IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 3, 2020, accepted July 20, 2020, date of publication July 23, 2020, date of current version August 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011598

A Bidirectional Iterative Algorithm for Nested Named Entity Recognition

SŁAWOMIR DADAS AND JAROSŁAW PROTASIEWICZ
National Information Processing Institute, 00-608 Warsaw, Poland

Corresponding author: Sławomir Dadas (slawomir.dadas@opi.org.pl)

This work was supported by the Ministerstwo Nauki i Szkolnictwa Wyższego under Grant 10.13039/501100004569.

ABSTRACT Nested named entity recognition (NER) is a special case of structured prediction in which annotated sequences can be contained inside each other. It is a challenging and significant problem in natural language processing. In this paper, we propose a novel framework for nested named entity recognition tasks. Our approach is based on a deep learning model which can be called in an iterative way, expanding the set of predicted entity mentions with each subsequent iteration. The proposed framework combines two such models trained to identify named entities in different directions: from general to specific (*outside-in*), and from specific to general (*inside-out*). The predictions of both models are then aggregated by a selection policy. We propose and evaluate several selection policies which can be used with our algorithm. Our method does not impose any restrictions on the length of entity mentions, number of entity classes, depth, or structure of the predicted output. The framework has been validated experimentally on four well-known nested named entity recognition datasets: GENIA, NNE, PolEval, and GermEval. The datasets differ in terms of domain (biomedical, news, mixed), language (English, Polish, German), and the structure of nesting (simple, complex). Through extensive tests, we prove that the approach we have proposed outperforms existing methods for nested named entity recognition.

INDEX TERMS Information extraction, natural language processing, nested named entity recognition.

I. INTRODUCTION

Named entity recognition (NER) is a well-established technique in natural language processing (NLP) which involves finding and classifying named entities in text. These tasks have been widely applied in various domains for identification of names of people, organizations, temporal expressions, geographic locations, or specialized entities in scientific documents [1]. From a practical point of view, finding named entities or, more generally, entity mentions is useful in solving more complex problems such as information retrieval, knowledge base population, or natural language understanding (NLU). From a supervised learning perspective, named entity recognition is an example of structured prediction, that is, the prediction of structured objects from the data rather than simple categories (classification) or numeric values (regression). In NER, those structures are defined as single words or phrases which refer to named entities. Typically, the task is simplified by the assumption that named entity mentions cannot overlap. In practice, however, named entities frequently contain names of other entities, and many applications could benefit from the ability to identify hierarchical structures of mentions. In Figure 1, we demonstrate an example of a sentence from the NNE [2] corpus annotated with a set of overlapping tags.

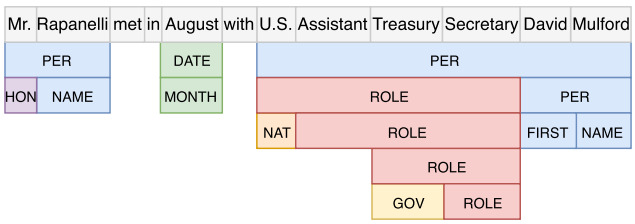

FIGURE 1. An example of a sentence annotated with nested named entity tags with up to five levels of nesting. This sentence is sourced from the NNE dataset [2].

The problem of nested named entity recognition has often been ignored, mostly because of the computational complexity involved and other technical limitations. Only in recent years has it gained recognition, as an effect of rapid development in the field of machine learning, specifically deep

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Shariq Imran.

learning techniques. As a result, several neural models for achieving nested named entity recognition have been developed in recent years.

In this paper, we present a novel algorithm for nested named entity recognition. The algorithm utilizes a neural architecture consisting of a character-based language model, bidirectional long short-term memory (LSTM) [3] layers, and an inference layer consisting of linear-chain conditional random fields (CRFs) [4]. The distinctive feature of our model is the ability to generate predictions in an iterative way. The same sequence of words can be passed to the model several times along with the set of previously generated named entity annotations, and the model outputs a new set of predictions in each iteration. This simple approach can easily adapt to many nested named entity recognition tasks with various output structures and depth of nesting. We offer the following contributions in this work:

1) We propose an effective neural architecture for nested named entity recognition. Our model works iteratively and does not impose any restrictions on the length of entity mention, number of entity classes, depth, or structure of the predicted output.
2) We introduce a bidirectional algorithm for nested named entity recognition which combines two iterative models trained independently: one in the outside-in and the other in the inside-out direction. The predictions of both models are merged into a single coherent set of entity mentions, using the pre-defined selection function, policy. We propose several selection policies that can be used with our algorithm.
3) We evaluate our approach on four well-known nested NER datasets, each of which operates in different languages and domains. On each dataset, our algorithm outperforms other recently introduced methods for nested named entity recognition. We demonstrate how applying different selection policies affects the characteristics of the generated output and by extension, the performance of our method.

II. RELATED WORK

Early approaches to nested named entity recognition started to appear following the release of GENIA corpus, a biomedical dataset wherein some of the annotated entities are embedded inside other entities [5]. At that time, most studies ignored this fact, and focused on detecting outer entities only. The first solutions that attempted to predict nested structures used a combination of Hidden Markov Models (HMM) to detect subsets of named entities, and handcrafted rules to expand these subsets. [6]–[8]. Support vector machines (SVM) have also been used - Zhou [9] combined such a model with a rule-based approach, while Gu [10] used two separate SVM models to detect the innermost and outermost entities. From these beginnings, we now recognize methods based on conditional random fields (CRF), hypergraphs, and neural networks in the literature of nested named entity recognition.

A. CRF-BASED METHODS

One publication in particular by Alex *et al.* [11] proved to be important when addressing the problem of nested named entity recognition. They proposed a number of techniques based on CRF, namely layering, cascading, and joined label tagging. In layering, several CRF models are trained to detect subsequent levels of named entities. In addition to word-level attributes, all models except the first use the predictions of the previous model as features. Layering can be implemented in one of two directions: inside-out layering starts with the identification of the innermost entities and detects the outer entities in the subsequent layers; whereas outside-in layering begins with the outermost entities and detects the inner entities next. Cascading works by training separate CRF models to identify separate types of named entity. As with layering, the models have a specified order, and each model can use the predictions of the previous model as an input. A major drawback of cascading is that it cannot detect nested entities of the same type for the reason that type-specific CRF models generate flat predictions. The final method, joined label tagging, trains a single CRF model, but expands the set of all possible labels by joining nested tags from all levels into a single label. Two years later, Finkel and Manning [12] formulated the task of nested NER as a parsing problem, and proposed a CRF-based algorithm with $O(n^3)$ time complexity for solving it.

B. HYPERGRAPH-BASED METHODS

Another group of methods models the representation of entity mentions in a sentence as a hypergraph. Such a hypergraph consists of various types of node that represent specific mention properties, such as its left and right boundaries, labels, or words that are included in the mention. This approach was first proposed for nested NER by Lu and Roth [13]. The construction of an optimal mention hypergraph was solved using log linear modeling based on a number of handcrafted features. This work was later improved by Muis and Lu [14], who introduced a novel encoding scheme for nested mentions that assigns labels to the gaps between words. Wang and Lu [15] proposed a modified hypergraph representation that did not suffer from structural ambiguity, and included representation generated by bidirectional LSTM for span, word, and character-level features. Katiyar and Cardie [16] proposed a method for building entity mention hypergraphs which varied from the previous approaches inspired by Lu and Roth [13]. They used a multi-layer encoder-decoder style neural network with bidirectional LSTM layers. Their decoder layer constructs a simple hypergraph based on the BILOU (beginning-inside-last-outside-unary) tagging scheme. Hypergraph-based approaches are flexible and capable of modeling many types of nested structure. However, their computational complexity is affected by some of dataset-specific properties such as sequence length, maximum length, depth of entity mention, and the number of possible entity labels. As a result, they

can quickly become computationally inefficient for larger datasets with complex entity structures.

