Intelligent Data Analysis 14 (2021) 1–21
IOS Press

HELD: Hierarchical Entity-Label Disambiguation in Named Entity Recognition Task Using Deep Learning

Bárbara Stéphanie Neves Oliveira [a,*], Andreza Fernandes de Oliveira [a], Vinicius Monteiro de Lira [b] , Ticiana Linhares Coelho da Silva [a] and José Antônio Fernandes de Macêdo [a]
[a] *Insight Data Science Lab, Federal University of Ceará, Ceará, Brazil*
E-mails: barbaraneves@insightlab.ufc.br, andrezafernandes@insightlab.ufc.br, ticianalc@insightlab.ufc.br, jose.macedo@insightlab.ufc.br
[b] *Institute of Information Science and Technologies, National Research Council, Pisa, Italy*
E-mail: vinicius.monteirodelira@isti.cnr.it

Abstract. Named Entity Recognition (NER) is a challenging learning task of identifying and classifying entity mentions in texts into predefined categories. In recent years, deep learning (DL) methods empowered by distributed representations, such as word- and character-level embeddings, have been employed in NER systems. However, for information extraction in Police narrative reports, the performance of a DL-based NER approach is limited due to the presence of fine-grained ambiguous entities. For example, given the narrative report "Anna stole Ada's car", imagine that we intend to identify the VICTIM and the ROBBER, two sub-labels of PERSON. Traditional NER systems have limited performance in categorizing entity labels arranged in a hierarchical structure. Furthermore, it is unfeasible to obtain information from knowledge bases to give a disambiguated meaning between the entity mentions and the actual labels. This information must be extracted directly from the context dependencies. In this paper, we deal with the Hierarchical Entity-Label Disambiguation problem in Police reports without the use of knowledge bases. To tackle such a problem, we present HELD, an ensemble model that combines two components for NER: a BLSTM-CRF architecture and a NER tool. Experiments conducted on a real Police reports dataset show that HELD significantly outperforms baseline approaches.

Keywords: Fine-grained entity labels, Hierarchical Entity-Label Disambiguation using Context, Named Entity Recognition, Deep Learning, Police reports domain

1. Introduction

Named Entity Recognition (NER) is a challenging learning task of identifying named-entity mentions in texts and classifying them into predefined categories, such as person, location, and organization. NER is an essential task for a variety of Natural Language Processing (NLP) applications, such as topic detection and speech recognition. Early NER systems often require significant human effort in carefully designing rules or features. This trend has motivated the use of deep learning (DL) in NER over the past few years due to requiring minimum feature engineering. Besides, DL-based NER approaches have been empowered by distributed representations, which consider word- and character-level embeddings as well

[*]Corresponding author. E-mail: barbaraneves@insightlab.ufc.br.

as the incorporation of additional features like orthographic features and language-specific knowledge resources, such as gazetteers [1]. However, language-specific resources and features are costly to develop, especially for new languages and new domains, making NER a challenge to adapt [2].

Furthermore, traditional NER tasks focus on a specific set of entity labels and one label per named entity. For relation extraction in domain-specific applications, it is more useful to work with an authentic set of fine-grained labels for the domain [3, 4]. Typically, fine-grained NER tasks focus on a larger number of entity labels arranged in a hierarchical structure, where an entity mention can be assigned to multiple labels. In the attempt for finer granularity, complex knowledge bases have been used to leveraged NER models [5]. Existing systems first extract the entity mentions and then link the mentions to one or more referent entities in a knowledge base [6–8]. Such ability can be performed by Named Entity Disambiguation (NED) methods. For example, given the sentence "*Paris is the capital of France*", NER would pass the mentions of *Paris* and *France* to the NED stage, which would identify *Paris* as the capital city of France and not as something else as a building or even a person (e.g., the businesswoman Paris Hilton). Within this context, *Paris* is a CAPITAL_CITY, a sub-label of CITY, which, in turn, is a sub-label of LOCATION. If a knowledge base has these three matching labels, all of them must be assigned to *Paris*. The knowledge bases for NED are commonly derived from Wikipedia, YAGO, or a complex combination of several resources, including, among others, WordNet and Wiktionary [6–8], or by using linked datasets [9].

