokens. These tokens are pushed through the RNN language model which forms the task model and iteratively returns a reward to the controller.

Table 1: Major Hyperparameters

Parameters	Value
BiLSTM size	800
BiLSTM layer	1
Optimizer	SGD
Learning rate	[0.1, 7.8125e-4]
Epochs	150
Episodes	20

This model first contains a dropout layer, then an encoder layer is added which contains the embedding. The next layer is an LSTM layer with a hidden size of 800. Finally, the representations are fed into a linear-chain CRF layer to predict the final label sequence, where a linear layer is applied for the representations to score each entity label. The above deep learning model has been built using PyTorch[1] along with the transformer embeddings that have been used.

The learning rate was set at 0.1 at the beginning with a patience level set to 5, i.e. the learning rate was halved if there was no improvement in monitored metrics for the patience, that is the validation loss. The batch size was set to 64 and a maximum of 150 epochs were allowed for 20 episodes.

6 Results

The table below shows the micro and macro F1 scores of prediction on the development set. As the tags of the test set are unavailable to us, the model's performance was judged on dev set only and the results of the best 3 models were ensembled in the end, for submission. The best performing model was achieved in a model, where the following embeddings were concatenated: Finetuned BERT-large-uncased, Finetuned RoBERTa-large, ELMo original, FastCharacterEmbeddings, Glove, FastWordEmbeddings-english, Flair news-en embedding. The model was trained on with the following hyperparameters setting 150 epochs, 20 episodes, SGD optimizer, 800 hidden units of BiLSTM-CRF, starting with a learning rate of 0.1 and a batch size of 64.

Table 2: Results with concatenated BERT embeddings

Concatenations	Micro-F1	Macro-F1
TransformerWordEmbedding: BERT-large-uncased ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	**0.8867**	**0.8745**
TransformerWordEmbedding: BERT-large-cased ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	0.8596	0.8470
TransformerWordEmbedding: BERT-base-cased ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	0.8597	0.847
TransformerWordEmbedding: BERT-base-uncased ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	0.8861	0.8693

[1] https://pytorch.org/

Table 3: Results with concatenated Xlm-RoBERTa embeddings

height	**Concatenations**	**Micro-F1**	**Macro-F1**
	TransformerWordEmbedding: Xlm-RoBERTa-large-finetuned-conll03-english (finetuned on current dataset) ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	0.868	0.8536
	TransformerWordEmbedding: Xlm-RoBERTa-large-finetuned-conll03-english ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	0.8563	0.8423
	TransformerWordEmbedding-0: Xlm-RoBERTa-large-finetuned-conll03-english TransformerWordEmbedding-1: RoBERTa-large ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	0.8728	0.8592
	TransformerWordEmbedding-0: Xlm-RoBERTa-large-finetuned-conll03-english TransformerWordEmbedding-1: BERT-large-uncased ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	**0.8997**	**0.8881**
	TransformerWordEmbedding: Xlm-RoBERTa-base ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	0.8465	0.8332
	TransformerWordEmbedding: Xlm-RoBERTa-large ELMoEmbedding: original || FastCharacterEmbeddings FastWordEmbedding-0: glove || FastWordEmbeddings-1: en FlairEmbedding: news-forward	0.8596	0.8456

These results were ensembled with two other models which were developed as follows:

- BERT-base-uncased, BERT-large-uncased, and RoBERTa-large with the rest of the embeddings as it is.
- Finetuned BERT-large-uncased with the rest of the embeddings as it is.

7 Result Analysis

Experiments were conducted to analyze the performance of systems by using different combinations of embeddings. The best concatenation of embeddings can be easily seen in the model containing BERT-large-uncase transformer embedding and ELMo original, FastCharacterEmbeddings, GloVe FastWord: english, and Flair: news-forward with a Macro-F1 score of 0.8745 on the dev set. This model generated the best results after ensembling the results with the models containing BERT-base-uncased embeddings and BERT-large-uncased finetuned on the current dataset, which generated a Macro-F1 score of **0.7418**[2] on the test dataset.