C. NEURAL METHODS

In recent years, several neural architectures for nested named entity recognition have been developed. Due to the variety of neural approaches, they cannot be fully categorized. However, we wish to highlight a few popular methodologies.

Exhaustive methods, enumerating all subsequences up to a certain length, have been investigated. Xu *et al.* [17] built a feedforward neural network classifier that examines all possible entity candidates, and tags them with an appropriate entity label. A similarly exhaustive approach is presented in the research of Sohrab and Miwa [18], in which an LSTM encoder is used to build word and sequence representations, and the output layer predicts a label for each possible sequence.

The idea of combining several models into one nested named entity recognition system is often exploited. Lin *et al.* [19] train two neural models - one for identifying the *anchor words* of a mention, and the other for detecting word regions around those anchors. A framework composed of two neural models has also been proposed by Xia *et al.* [20]. The first model, detector is responsible for identifying which word segments are named entity mentions: the second, classifier is responsible for assigning categories to those mentions. The notion of separating boundary detection and named entity classification is also utilized in Chen *et al.* [21]. The first of their two models is used for identifying entity boundaries which are the assembled into entity candidates using greedy matching. The second model assigns a label to each entity candidate, based on its right and left context as input features.

Another popular approach involves the use of transition-based parsing. Wang *et al.* [22] combine this method with a neural model to predict a set of nested entity tags represented as a forest where each outermost entity is a root of a tree, and nested entities are its children. The forest is constructed sequentially, where in each step the system decides which of the possible transition actions to execute, given its current state. Transition based parsing is also utilized by Marinho *et al.* [23], but with a different set of parser actions and predicted representation.

Methods not falling into any of the above categories are usually based on complex, multilayer neural architectures. A model constructed by Ju *et al.* [24] detects nested entities in the inside-out order with a stack of LSTM layers. Each layer has a CRF output generating predictions for the current level of nesting, and its hidden states are passed to the next layer until no new entities are detected. Zheng *et al.* [25] have trained an LSTM-based multitask model to jointly detect the boundaries of named entities and classify them. Recent research has started to utilize transformer [26] architecture. For example, Sun *et al.* [27] encoded sentences using two unidirectional transformers followed by a convolutional layer. Features representing word spans, generated by convolution, are then transformed into a fixed-length vector with a specialized pooling operation. Finally, a fully connected layer with a softmax activation assigns an entity label to each such vector. Shibuya and Hovy [28] employed a BiLSTM-CRF architecture with contextual word representations from a pre-trained BERT [29] model. The main contribution of their work is the modification of a Viterbi decoding algorithm in the CRF layer to recursively identify nested entities in the outside-in manner.

III. RESEARCH METHODOLOGY

In this section, we describe our approach to nested named entity recognition. First, we introduce a neural iterative model which can be used for solving sequence tagging problems with nested structures. Next, we present a bidirectional algorithm which utilizes two instances of that model and a selection policy component. We then describe six selection policies we have implemented in our work.

A. NEURAL ITERATIVE MODEL

In a typical sequence tagging model, data is processed sequentially, in which a single input represents a sentence from a text corpus. In our case, a sentence of length n is defined as a sequence of words:

$$x = [x_1, x_2, \ldots, x_{n-1}, x_n] \tag{1}$$

Let m be the predicted entity mention of length k, defined as a triplet consisting of k consecutive words from the input sequence x, the entity label l, and the confidence score of the mention $c(m)$:

$$m = ([x_i, x_{i+1}, \ldots, x_{i+k-1}], l, c(m)) \tag{2}$$

The goal of an entity recognition model is to find and correctly classify all entity mentions in a sentence. Our method achieves this goal through multiple inference steps, each of which identifies a new subset of entity mentions.

Let S denote the set of all predicted unique entity mentions in the sentence x. We wish to learn a function f that, given an input sequence x and a set of previously found entity mentions S, outputs a sequence of entity labels y:

$$f : (x, S) \rightarrow y \tag{3}$$

The iterative prediction works as follows. Before the first iteration, S is initialized to be an empty set. After each iteration, we add all entity mentions from the output of the model to S. The process is repeated until no new mentions are found. After the final iteration, S represents the final predictions of the model. This process is similar to the layering technique described in Alex *et al.* [11], but with two important distinctions. First, in layering each iteration is handled by a separate model, so the number of models that need to be trained is equal to the maximum depth of entity nesting in the dataset. In our case, only one model needs to be trained, and it is used to handle all iterations. Unlike in Alex *et al.* [11] and many other nested named entity solutions, the maximum depth of nesting does not affect the architecture of our model,

and does not need to act as a pre-defined hyperparameter. In practice, our model exploits the structure of a training set, including the depth of nesting. In our experiments our model never exceeded the number of iterations needed to detect the deepest mentions found in the data. Secondly, in the classic layering approach, only predictions from the previous model are passed as inputs to the next model. We assumed that it would be reasonable to take into account all of the predictions from preceding iterations, so the set S is used as an input, which includes all entity mentions identified so far.

Figure 2 illustrates the simplified architecture of our model. From the sequence of words, we construct an intermediate vectorized representation. Each word includes three components: a contextual representation from a character-based language model; a word embedding vector; and a multi-hot vector encoding the set of named entity labels predicted for that word in past iterations. The input to the model is a concatenation of these three vectors. The dimensionality of a single sequence element is therefore equal to the sum of the dimensions of the respective vectors.

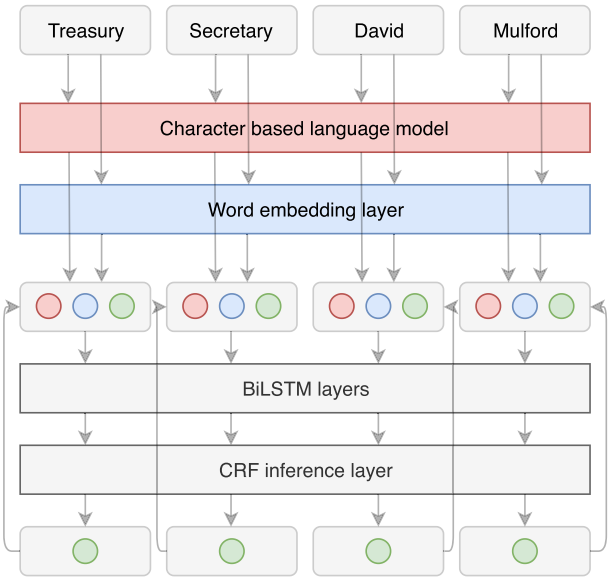

FIGURE 2. The architecture of our iterative neural model. The intermediate representation of a word is constructed by concatenating the output of a character-based language model (red color), static word embedding (blue color), and a vector of encoded predictions for that word from previous iterations (green color). This representation is then used as an input for a stack of BiLSTM layers and the CRF inference layer, which generates predicted word annotations for the current iteration.