However, not all domains have well-defined knowledge bases that provide a background repository to fine-grained named entity disambiguation. For instance, in the Police reports domain, it is impractical to have a knowledge base that describes previously some details about all incidents, such as knowing from the persons involved who is the victim, witness, robber, etc. Besides, each case must be analyzed individually. Consider a part of a real Police narrative report: "*The above-qualified declarant stated that her sister Augusta Dias was beaten and had her car stolen on the day, place, and time mentioned above. The declarant informs that the author was a person known as Augus. Augusta was seriously injured at the incident location. With nothing else to declare.*"[1] Imagine that we intend to identify in the text the VICTIM and the ROBBER, two sub-labels of PERSON. Notice that our objective of recognizing the mentions of *Augusta Dias* and *Augusta* as the VICTIM and *Augus* as the ROBBER is very challenging since both names preserve the same orthographic features. It is unfeasible to derive the information that someone is a VICTIM or a ROBBER from Wikipedia, YAGO, or by using linked datasets as done in previous works [7–9]. Therefore, this information must be extracted from the context dependencies.

In this work, we deal with the Hierarchical Entity-Label Disambiguation problem in Police reports. Our proposed solution, called HELD, enhances a state-of-the-art NER tool by using an ensemble architecture and is a domain-independent approach (i.e., it can be used by various real-world applications or even by NER systems in general domain). We build HELD to be free from knowledge bases, language-specific resources (e.g., gazetteers, word clusters id, and part-of-speech tags), or hand-crafted features (e.g., word spelling and capitalization patterns). HELD is an ensemble model that combines two sequence labeling components for NER: a bidirectional Long-Short Term Memory neural network with a subsequent Conditional Random Field decoding layer (BLSTM-CRF) and an off-the-shelf NER tool from the spaCy library, to learn from the context how to disambiguate fine-grained entity labels.

As stated in [1], there are some strengths of why using DL architectures for the NER task, which explain the considerable number of studies that applied DL-based NER systems and successively advanced the state-of-the-art performance. The same motivations are considered for this work. First, compared to

[1]For reasons of confidentiality and preservation of those involved, the names mentioned in the narrative are fictitious.

feature-based approaches, deep learning is beneficial in discovering hidden features automatically. DL-based models use non-linear activation functions to learn complex features from raw data. In fact, the Hierarchical Entity-Label Disambiguation problem corresponds to a non-linear transformation between the input and output. Second, DL-based models can learn useful representations and underlying patterns from Police reports, saving significant effort in designing features to perform fine-grained label disambiguation.

Several approaches propose NER models derived from LSTM [10–12], by combining BLSTM with CNNs [13] or by solving the problem of word disambiguation [14]. None of these works solves the label disambiguation problem given an input text without the use of knowledge bases. Current fine-grained NER systems also leveraged their models and automatically annotate training corpora with over a hundred labels via knowledge base lookup [3, 15–18]. In [19], the HYENA model performs a top-down hierarchical fine-grained label classification based on an extrinsic study with a NED tool, however, the authors build a set of classifiers that mark entity mentions connected to a knowledge base. Recently, BERT [20] is becoming a new paradigm for NER task as proposed in [21–24]. While in practice, BERT and other contextualized language-model embeddings [25–27] have a prohibitively large number of parameters, require a massive amount of training data and powerful computing resources to ensure promising results for a specific language or domain. Due to the limitation of an available huge dataset of annotated Police reports, we will investigate language models to our problem as future work. The contributions of this paper are as follows:

(1) We introduce a formalization of the Hierarchical Entity-Label Disambiguation problem in Police reports. The formalization defines the hierarchical structure present in the fine-grained named entities of Police reports.
(2) We propose and developed HELD, an ensemble and domain-independent approach that extends a pre-trained NER tool from the spaCy library and a BLSTM-CRF model to solve the Hierarchical Entity-Label Disambiguation problem in Police reports without the use of knowledge bases.
(3) We explore some different approaches for coping with the data imbalance problem present in a real-world manually annotated dataset for the Police domain.
(4) An in-depth study to provide the best word-level representation (pre-trained or domain-specific), that most effectively represent the Police report's vocabulary, for one of the main components in HELD.
(5) An extensive experimental evaluation over a real-world dataset, where we assess the validity of HELD in terms of quality of results. Our proposal can surpass F_1-score comparing to baseline approaches.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 provides the problem definition. Section 4 presents the methodological details of HELD. Section 5 presents the experiments and discusses the experimental evaluation. Finally, Section 6 summarizes this work and discusses future directions.