Table 4: Comparison in baseline and our best model

Models	**Micro-F1 Score**[3]
Baseline	0.715
Our Best	0.8867

One clear distinction in the result can be seen among the BERT uncased and cased models. In the BERT-uncased models, the text is set to lowercase before the WordPiece tokenization step, hence case is irrelevant in it, while in BERT-cased models, the case of words is also considered. In our dataset, the tokens were already lowercase.

For comparison's sake, we have experimented with RoBERTa embeddings as well and they have produced slightly better results than BERT embeddings alone. One difference in the functioning of BERT and RoBERTa was noticed during experimenting with their tokenizers. The BERT tokenizer sometimes separated hyphen-separated entities and treated them as individual entities while RoBERTa tokenizer treated them as a combined single entity and generated a representation for it.

For comparison, we have also experimented with XLMR embeddings during result compilation to show the difference (or similarity) between results. After removing a few of the other embeddings like ELMo, Flair, and Glove, we have also collected results from the above model. All the results can be referred to in Table 3.

8 Conclusion and Future work

In this paper, we present our approach to SemEval-2022 Task 11:MultiCoNER Multilingual Complex Named Entity Recognition. Our best submission gave us an F_1 score of **0.7418**, placing us 10^{th} on the Evaluation Phase Leaderboard. Future work includes experimenting with multilingual data and embeddings and different optimizers.

[2]This is the final Macro-F1 score on the test set

[3]Micro-F1 score was the only metric available for baseline

References

Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 400–408.

Sandeep Ashwini and Jinho D Choi. 2014. Targetable named entity recognition in social media. *arXiv preprint arXiv:1408.0782*.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. *arXiv preprint arXiv:1704.02853*.

Andrew Eliot Borthwick. 1999. *A maximum entropy approach to named entity recognition*. New York University.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT conference on empirical methods in natural language processing and very large corpora*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. 2017. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*.

Besnik Fetahu, Anjie Fang, Oleg Rokhlenko, and Shervin Malmasi. 2021. Gazetteer Enhanced Named Entity Recognition for Code-Mixed Web Queries. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1677–1681.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Shervin Malmasi, Anjie Fang, Besnik Fetahu, Sudipta Kar, and Oleg Rokhlenko. 2022a. MultiCoNER: a Large-scale Multilingual dataset for Complex Named Entity Recognition.

Shervin Malmasi, Anjie Fang, Besnik Fetahu, Sudipta Kar, and Oleg Rokhlenko. 2022b. SemEval-2022 Task 11: Multilingual Complex Named Entity Recognition (MultiCoNER). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons.

Tao Meng, Anjie Fang, Oleg Rokhlenko, and Shervin Malmasi. 2021. GEMNET: Effective gated gazetteer representations for recognizing complex entities in low-context input. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1499–1512.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

David Newman, Chaitanya Chemudugunta, and Padhraic Smyth. 2006. Statistical entity-topic models. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 680–686.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155.

Lisa F Rau. 1991. Extracting company names from text. In *Proceedings the Seventh IEEE Conference on Artificial Intelligence Application*, pages 29–30. IEEE Computer Society.

Tim Rocktäschel, Michael Weidlich, and Ulf Leser. 2012. Chemspot: a hybrid system for chemical named entity recognition. *Bioinformatics*, 28(12):1633–1640.

Barry Schiffman, Ani Nenkova, and Kathleen McKeown. 2002. Experiments in multidocument summarization.

Rohini K Srihari. 2000. A hybrid approach for named entity and sub-type tagging. In *Sixth Applied Natural Language Processing Conference*, pages 247–254.

Ralf Steinberger, Bruno Pouliquen, Mijail Kabadjov, and Erik Van der Goot. 2013. Jrc-names: A freely available, highly multilingual named entity resource. *arXiv preprint arXiv:1309.6162*.

Koichi Takeuchi and Nigel Collier. 2002. Use of support vector machines in extended named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Automated concatenation of embeddings for structured prediction.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.