The multi-hot encoded representation is a binary vector with dimensionality equal to the number of labels plus 1, in which each position corresponds to a specific label, and the extra bit is used to mark the initial iteration. The value at index i is equal to 1 only if the i-th label has been predicted for that word in previous iterations, or otherwise equals zero. For instance, let us assume we have a set of four labels - three for entity types, and one for outside words $\{PER, GEO, ORG, O\}$. At the first iteration, the multi-hot vector for each word is equal to $[0, 0, 0, 0, 1]$, with the extra bit set to 1, indicating that no predictions have yet been generated. If a word was tagged with the *PER* label, its vector for the second iteration would be equal to $[1, 0, 0, 0, 0]$. From this moment, each new prediction for that word would activate additional bits in the vector for subsequent iterations. Note that for simplicity, we assumed that labels directly corresponded to entity types. In practice, many modern approaches to named entity recognition utilize tagging schemes, such as BIO (beginning-inside-outside) or BILOU (beginning-inside-last-outside-unary), to be able to differentiate the inner and boundary words of an entity mention. In this work, we used the BIO scheme for encoding entity types. Therefore the dimensionality of multi-hot vector and the number of labels was three times the number of entity types.

The three-component intermediate word representation is used as an input to two bidirectional LSTM (BiLSTM) layers that compute context-dependent hidden states. Finally, the output of the last BiLSTM is sent to a CRF inference layer, which is responsible for predicting a sequence of labels y which maximizes the conditional probability $P(y|x, S)$ for the current iteration. The confidence score of the mention $c(m)$ of type l from Equation 2 is defined as the arithmetic mean of probabilities of its individual words belonging to class l:

$$c(m) = \frac{\sum_{j=i}^{i+k-1} P(y_j = l|x, S)}{k} \tag{4}$$

B. TRAINING PROCEDURE

The training procedure is similar to that of flat NER models which employ BiLSTM-CRF architecture, and their modern variants which use pre-trained language models. In our case, we split each nested NER example into several inside-out and outside-in layers, and each layer is treated as a separate training sample. The model can be trained either with an inside-out or outside-in split, as is outlined in the classic layering approach [11].

In some cases, the order of entity mentions can be ambiguous, for example, the *DATE* and *MONTH* labels in the sentence from Figure 1. In these situations, we collect the frequency of parent-child relationships between the two labels from all unambiguous examples in the training set, and order the labels in accordance with the more frequently occurring relation of the two. A single training example consists of two input sequences: the sequence of words, and the sequence of vectors representing the predictions of previous iterations. At the training stage, the sequence of previous predictions is created synthetically based on the gold labels in the training set, as if it were generated by a perfect classifier. We use the real predictions of the model only at the stage of inference. By following this process, the samples are independent of each other during training, and the process need not be altered to fit the iterative nature of the model. This training procedure, also known as teacher forcing [30], has been successfully applied in the past for generating other complex structures, such as images [31] or audio [32]. Since we use a CRF output layer with Viterbi

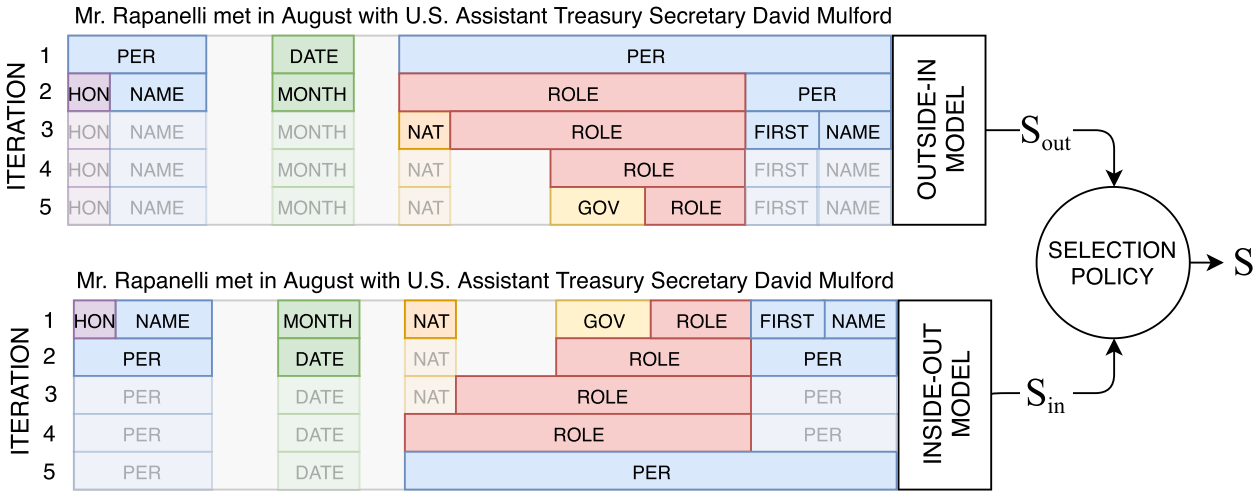

FIGURE 3. A high-level overview of a bidirectional iterative algorithm for nested named entity recognition. Two neural models are trained on the same dataset to identify entity mentions in the inside-out and the outside-in orders. After all named entities for a specific sequence of words have been detected, a model is expected to repeat the last prediction in subsequent iterations. The outputs of both models are combined using a selection policy that constructs the final set of predictions of the algorithm.

decoding, the loss function is the sentence level negative log likelihood.

C. BIDIRECTIONAL ITERATIVE ALGORITHM

We have described the neural model for iterative detection of nested entity mentions. In this subsection, we propose an algorithm that utilizes two such models, and combines their outputs to generate the final set of entity mentions. Both models are trained on the same dataset, but their goals are to detect named entities in opposite directions: using the inside-out and outside-in approach. Although it is possible to use a standalone iterative model, our hypothesis is that combining two views of the same output structure identifies nesting patterns that a single model would be unable to detect. Figure 3 demonstrates the core concept of our algorithm. Please note that while both nested structures are always divided into the same number of iterations for the same sentence, they are not simply reversed versions of each other, as the model is trained to detect entity mentions in the earliest iteration possible. The algorithm works as follows: both models produce their outputs independently, using the approach described in the previous subsection. The predictions of the inside-out (S_{in}) and outside-in (S_{out}) models are then processed by function g, which we have named the *selection policy*. g takes two sets of entity mentions and produces a filtered set, S, that includes zero or more elements from S_{in} and S_{out}:

$$g : (S_{in}, S_{out}) \rightarrow S \tag{5}$$

In this study, we evaluated the following six selection policies:

1) **Outside-in model only** - This is a baseline policy in which only entity mentions from the outside-in model are selected. Applying this policy is equivalent to using a standalone outside-in model.
2) **Inside-out model only** - This is another baseline policy analogous to the one above, but accounting only for the predictions from the inside-out model.
3) **Model intersection** - In this policy, an entity mention is included in the resulting set, S if and only if it is present both in S_{in} and S_{out}.
4) **Model union** – In this policy, an entity mention is included in the resulting set, S if it is present in any of the sets, S_{in} and S_{out}.
5) **Probability-based selection** - This is a simple policy that accounts for the confidence score of a mention, $c(m)$. In this policy, the mention, m is accepted if and only if its confidence score, $c(m)$ is greater than or equal to a specified threshold hyperparameter. We use separate threshold values for sets, S_{in} and S_{out}. Those hyperparameters are found using a linear search to maximize the F1-score on the validation set.
6) **Linear classifier selection** - This policy uses an additional logistic regression model which decides whether to accept or reject a mention, m. The model is trained on a list of features extracted from the sets of predictions, S_{in} and S_{out} generated from the training data. For a specific mention, m, those features include: the confidence scores of m in S_{in} and S_{out} (or 0 if m is not present in the set), the label of m in S_{in} and S_{out} (or *null* if m is not present in the set), the label of a parent entity mention in S_{in} and S_{out} (or *null* if m is not present in the set or has no parent entity), the confidence of a parent entity mention in S_{in} and S_{out} (or 0 if m is not present in the set or has no parent entity), the minimum and maximum confidence of mention m found in the sets S_{in} and S_{out}. Let $s(x)$ denote the sigmoid function of this logistic regression model. As with probability-based selection policy, we accept only those mentions for

which $s(x)$ is greater or equal to a specified threshold hyperparameter. The threshold is set to maximize the F1-score on the validation set.