2. Related Work

Several works have presented models that use well-formatted documents heavily depend on a phrase's local linguistic features, such as capitalization, part-of-speech (POS) tags of previous words, external resources, such as gazetteers, or large dictionaries of entities gathered from Freebase, Wikipedia, and

YAGO, to perform NER and NED tasks [6–8, 28–31]. Other examples include current works that deal with a hundred fine-grained entities that also use knowledge bases to leverage their models, such as automatically annotate training corpora [3, 15–18]. In recent years, deep learning models have attracted attention to solving NER tasks due to require minimal feature engineering, as stated in [1] that reviews the literature based on varying deep learning models for NER.

Apart from the English language, there are many studies on the classification of named entities in documents written in other languages or cross-lingual setting. For example, [32] investigated a deep learning method to recognize clinical entities in Chinese clinical documents using the minimal feature engineering approach. In [33], the authors incorporated dictionaries into deep neural networks for the Chinese named entity recognition task. In addition to Chinese, there also exist studies that have been conducted for named entity detection in documents in other languages. Examples include Portuguese [34], and Japanese [35]. Each language has its characteristics for understanding the fundamentals of the NER task for that language, which makes NER models very challenging to adapt.

Many studies as [36, 37] use a self-attention mechanism to the neural architecture to solve the NER problem in a cross-lingual setting by transferring knowledge from a source language to a target language with few or no labels. Another interesting work is [38], which examines the effects of transfer learning for deep hierarchical recurrent networks across domains, applications, and languages, showing that significant improvements can often be achieved in several tasks, including NER. There has also been a long history of research involving neural networks for entity recognition in documents, even with fine-grained entities. [11] attempted NER with a single direction LSTM network. The work [10] proposes two neural architectures for sequence labeling: one based on bidirectional LSTMs and a CRF model, and the other that constructs and labels segments using a transition-based approach inspired by shift-reduce parsers.

Similarly, [13] and [39] combine bidirectional LSTM with CNNs, while [40] re-implemented the NER model described by [39], adjusting it to work with fine-grained labels for the English language. For Japanese, [40] removed the CNN layer, which previously learns character-level embeddings, to use dictionary (gazetteer feature) and category embeddings. [16] use an LSTM to encode the fine-grained entity mentions representations and a bidirectional LSTM as context encoder, and perform a feature and model level transfer learning. [18] combining token embeddings from the ELMo contextualized language model [25], which are fed into a residual LSTM module, to finally pass the detected entities into the Wikidata knowledge base. [41] uses a CNN over a sequence of word embedding with a CRF on the top. [34] is based on the CharWNN deep neural network, which uses word and character embeddings to perform sequential classification. [14] addresses an orthogonal problem called word sense disambiguation problem. Its contribution consists of models from a traditional LSTM-based model, a variant that incorporates an attention mechanism and an encoder-decoder architecture.

Recently, the contextualized language models, such as BERT, GPT [26], ELMo, and Flair Embeddings [27], are becoming a new paradigm of NER. BERT, which uses the Transformer architecture [42], among with its derived models, such as RoBERTa [43] and Albert [44], is one of the most adopted models. Some works have achieved promising performance via leveraging the combination of traditional embeddings (e.g., Google Word2Vec, Stanford GloVe, etc.) and BERT or by fine-tuning BERT with one additional output layer for the NER task as [21–24]. However, BERT has a prohibitively large number of parameters and require substantial computational resources [45]. Besides, even though the Transformer encoder is more effective than LSTM, it fails the NER task if they are not pre-trained, and when training data is limited [1]. The pre-trained contextualized embeddings are data-hungry and require a massive amount of training data to be fine-tuning for a domain-specific NER task. Due to this limitation, we will jointly investigate our approach with the pre-trained language models in future works.

It is worth mentioning that none of the previously mentioned related works address our problem. The problem addressed by [19] solved by the HYENA model is adjacent to the Hierarchical Entity-Label Disambiguation problem. Different from HELD in practice, HYENA is a representative supervised method that uses a top-down hierarchical classifier. Its features include the words in the named entity mention, in sentence and paragraph, and POS tags. It performs basic co-reference resolution and marks entity mentions connected to the fine-grained labels present in the YAGO knowledge base.