Algorithm 1 A Bidirectional Iterative Algorithm

```
function BidirectionalIterativeAlg(x)
    f_out ← outside-in NER model
    f_in ← inside-out NER model
    g ← selection policy function
    S_out ← IterativePrediction(f_out, x)
    S_in ← IterativePrediction(f_in, x)
    S ← g(S_out, S_in)
    return S
end function

function IterativePrediction(f, x)
    S_0 ← ∅, i ← 0
    repeat
        i ← i + 1
        y ← f(x, S_{i−1})
        S_i ← S_{i−1} ∪ y
    until {S_i \ S_{i−1}} = ∅
    return S_i
end function
```

A summary of our approach is demonstrated in Algorithm 1, in which *IterativePrediction* is a prediction method for the single model, f described in the previous subsection, and *BidirectionalIterativeAlg* is a function combining predictions of both models to produce the set of entity mentions, S.

IV. EXPERIMENTS

Our experiments were conducted on four nested named entity recognition datasets: GENIA [5] (biomedical domain), NNE [2] (news domain), PolEval [33] (mixed texts, a Polish corpus), and GermEval [34] (news and Wikipedia, a German corpus). Each dataset was split into three parts: training, validation, and test sets. Where possible, we used the official splits provided by the authors of the datasets, or the most common splits from existing literature. The model is trained using the training set. The validation set is utilized for hyperparameter selection, early stopping, and finding the optimal threshold parameters for probability-based and linear classifier selection policies. The test set is used only for the final evaluation, and for reporting the results of the model. For each of the four datasets, we repeated the training and evaluation procedure three times. The reported precision, recall, and F1-score values are averaged over three runs. We did this to counter the effect of randomness on the training procedure. Since random initialization of the network's weights and shuffling of batches during training affects the final results of the model, reporting a single result might misrepresent the actual performance of our approach. We compared the results of our algorithm with other neural and non-neural methods, using scores provided by their authors or, in the case of NNE dataset, our own evaluation based on the released source code of the models.

The models were trained with the mini-batch gradient descent algorithm and a batch size of 32. We employed a training scheduler with a decreasing learning rate and early stopping, based on the validation set performance. More specifically, the initial learning rate was set to 0.1, and halved after three consecutive epochs with no improvement in validation loss. The training was stopped after reaching a learning rate of 0.0001, and the best model checkpoint (that with the lowest validation loss) was selected as the final model for evaluation.

A. HYPERPARAMETER SELECTION

In this subsection, we discuss the process of hyperparameter selection for our model. We consider the choice of the number of BiLSTM layers, the hidden size of a BiLSTM layer and a contextual word representation among the most commonly used pre-trained language models: Flair [35], [36], ELMo [37], and BERT [29]. In the case of traditional flat NER problems, the Flair model proposed by Akbik *et al.* [35] proved to be particularly effective, setting new state-of-the-art results on several popular named entity recognition datasets, such as CoNLL-2003 [38], OntoNotes [39], and WNUT 2017, as well as the CoNLL datasets for the German and Dutch languages. The initial configuration of our inside-out and outside-in models followed, where applicable, the hyperparameters of the best performing architecture from Akbik *et al.* [35]. The reason for this decision was that detecting nested entities with a single iterative model is comparable to applying multiple flat NER steps on the same input. Therefore, the optimal hyperparameters for an iterative model should not be different from the those of a traditional NER model. Each model by default had two bidirectional LSTM layers with a hidden size of 256. During training, we used word-level dropout with a probability of 0.05, and variational dropout [40] with a probability of 0.5 before each BiLSTM layer.

To validate our assumptions, we conducted a series of pretests. In those experiments, we trained separate inside-out and outside-in iterative models with different hyperparameter values. We used the NNE dataset, which is the largest English language nested named entity recognition dataset with a deep and complex structure of nested entities. Only the training and validation splits were utilized in this experiment - we trained the models with the training part of the data, and assessed the results on the validation part. As with the other experiments, we repeated the procedure three times and reported the average F1-scores. We tested three different word representations, four LSTM hidden sizes (from 128 to 512 neurons), and architectures from one to four stacked BiLSTM layers. We consider only one hyperparameter at a time. In each experiment, the initial values have been used for the remaining hyperparameters.

The results of the hyperparameter selection are shown in Table 1. We provide the F1-scores for both the inside-out

TABLE 1. The results of inside-out and outside-in iterative models trained with different hyperparameters. We examine the effect of the word representation used, the number of LSTM layers and the hidden size of a single LSTM cell.

Value	Outside-in model	Inside-out model
Contextual word representation		
Flair	**93.68**	**93.79**
BERT	92.22 (-1.46)	92.64 (-1.15)
ELMo	93.48 (-0.20)	93.48 (-0.31)
LSTM hidden size		
128	92.95 (-0.83)	93.11 (-0.74)
256	**93.78**	**93.85**
384	92.63 (-0.15)	93.58 (-0.27)
512	92.53 (-0.25)	93.72 (-0.13)
Number of BiLSTM layers		
1	93.47 (-0.31)	93.66 (-0.07)
2	**93.78**	**93.73**
3	93.37 (-0.41)	93.66 (-0.07)
4	93.34 (-0.44)	93.49 (-0.24)

and outside-in models. For suboptimal hyperparameter values, we additionally include an absolute difference to the best result. In the word representation experiment, we used the output of the final layer of each model as input features. The pre-trained models employed in the evaluation included the large English language ELMo model [37], the large BERT cased model [29], and Flair embeddings trained on a news corpus [36]. With our neural iterative model, Flair proved to be the most effective representation. The results are therefore consistent with studies on flat NER. The advantage of Flair in named entity recognition is often explained by the fact that it is a fully character-based language model, unlike ELMo and BERT. The importance of character level features in NER has been highlighted in several previous publications [41]–[43]. In the case of LSTM hidden size, we can observe a clear performance drop for the lowest value (128), which indicates an underparametrized model. The model achieves the best results with a hidden size of 256, but the drop in performance for higher dimensionalities is not as substantial as for the lower value. The number of BiLSTM layers affects the model in a similar manner, although using a single BiLSTM layer has a lesser impact on the model than reducing the hidden size. The performance of the outside-in model also appears to be more sensitive to the number of layers than that of the inside-out model.

We use the selected hyperparameters for all experiments described in this section, with the exception of the PolEval dataset, for which Dadas [44] have performed a comprehensive hyperparameter analysis. To make a direct comparison with their state-of-the art model, we apply the same hidden size, number of layers, and word representations. More details can be found in the PolEval subsection.

B. GENIA

GENIA is one of the most popular benchmarks for nested named entity recognition methods. The corpus contains 2000 abstracts from MEDLINE/PubMed, a biological and medical bibliographic database. Texts are annotated with entities coming from the biomedical domain, such as names of substances, tissues, cell types, or viruses. The dataset includes 51 546 outer named entities, of which 4895 (9.5%) contain nested entities. The maximum level of nesting is equal to 4. To make a fair comparison with other publications, we pre-processed the corpus following the guidance of Finkel and Manning [12]. Their procedure, which involved splitting the dataset and reducing the number of entity types, was reused by most of the studies included in our comparison. The remainder of this paragraph describes the precise procedure we applied to the corpus. The dataset was split in the following way: the first 90% of the corpus was selected for training, and the remaining 10% for evaluation. We divided the first part of the corpus again, and used 90% of it as a training set and 10% as a validation set. We also reduced the number of classes by collapsing all *DNA* subtypes into *DNA*; all *RNA* subtypes into *RNA*; and all *protein* subtypes into *protein*. We maintained the *cell line* and *cell type* classes, and removed all other entities. Only five possible entity types remained of the original 36.

For this experiment, we used domain-specific pre-trained word representations. As a word embedding layer, we used 400-dimensional Word2Vec [45] trained on biomedical abstracts from MEDLINE/PubMed library. We also utilized contextual string embeddings [35] had been trained on the same data source.

Table 2 demonstrates the performance of our algorithm using different selection policies, in comparison with other methods. We can observe that the selection function based on logistic regression achieves the highest F1-score while also being the most balanced in terms of precision-recall tradeoff. In specific practical cases, it could be beneficial to use an

TABLE 2. Results on GENIA dataset.

Model	P	R	F1
Non-neural methods			
Alex et al. [11]	74.5	66.0	70.0
Finkel and Manning [12]	75.4	65.9	70.3
Lu and Roth [13]	74.2	66.7	70.3
Muis and Lu [14]	75.4	66.8	70.8
Wang and Lu [15]	76.2	67.5	71.6
Neural methods			
Xu et al. [17]	71.2	64.3	67.6
Katiyar and Cardie [16]	79.8	68.2	73.6
Ju et al. [24]	78.5	71.3	74.7
Wang et al. [22]	78.0	70.2	73.9
Wang and Lu [15]	77.0	73.3	75.1
Sohrab and Miwa [18]	**93.2**	64.0	77.1
Marinho et al. [23]	74.0	72.0	73.0
Lin et al. [19]	75.8	73.9	74.8
Zheng et al. [25]	75.9	73.6	74.7
Sun et al. [27]	77.4	74.9	76.2
Shibuya and Hovy [28]	78.7	75.7	77.2
Our method with different selection policies			
Outside-in model only	75.4	79.1	77.2
Inside-out model only	77.9	75.7	76.8
Model intersection	80.4	73.8	77.0
Model union	73.4	**81.0**	77.0
Probability-based selection	78.5	75.5	77.0
Linear classifier selection	77.9	76.8	**77.3**

intersection or union policy which maintains an acceptable F1-score, but maximizes either the precision or recall of the algorithm. In the case of this dataset, there are no significant differences in F1-score between evaluated policies. This may be due to the fact that in this dataset relatively few entities are nested, and the performance is primarily dependent on the effectiveness of flat rather than nested entity recognition. Of other approaches, only the deep exhaustive model by Sohrab and Miwa [18] and recursive Viterbi-decoding approach by Shibuya and Hovy [28] offer comparable results.

In addition to standard evaluation, we conducted an ablation study for GENIA, the results of which are shown in Table 3. The first three positions correspond to the best performing selection policy from Table 2, a single outside-in model and a single inside-out model. We also demonstrate the effect of running the algorithm in a non-iterative manner (*No iteration*). In this case, we run both models only once to predict a set of the innermost and outermost flat entity mentions, and combine those predictions using linear classifier selection policy. It is evident from the results that disabling iteration has a minor effect on the F1-score. We also studied the effect of replacing our neural models with simpler baselines which do not include contextual word representations from the language model (*No pre-trained language model*). The baseline was derived from a well-known BiLSTM-CRF architecture by Lample *et al.* [41], in which word representation is constructed from static word embeddings and a BiLSTM character-level encoder. Finally, to ensure a fair comparison with all the methods from Table 2, we retrained our model without using any domain-specific word representations (*No biomedical word representation*). In this experiment, we replaced the language model and word embeddings trained on the MEDLINE/PubMed corpus with original character-level LM by Akbik *et al.* [35] trained on the news corpus, and 300-dimensional GloVe embeddings by Pennington *et al.* [46]. To the best of our knowledge, Sohrab and Miwa [18], Zheng *et al.* [25], Sun *et al.* [27] and Shibuya and Hovy [28] used biomedical word representations in their work. Without domain pre-training, our algorithm performs 1.77% worse in terms of F1-score. However, comparing our approach only to those methods that do not use biomedical embeddings, it still achieves the highest F1-score of 75.6%.

TABLE 3. Ablation study for GENIA dataset.

Model	P	R	F1
Full model	77.9	76.8	77.3
Outside-in model	75.4	79.1	77.2 (-0.15)
Inside-out model	77.9	75.7	76.8 (-0.56)
No iteration	77.1	76.9	77.0 (-0.33)
No pre-trained language model	77.4	75.3	76.3 (-1.00)
No biomedical word representation	76.4	74.8	75.6 (-1.77)

C. NNE

The Nested Named Entity (NNE) corpus is a large dataset explicitly targeted for the evaluation of nested named entity recognition methods. It contains annotations of the Wall Street Journal portion of the Penn Treebank (PTB). The total number of named entities is 279 795, of which 118 525 are outer entities. Over 60% of the outer entities contain nesting, with a maximum depth of six levels. The dataset uses a rich, fine-grained ontology of 112 entity types. The most commonly occurring entity types are typical for news-related NER datasets, and include classes such as names of people, organizations, geographical objects, dates, or cardinals.

For this experiment, we used contextual string embeddings [35] trained on the news corpus, and 300-dimensional GloVe embeddings trained on Wikipedia and the Gigaword corpus [46]. We also used the original train, test, and validation splits suggested by the authors of the dataset.

The NNE dataset was introduced recently in a study by Ringland *et al.* [2] which also included the evaluation of three baseline models, in addition to a transition-based model by Wang *et al.* [22], and a hypergraph based model by Wang and Lu [15]. We compare our algorithm with these approaches in Table 4. The three baselines shown in the table are based on the BiLSTM-CRF neural architecture [41], and were trained to detect only the outermost or innermost entity mentions. Baselines (a) and (b) are simple flat NER models detecting the outermost and innermost entities respectively. Baseline (c) is a concatenation of predictions from the first two models. In addition, we have retrained and performed an evaluation of other recent methods, implementations of which are publicly available: Zheng *et al.* [25], Ju *et al.* [24], and Shibuya and Hovy [28]. We can see that the linear classifier policy is again the best performing one, achieving an F1-score of 94.1%, an improvement of 0.6% compared to the work of

TABLE 4. Results on NNE dataset. Ringland *et al.* [2] evaluated three variants of baseline model based on BiLSTM-CRF architecture: (a) a flat NER model detecting only the outermost entities; (b) a flat NER model detecting only the innermost entities; and (c) a combination of predictions of those two models. Our comparison also includes other known methods, retrained and evaluated using publicly available source codes. † denotes methods evaluated by Ringland *et al.* [2], ‡ denotes methods which we evaluated.

Model	P	R	F1
Neural methods			
Ringland et al. [2] (a) †	89.9	38.0	53.5
Ringland et al. [2] (b) †	93.8	62.0	74.7
Ringland et al. [2] (c) †	92.2	85.8	88.9
Wang et al. [22] †	77.4	70.1	73.6
Wang and Lu [15] †	91.8	91.0	91.4
Zheng et al. [25] ‡	89.3	88.5	88.9
Ju et al. [24] ‡	92.3	90.0	91.1
Shibuya and Hovy [28] ‡	93.3	93.7	93.5
Our method with different selection policies			
Outside-in model only	93.7	93.0	93.3
Inside-out model only	93.4	93.8	93.6
Model intersection	**95.0**	91.6	93.3
Model union	92.2	**95.2**	93.7
Probability-based selection	93.8	94.0	93.9
Linear classifier selection	94.2	93.9	**94.1**

Shibuya and Hovy [28], and a significant increase over all other methods.