In our previous work [46], we also tackle a quite similar problem, i.e., the label or class disambiguation in Police reports documents. We proposed a Char-BLSTM-CRF model that concatenates char and word embeddings to combine word- and character-level representations to feed them into BLSTM to model context information of each word. On the top of BLSTM, there is a sequential CRF to decode labels for the whole sentence jointly. In this paper, we propose HELD, an ensemble model that combines a variation of Char-BLSTM-CRF to enhance an off-the-shelf NER tool to solve the Hierarchical Entity-Label Disambiguation problem in Police reports. Compared to existing works in the literature, we highlight that our approach does not rely on knowledge bases to disambiguate the entities present in the text. This brings advantages to HELD in several domains where it is unfeasible to create knowledge bases to support NER tasks, such as in the Police domain.

3. Problem Definition

In this section, we introduce the formulation of our Hierarchical Entity-Label Disambiguation problem in Police reports, and some basic concepts and notations used throughout the paper. A Police narrative report is a document that describes all of the raised facts, circumstances, and timeline events surrounding an incident. The process of writing and the protocols applied to the Police reports might vary from one agency to another, however, the general information and function are relatively the same.

A challenge to automate the information extraction from such a document is the correct interpretation of the domain-specific named entities according to their roles in the narrative. In literature, NER is the task to identify entity mentions from text belonging to predefined categories, such as person, location, and organization, which define the well-known generic category of entity labels. In order to intelligently understand different texts and extract a wide range of information, it is useful to precisely determine the labels of entity mentions for domain-specific NER systems. Thus, for our Police reports domain, the entity labels may reflect the victim, robber, witness, etc.

Specifically, the entity recognizer for our Police domain must categorize the entities into fine-grained labels. This kind of NER task is called as fine-grained NER, which allows one entity mention to have multiple labels. Together, these labels constitute a path in a given label hierarchy, depending on the local context (i.e., the sentence context). The formal definitions of named entities and fine-grained entity labels in Police reports are as follows:

Definition 1. *(Named Entities): The named entities are the real-world objects written in the narrative reports, represented by words or phrases that serve as a name for something or someone.*

Definition 2. *(Fine-Grained Entity Labels): The fine-grained entity labels are the roles or specialization of the named entities present in the narrative reports (e.g., victim, witness, incident location, etc.), organized in a hierarchy (e.g., victim is a sub-label of person and incident location is a sub-label of location).*

In particular, when dealing with Police reports, it is crucial to avoid misunderstandings between the fine-grained entities. Most of them are proper nouns or terms often ambiguous. For example, the actual victim can not be recognized as the robber, and vice-versa. The same is also true for entities concerning the locations of the events present in the reports: the location of the incident should not be classified as the victim's residence location. The lack of a knowledge base to give a disambiguated meaning to entity mentions in the narratives increases the complexity of the problem. Consequently, it requires to perform the disambiguation of these named entities based on the context.

Recognize and disambiguate named entities is still a limitation to a NER-based approach, especially for the categorization of fine-grained entity labels. Thus, the problem that we want to solve here is to classify the named entities present in the Police reports according to their entity labels, arranged in a hierarchical structure. For this task, the classifier needs to solve the label disambiguation problem when assigning the proper role of the existing entities. Therefore, we define our Hierarchical Entity-Label Disambiguation problem:

Problem Statement. *(Hierarchical Entity-Label Disambiguation using Context): We consider a variable-length sequence of input symbols* $s = \{x_1, x_2, \cdots, x_N\}$*, and we aim to predict a sequence of output symbols* $y = \{y_1, y_2, \cdots, y_{N'}\}$*. Input symbols are word tokens drawn from a given vocabulary* V*. Output symbols are labels drawn from a given set of fine-grained entity labels* $L = \{l_1, l_2, \cdots, l_C\}$ *organized in a hierarchy of two or more levels, where* C *is the total number of labels. The top-level of this hierarchy consists of the super-labels* E*. Consider that* L *might present labels that are ambiguous for a given super-label* $e_z \in E$*. The problem tackled in this paper is to deal with the fine-grained entity labels disambiguation by regarding the sentence context.*

In practice, to solve the Hierarchical Entity-Label Disambiguation problem, we consider for the experimental procedures that our fine-grained entity labels are arranged in a two-level label hierarchy. The first level corresponds to generic super-labels, whereas, the second level contains fine-grained entity labels for the Police domain. For instance, named entities such as the victim's name and the robber's name should be assigned labels of type VICTIM and ROBBER, respectively. Also, both VICTIM and ROBBER entity labels correspond to only one super-label, in this case, PERSON. We will present and discuss further below in the next sections on this hierarchical structure present in the fine-grained labels of our domain.