As in the case of GENIA, we conducted an ablation study of our model on NNE, the results of which are presented in Table 5. The list of compared approaches is the same as in the previous study, except for the last model, which is not applicable here. It is worth noting that, unlike for GENIA, there is a significant difference in F1-score between the full model and the non-iterative version. This suggests that the best performing flat NER models would not be able to reach the level of nested NER methods for this dataset. In cases which the predicted structures are complex, iterativeness becomes a crucial feature of the algorithm. As for the other variants of our model, even after the drop in F1-score, they still outperform the methods evaluated in Ringland *et al.* [2] by 2% or more.

TABLE 5. Ablation study for NNE dataset.

Model	P	R	F1
Full model	94.2	93.9	94.1
Outside-in model	93.7	93.0	93.3 (-0.76)
Inside-out model	93.4	93.8	93.6 (-0.45)
No iteration	94.3	86.4	90.2 (-3.90)
No pre-trained language model	93.7	92.9	93.3 (-0.76)

D. PolEval (2018)

PolEval is a competition for natural language processing in Polish language inspired by SemEval.[1] It is a regular event in which a variety NLP challenges are held. PolEval 2018, which lasted from June to August, included an evaluation of named entity recognition methods [33]. The task was based on two datasets: NKJP (The National Corpus of Polish [50]), which was used as a training corpus, and an additional evaluation corpus created specifically for the challenge. Both datasets were manually annotated using the same guidelines with 14 entity types. Of these entity types, some can appear both as top-level and as nested entities, and some are defined as subtypes, and therefore can appear only as nested entities. The combination of training and evaluation sets contains 121 707 sentences annotated with 127 937 named entity mentions. Approximately 42% of outer entities contain nesting, and the maximum depth of nested entities is 6. The maximum length of a named entity is 16 words. The corpus consists of various types of text, including literature (contemporary and historical), reportage, press releases, scientific texts, legal texts, internet discussions, and transcriptions of speech. The types of entities include names of real and fictional people, names of organizations, names of geographical and man-made structures, geopolitical names, dates, and times.

Since there was no official validation set for this task, we decided to use 10% of the training corpus for validation. For this experiment, we utilized pre-trained word representations for Polish language: the ELMo language model and 100-dimensional Word2Vec embeddings from Dadas [44]. We also changed the number of LSTM layers to three and the hidden representation size to 100, in accordance with the optimal hyperparameters used in their study.

Table 6 highlights the results of our method in comparison with the three best solutions from the PolEval competition [47]–[49], and two variants of the model introduced in Dadas [44]: one with static and one with contextual word representation. In the case of PolEval, we used the official metrics proposed by the organizers rather than precision, recall, and F1-score. The metrics are as follows: exact match score (ES), which is based on a standard F1-score measure; overlap score (OS), based on an F1-score computed for partial matches of entity mentions; and final score (FS), which is a weighted average of the two aforementioned measures, in which the weight is equal to 0.2 for exact and 0.8 for overlap score. We can observe that for this dataset, the differences between the individual selection policies are minor, with the exception of the model union, for which the final score is noticeably worse. Model intersection, probability-based selection, and linear classifier selection all perform equally well. Comparing our iterative approach with other methods, we can observe only a slight increase in overlap score over the best performing model (0.2%), but a significant improvement in exact match score (2.5%). This may be due to the fact that the other approaches evaluated on this task could only detect nested entities to a limited extent, usually up to a certain level of nesting. Our method can handle a set of deeply nested entities that were undetectable with simpler models.

TABLE 6. Results on the PolEval dataset. Unlike in the case of other tasks, we have not compared methods using precision, recall, and F1-score. We report official metrics from the PolEval competition instead, produced using a script provided by the organizers. The metrics are: the exact match score (ES), the overlap score (OS), and the final score (FINAL). The final score is based on a combination of the exact and overlap scores. Final score = 0.8 * Overlap score + 0.2 * Exact match score.

Model	ES	OS	FINAL
Marcińczuk et al. [47]	77.8	81.8	81.0
Marcińczuk et al. [48]	82.2	85.9	85.1
Borchmann et al. [49]	82.6	87.7	86.6
Dadas [44] (Word2Vec)	80.2	85.7	84.6
Dadas [44] (ELMo)	84.5	88.8	87.9
Our method with different selection policies			
Outside-in model only	86.7	88.8	88.4
Inside-out model only	86.4	88.7	88.2
Model intersection	**87.0**	**89.0**	**88.6**
Model union	86.1	88.1	87.7
Probability-based selection	86.9	89.0	88.6
Linear classifier selection	**87.0**	**89.0**	**88.6**

E. GermEval (2014)

GermEval is a series of natural language processing challenges in German language. A nested named entity recognition task formed part of the series in 2014. For this purpose, a dataset consisting of 31 297 sentences annotated with 12 named entity types was created [34]. From the point of view of the nested NER, this is a fairly simple dataset, as there

[1] https://en.wikipedia.org/wiki/SemEval

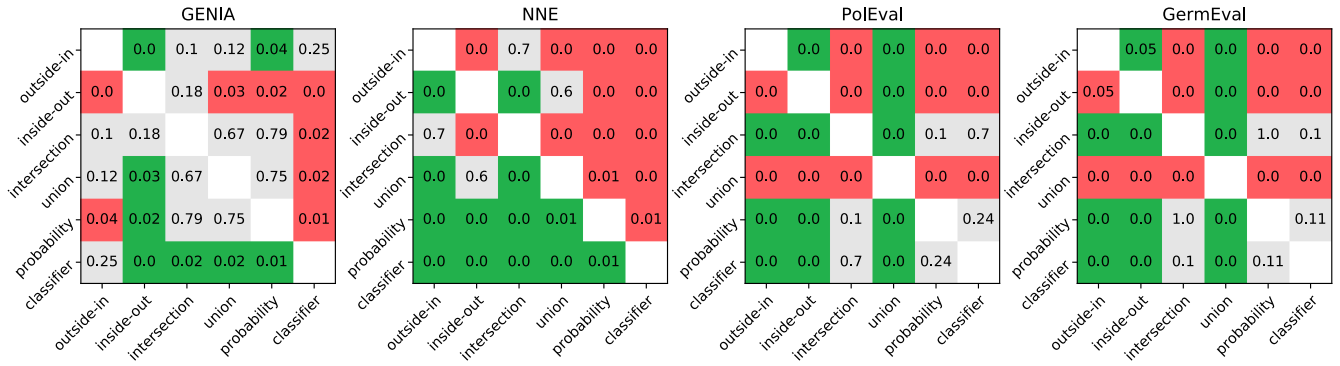

FIGURE 4. Statistical significance of differences between selection policies for each of the evaluated tasks.

are only two levels of nesting. Of the 37 832 outer entity mentions, only 3173 contain nested entities (approximately 8%). The maximum length of a named entity is 20 words. The corpus is composed of text from the German language version of Wikipedia and German online news. The set of entity types is typical for news corpora, and includes names of locations, organizations, people, and other entities.

For word representations in this experiment, we used German contextual string embeddings from Akbik *et al.* [35], and German FastText word embeddings [53]. We also used the same train, test and validation dataset splits as in the GermEval 2014 competition.

The results of our evaluation are shown in Table 7. We compare our approach with the top three models from the GermEval competition: ExB Group [51], UKP [52], and MoSTNER, as well as recent nested named entity recognition methods: Sohrab and Miwa [18], Ju *et al.* [24], and Zheng *et al.* [25]. We can see that in the case of this dataset, our model achieves the best F1-score by a significant margin. However, due to the fact that this is not a challenging task in the context of nested NER, we cannot attribute this advantage to the architecture of our framework. This is most likely the effect of utilizing the German language model pre-trained by Akbik *et al.* [35], as all other methods use either static word embeddings or handcrafted features as their word representations.