4. HELD: A Method for Hierarchical Entity-Label Disambiguation in Police Reports

In this section, we present HELD from bottom to top, characterizing the layers of the architecture. HELD is an ensemble approach designed to solve the Hierarchical Entity-Label Disambiguation problem in Police reports. Although the problem definition and our method were developed within the context of narrative reports, HELD is a domain-independent approach that can be applied to texts from different domains. As defined in the previous section, these texts must have ambiguous fine-grained entity labels organized in a hierarchy.

Refer to Figure 1 for an example illustration that provides an overview of HELD. Overall, given an input sentence $s = \{x_1, x_2, \cdots, x_N\}$ that contains the individual words of a Police report drawn from a vocabulary V, the sequence of words s is used as input for two sequence labeling components for NER: a deep learning model, and a NER tool, that also uses DL-based methods. The Disambiguation Mask component combines the predictions of the two main components to tags each symbol $x_i \in V$

with an output symbol y_i. Specifically, the first two components in our approach are a NER model from the spaCy library and a BLSTM-CRF model. SpaCy has an off-the-shelf NER tool frequently used by academia and industry projects, while the BLSTM-CRF is the most common architecture for NER using deep learning [1]. We provide more details about the HELD components in the next sections.

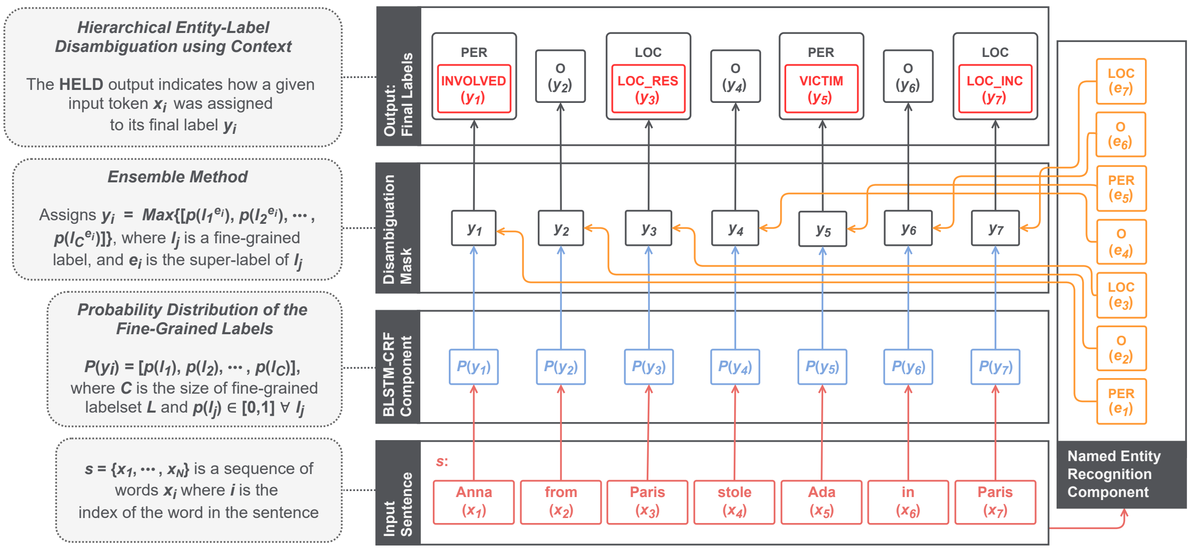

Fig. 1. The overall architecture of HELD. An illustrative sentence is provided as input to the two main components of HELD: the NER component and the BLSTM-CRF component. For this example, the named entities we are interested in are: *Anna*, *Ada*, and *Paris*. All entities preserve the same orthographic features but observe that *Paris* is mentioned twice in different contexts. At first, it indicates *Anna*'s residence, and, finally, the place where the incident occurred. The NER component recognizes the entities with the two super-labels PER (PERSON) and LOC (LOCATION). From the BLSTM-CRF component, we obtain the *Probability Distribution of the Fine-Grained Labels*. The output of the two components is used to feed into the Disambiguation Mask, which is the ensemble method that disambiguates the meaning. The final model output contains each sentence token assigned with its label: (i) *Anna* is the person involved in the incident; (ii) the first occurrence of *Paris* indicates that it is a residence location; (iii) *Ada* is the victim; and (iv) the second occurrence of *Paris* is the incident location.

As illustrated in Figure 1, HELD must intrinsically extract information from the context. One may think HELD could use OSM[2] as a knowledge base or GeoNames[3] gazetteer to disambiguates the fine-grained entity labels related to the LOCATION super-label. However, OSM or GeoNames would not be applicable for other inherent entity labels in the Police domain (e.g., people present in the incident, such as the victim, robber, witnesses, among others). Another idea that comes up to help in the annotation process is a domain expert to supervise the entire learning process. This is similar to our previous work [47] called HNERD, which is an interactive framework designed to assist the user in the human annotation process and to perform NER tasks. In experimental procedures, we use HNERD to manually annotate a Police corpus. All in all, HELD is already effective in automatically learning useful representations and underlying factors from raw data.

[2]https://wiki.openstreetmap.org/wiki/Main_Page/
[3]https://www.geonames.org/

4.1. Named Entity Recognition Component

In our proposed ensemble model, the NER component is the one in charge of recognizing the named entities in Police reports into the top-level labels of the hierarchical structure present in our fine-grained entity labels. This component does not disambiguate the fine-grained entities. Instead, it identifies and classifies the entities belonging to the super-labels by performing a traditional sequence labeling NER task. Particularly in our context, we are interested in the recognition of two main types: LOCATION and PERSON. Afterward, the output of this component, jointly with the BLSTM-CRF output, will feed into the Disambiguation Mask.

We use spaCy NER[4] as the NER component. This off-the-shelf NER tool is frequently awarded in many industry projects, given your sophisticated neural network-based model that achieves state-of-the-art performance. In the latest release, spaCy v2.0's deep learning models are reported to be 10 times smaller, 20% more accurate, and cheaper to run than the previous generation [48]. SpaCy supports online learning, so the entity recognizer can be updated with new examples using an existing available pre-trained statistical model.

The spaCy NLP models, especially NER, follow a simple four-step formula: embed, encode, attend, and predict. First, the model receives the text and transforms the words into unique numerical values. In the embedding stage, features such as the prefix, suffix, shape, and lowercase are used to extract the similarities between the words. To encode the context-independent embeddings, the values pass through a CNN network, producing a context-sensitive sentence matrix. Before the prediction, the matrix has to pass through the CNN Attention layer to be converted into a single vector. Then, a standard Multi-layer Perceptron (MLP) with a Softmax layer is used as a tag decoder layer for class prediction. After the training process, the spaCy model is ready for several NLP tasks.

4.2. BLSTM-CRF Component

The BLSTM-CRF component in HELD is the one in charge of learning how to disambiguate the fine-grained entities present in the Police reports. In other words, while the NER component only recognizes the entities to the super-labels in the two-level hierarchy (i.e., PERSON and LOCATION), the BLSTM-CRF must recognize and disambiguate them accordingly to their actual fine-grained labels (i.e., the PERSON and LOCATION sub-labels). As previously mentioned, the input for the Disambiguation Mask is the combined output of these two main components.

In the setting of sequence labeling for NER, our BLSTM-CRF component first feeds word-level representations into a BLSTM layer to encode context information of each word. On top of the BLSTM, a sequential CRF layer takes context into account to decode labels for the whole sentence jointly. In the following sections, we describe in detail the architecture of this deep learning model.

4.2.1. Word-Level Representation

Using as the input, traditional word embeddings can capture semantic and syntactic properties of words, which do not explicitly present in the input text. We consider two types of non-contextualized word-level representations in this research: (a) domain-specific or (b) pre-trained word embeddings. We call domain-specific word embeddings the word vectors that have their weights randomly initialized in the word embedding layer by using a vocabulary to map the integer indices from the training set

[4]https://v2.spacy.io/api/entityrecognizer

data to dense vectors. Iteratively, during training, these word vectors are gradually adjusted via back-propagation, structuring the space into something the model can exploit. Once fully trained, the domain-specific embedding space will show a structure specialized for the problem.