TABLE 7. Results on GermEval dataset.

Model	P	R	F1
ExB Group (1st place) [51]	78.1	74.8	76.4
UKP (2nd place) [52]	79.5	71.1	75.1
MoSTNER (3rd place)	79.2	65.3	71.6
Sohrab and Miwa [18]	75.0	60.8	67.2
Ju et al. [24]	72.9	61.5	66.7
Zheng et al. [25]	74.5	69.1	71.7
Our method with different selection policies			
Outside-in model only	82.9	82.9	82.9
Inside-out model only	82.4	83.2	82.8
Model intersection	**86.6**	80.6	**83.5**
Model union	79.3	**85.5**	82.3
Probability-based selection	86.2	80.9	**83.5**
Linear classifier selection	86.4	80.5	83.3

F. DISCUSSION

Each of the experiments performed in this study was repeated three times. When comparing our model to other approaches, we reported average precision, recall, and micro F1-scores. We also conducted a series of unpaired Student's t-tests to compare the selection policies proposed in this study, and check if the reported differences in results were statistically significant. The statistical significance, α of all tests indicated is judged at the 0.05 significance level. Figure 4 offers a summary of those tests for the datasets evaluated. Each row in the grid represents a comparison between a particular selection policy and all other policies. Green color indicates that the mean F1-score (with the exception of PolEval, for which a custom task-specific score is reported) for the current policy (row) was higher than the compared policy (column,) and the result was statistically significant. Red color indicates that the score was lower. Gray color indicates that the null hypothesis could not be rejected. The number in each cell is equal to the p-value for that specific test.

Based on the evaluation results on all four datasets we can conclude that linear classifier policy works well in most cases, and is an appropriate default choice for our algorithm. In 15 out of 20 cases, it was found to be significantly better than in other selection methods. It was also the only selection policy which outperformed the previous approaches on all four datasets, including GENIA in which the performance of our method was the closest to the results of previous state-of-the-art methods. Among the other selection policies, fixed probability policy had a comparable performance on three of the tasks, but was slightly worse for GENIA. The performance gap increased in favour of our method in cases of more demanding datasets with complex nested structures, and a large number of entity classes, such as NNE or PolEval. In these cases, at least four of the six policies evaluated proved to be better than other the approaches. These results, in addition to those of ablation studies, suggest that the iterative approach is especially effective for deeply nested

entities. This may be due to the fact that the predictions of subsequent layers of named entities are explicitly conditioned on previous layers. This allows our model to learn both the horizontal (between words in the sentence) and vertical (between iterations) relationships.

It is also worth noting that performance differences between outside-in and inside-out models are task-specific - for some problems one works better: for others the other. We also wish to highlight that from a practical point of view, the ability to choose a selection policy for a particular application is an advantage of our algorithm. For example, in some cases it could be useful to choose intersection or union policy if the precision or recall of the solution is more important than a balanced F1-score. For many other known models, there is no easy method of controlling the precision/recall balance.

V. CONCLUSION

In this paper, we have proposed an iterative deep learning model for nested named entity recognition in which the representation of words is constructed from the character and word-level features, and a vector of encoded entity types identified in the previous iterations. We then proposed a bidirectional algorithm that combines the predictions of two iterative models using a pre-defined selection policy. We have presented several selection policies that can be used with the algorithm. Since the performance of specific selection policies appears to be dependent on the features of the dataset, an interesting direction for further development would be to identify these features, and automatically recommend the best selection policy for a particular dataset type. We conducted experiments on four datasets sourced from different domains and differing in terms of frequency and structure of nested mentions. The evaluation of our method shows that it is competitive for tasks involving mostly flat mentions with occasional nesting, as well as tasks with a large number of complex nested structures.

REFERENCES

[1] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," in *Proc. 27th Int. Conf. Comput. Linguistics*. Santa Fe, NM USA: Association Computational Linguistics, Aug. 2018, pp. 2145–2158. [Online]. Available: https://www.aclweb.org/anthology/C18-1182

[2] N. Ringland, X. Dai, B. Hachey, S. Karimi, C. Paris, and J. R. Curran, "NNE: A dataset for nested named entity recognition in english newswire," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*. Stroudsburg, PA, USA: Association Computational Linguistics, 2019, pp. 5176–5181.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Mach. Learn.* San Francisco, CA, USA: Morgan Kaufmann, 2001, pp. 282–289. [Online]. Available: http://dl.acm.org/citation.cfm?id=645530.655813

[5] T. Ohta, Y. Tateisi, and J.-D. Kim, "The GENIA corpus: An annotated research abstract corpus in molecular biology domain," in *Proc. 2nd Int. Conf. Human Lang. Technol. Res.* San Francisco, CA, USA: Morgan Kaufmann, 2002, pp. 82–86. [Online]. Available: http://dl.acm.org/citation.cfm?id=1289189.1289260

[6] D. Shen, J. Zhang, G. Zhou, J. Su, and C.-L. Tan, "Effective adaptation of a hidden Markov model-based named entity recognizer for biomedical domain," in *Proc. ACL Workshop Natural Lang. Process. Biomed.* Stroudsburg, PA, USA: Association Computational Linguistics, 2003, pp. 49–56.

[7] J. Zhang, D. Shen, G. Zhou, J. Su, and C.-L. Tan, "Enhancing HMM-based biomedical named entity recognition by studying special phenomena," *J. Biomed. Informat.*, vol. 37, no. 6, pp. 411–422, Dec. 2004.

[8] Z. GuoDong and S. Jian, "Exploring deep knowledge resources in biomedical name recognition," in *Proc. Int. Joint Workshop Natural Lang. Process. Biomed. Appl. (JNLPBA)*. Stroudsburg, PA, USA: Association Computational Linguistics, 2004, pp. 96–99.

[9] G. Zhou, "Recognizing names in biomedical texts using mutual information independence model and SVM plus sigmoid," *Int. J. Med. Informat.*, vol. 75, no. 6, pp. 456–467, Jun. 2006.

[10] B. Gu, "Recognizing nested named entities in GENIA corpus," in *Proc. HLT-NAACL BioNLP Workshop Linking Natural Lang. Biol. (LNLBioNLP)*. Stroudsburg, PA, USA: Association Computational Linguistics, 2006, pp. 112–113.

[11] B. Alex, B. Haddow, and C. Grover, "Recognising nested named entities in biomedical text," in *Proc. Workshop BioNLP Biol., Transl., Clin. Lang. Process. (BioNLP)*. Stroudsburg, PA, USA: Association Computational Linguistics, 2007, pp. 65–72.

[12] J. R. Finkel and C. D. Manning, "Nested named entity recognition," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Stroudsburg, PA, USA: Association Computational Linguistics, 2009, pp. 141–150.

[13] W. Lu and D. Roth, "Joint mention extraction and classification with mention hypergraphs," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Stroudsburg, PA, USA: Association Computational Linguistics, 2015, pp. 857–867.

[14] A. O. Muis and W. Lu, "Labeling gaps between words: Recognizing overlapping mentions with mention separators," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Stroudsburg, PA, USA: Association Computational Linguistics, 2017, pp. 2608–2618.

[15] B. Wang and W. Lu, "Neural segmental hypergraphs for overlapping mention recognition," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 204–214.

[16] A. Katiyar and C. Cardie, "Nested named entity recognition revisited," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2018, pp. 861–871.

[17] M. Xu, H. Jiang, and S. Watcharawittayakul, "A local detection approach for named entity recognition and mention detection," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1. Stroudsburg, PA, USA: Association Computational Linguistics, 2017, pp. 1237–1247.

[18] M. G. Sohrab and M. Miwa, "Deep exhaustive model for nested named entity recognition," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 2843–2849.

[19] H. Lin, Y. Lu, X. Han, and L. Sun, "Sequence-to-nuggets: Nested entity mention detection via anchor-region networks," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*. Florence, Italy: Association Computational Linguistics, Jul. 2019, pp. 5182–5192. [Online]. Available: https://www.aclweb.org/anthology/P19-1511

[20] C. Xia, C. Zhang, T. Yang, Y. Li, N. Du, X. Wu, W. Fan, F. Ma, and P. Yu, "Multi-grained named entity recognition," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*. Florence, Italy: Association Computational Linguistics, Jul. 2019, pp. 1430–1440. [Online]. Available: https://www.aclweb.org/anthology/P19-1138

[21] Y. Chen, Y. Wu, Y. Qin, Y. Hu, Z. Wang, R. Huang, X. Cheng, and P. Chen, "Recognizing nested named entity based on the neural network boundary assembling model," *IEEE Intell. Syst.*, vol. 35, no. 1, pp. 74–81, Jan. 2020.

[22] B. Wang, W. Lu, Y. Wang, and H. Jin, "A neural transition-based model for nested mention recognition," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 1011–1017.

[23] Z. Marinho, A. Mendes, S. Miranda, and D. Nogueira, "Hierarchical nested named entity recognition," in *Proc. 2nd Clin. Natural Lang. Process. Workshop*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 28–34.

[24] M. Ju, M. Miwa, and S. Ananiadou, "A neural layered model for nested named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2018, pp. 1446–1459.

[25] C. Zheng, Y. Cai, J. Xu, H.-F. Leung, and G. Xu, "A boundary-aware neural model for nested named entity recognition," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*. Hong Kong, China: Association Computational Linguistics, Nov. 2019, pp. 357–366. [Online]. Available: https://www.aclweb.org/anthology/D19-1034

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[27] L. Sun, F. Ji, K. Zhang, and C. Wang, "Multilayer ToI detection approach for nested NER," *IEEE Access*, vol. 7, pp. 186600–186608, 2019.

[28] T. Shibuya and E. Hovy, "Nested named entity recognition via second-best sequence learning and decoding," 2019, *arXiv:1909.02250*. [Online]. Available: http://arxiv.org/abs/1909.02250

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1. Minneapolis, Minnesota: Association Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[30] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, Jun. 1989.

[31] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. kavukcuoglu, O. Vinyals, and A. Graves, "Conditional image generation with pixelCNN decoders," in *Proc. Adv. Neural Inf. Process. Syst.*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 4790–4798. [Online]. Available: http://papers.nips.cc/paper/6527-conditional-image-generation-with-pixelcnn-decoders.pdf

[32] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*. [Online]. Available: http://arxiv.org/abs/1609.03499

[33] A. Wawer and E. Malek, "Results of the poleval 2018 shared task 2: Named entity recognition," in *Proc. PolEval 2018 Workshop*. Warsaw, Poland: Institute Computer Science, Polish Academy of Sciences, 2018, pp. 53–62. [Online]. Available: http://poleval.pl/files/poleval2018.pdf

[34] D. Benikova, C. Biemann, and M. Reznicek, "NoSta-D named entity annotation for German: Guidelines and dataset," in *Proc. 9th Int. Conf. Lang. Resour. Eval. (LREC)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 2524–2531.

[35] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proc. COLING, 27th Int. Conf. Comput. Linguistics*, 2018, pp. 1638–1649.

[36] A. Akbik, T. Bergmann, and R. Vollgraf, "Pooled contextualized embeddings for named entity recognition," in *Proc. NAACL-HLT, Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics*. Stroudsburg, PA, USA: Association Computational Linguistics, 2019, pp. 724–728.

[37] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2018, pp. 2227–2237.

[38] E. F. Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proc. 7th Conf. Natural Lang. Learn. (HLT-NAACL)*, 2003, pp. 142–147. [Online]. Available: https://www.aclweb.org/anthology/W03-0419

[39] R. Weischedel, S. Pradhan, L. Ramshaw, M. Palmer, N. Xue, M. Marcus, A. Taylor, C. Greenberg, E. Hovy, R. Belvin, and A. Houston, "Ontonotes release 5.0," in *Proc. Linguistic Data Consortium*, Philadelphia, PA, USA, vol. 23, 2013, p. 53.

[40] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1019–1027.

[41] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.* San Diego, CA, USA: Association Computational Linguistics, Jun. 2016, pp. 260–270. [Online]. Available: https://www.aclweb.org/anthology/N16-1030

[42] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2016, pp. 1064–1074.

[43] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks," 2017, *arXiv:1707.06799*. [Online]. Available: http://arxiv.org/abs/1707.06799

[44] S. Dadas, "Combining neural and knowledge-based approaches to named entity recognition in polish," in *Proc. Int. Conf. Artif. Intell. Soft Comput.* Berlin, Germany: Springer, 2019, pp. 39–50.

[45] R. McDonald, G. Brokos, and I. Androutsopoulos, "Deep relevance ranking using enhanced document-query interactions," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 1849–1860.

[46] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[47] M. Marcińczuk, J. Kocoń, and M. Janicki, "Liner2–a customizable framework for proper names recognition for polish," in *Intelligent Tools for Building a Scientific Information Platform*. Berlin, Germany: Springer, 2013, pp. 231–253.

[48] M. Marcińczuk, J. Kocoń, and M. Gawor, "Recognition of named entities for polish-comparison of deep learning and conditional random fields approaches," in *Proc. PolEval Workshop*, M. Ogrodniczuk and L. Kobyliński, Eds. Warsaw, Poland: Institute Computer Science, Polish Academy of Science, 2018, pp. 77–92.

[49] L. Borchmann, A. Gretkowski, and F. Graliński, "Approaching nested named entity recognition with parallel LSTM-CRFs," in *Proc. AI NLP Workshop*, 2018, pp. 63–73.

[50] A. Przepiórkowski, M. Banko, R. L. Górski, and B. Lewandowska-Tomaszczyk, *National Corpus of Polish*. Warsaw, Poland: Polish Scientific Publishers PWN, 2012. [Online]. Available: http://nkjp.pl/settings/papers/NKJP_ksiazka.pdf

[51] C. Hänig, S. Thomas, and S. Bordag, "Modular classifier ensemble architecture for named entity recognition on low resource systems," in *Proc. GermEval*, 2014, pp. 1–4.

[52] N. Reimers, J. Eckle-Kohler, C. Schnober, J. Kim, and I. Gurevych, "Germeval-2014: Nested named entity recognition with neural networks," in *Proc. GermEval*, 2014, pp. 1–4. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:gbv:hil2-opus-3023

[53] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proc. 11th Int. Conf. Lang. Resour. Eval. (LREC)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018, pp. 1–5. [Online]. Available: https://www.aclweb.org/anthology/L18-1550

SŁAWOMIR DADAS received the M.S. degree in computer science from the Warsaw School of Information Technology, Warsaw, Poland, under the auspices of the Polish Academy of Sciences, Warsaw. He is currently pursuing the Ph.D. degree in machine learning. He has been with the National Information Processing Institute, Warsaw, since 2011, where he was involved in the design and implementation of information systems incorporating machine learning solutions, primarily in the field of natural language processing. He works as the Leader of a Research and Development Team. His research interests include natural language processing, applications of machine learning in scientometric research, distributed computing, algorithms, and data structures.

JAROSŁAW PROTASIEWICZ received the Ph.D. degree in computer science with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. He is currently an Assistant Professor and the Head of the National Information Processing Institute, Warsaw. His research interests include agile project management, software design and development, big data, machine learning, and bio-inspired algorithms.