For the pre-trained word embeddings, we consider those available in online repositories, typically trained over large collections of text reflecting a wider domain, such as FastText, GloVe, Wang2Vec, and Word2Vec. By using a domain-specific dataset, we can face an out-of-vocabulary (OOV) problem, which happens when some words from our data do not exist in the pre-trained word embeddings vocabulary. To handle this, we use the Levenshtein Edit Distance metric, since we do not have a vector representation for these words (i.e., domain-specific or misspelled words). Also, we want to ensure that the position of similar words in the high-dimensional space can remain the same or improve during training, in order to achieve a consistent domain-specific representation.

Specifically, the word embedding layer of the BLSTM-CRF component converts each word $x_i \in s$ into a real-valued d-dimensional vector w_{x_i} with a fixed size d via embedding matrix $W \in \mathbb{R}^{dx|V|}$. Concerning the pre-trained word embeddings only, we first replace each OOV x_i in the input text with another word with the minimum edit distance value to x_i from the pre-trained word embedding vocabulary. Then, we manually pass via embedding matrix the final pre-trained vectors. In our context, we state that the Police reports do not exceed a constant number of words M. Thus, for a given number of Police reports N received as input, we have a $(M * d) \times N$ dimension matrix as the output of the word embedding layer. The Police reports with fewer than M words are padded at the end. The output matrix will then feed into the BLSTM layer.

4.2.2. BLSTM for Context Encoder with a CRF Decoder

The BLSTM-CRF component must capture the context dependencies, taking into account the local context of each embedded word to predict labels for the tokens in the input sequence. For many sequence labeling tasks, it is beneficial to have access to both past (left) and future (right) contexts. However, LSTM's hidden state takes information only from the past, knowing nothing about the future. An elegant solution is bidirectional LSTM (BLSTM), which consists of using two regular LSTM layers. Each LSTM layer processes the input sequence in one direction (chronologically and anti-chronologically) and then merging their representations. By treating a sequence in both ways, a BLSTM can catch patterns that may be missed by the chronological-order version alone. To capture such contextual information, we use a stacked BLSTM layer.

For prediction, it is beneficial to consider the correlations between labels in neighborhoods and jointly decode the best chain labels for a given input sequence [39]. For example, consider a category of generic entity labels, such as PERSON, LOCATION, and ORGANIZATION. It is known that a token that identifies an ORGANIZATION cannot follow a PERSON token. Having this in mind, we use a Conditional Random Fields (CRF) statistical model as the tag decoder. The output vectors of the BLSTM layer are fed into a CRF layer to jointly decode the best sequence of labels, focus on sentence-level instead of individual positions. CRFs takes context into account, and it is powerful to capture label transition dependencies when adopting word embeddings, producing a higher accuracy performance in general [1, 12].

Eq. (1) formalizes the combination of a BLSTM neural network with a CRF model. Let $v = \{v_1, \cdots, v_T\}$ be some observed input data sequence, in our case, the output vector of BLSTM. Let S be a set of FSM (Finite State Machine) states, each of which is associated with a label, $y_i \in L$. Let $s = \{s_1, \cdots, s_T\}$ be some sequence of states (the values on T output nodes). CRF defines the conditional probability of a state sequence given an input sequence to be $P(s|v)$, where Z_v is a normalization factor, $f_k(s_{t-1}, s_t, v, t)$ is an arbitrary feature function over its arguments, and λ_k is a learned weight for each feature function. Feature functions can ask powerfully arbitrary questions about the input sequence, including queries about previous words, next words, and conjunctions of all [49]. For further details, we refer the reader to [49, 50].

$$P(s|v) = \frac{exp(\sum_{t=1}^{T} \sum_k \lambda_k f_k(s_{t-1}, s_t, v, t))}{Z_v} \tag{1}$$

4.3. Disambiguation Mask

The Disambiguation Mask component is in charge of assigning the final fine-grained labels to the input tokens. For this task, the Disambiguation Mask combines the outputs of the NER and BLSTM-CRF components by performing a pairwise decision. Both main components in HELD identify approximately the same-named entities, dealing with them in different ways. Within this context, review Figure 1 which illustrates HELD. Given an input token x_i, for each pair of outputs, the Disambiguation Mask first looks at the NER component output. The NER component recognized x_i with a super-label $e_i \in E$, where E is the set of super-labels. Then, together with the output of the BLSTM-CRF component, in this case, the marginal probability distribution of the fine-grained labels $P(y_i) = [\, p(l_1), p(l_2), \cdots, p(l_C)\,]$, where $p(l_j) \in [0, 1] \; \forall \; l_j \in L$, the Disambiguation Mask assigns for the input token x_i, the label l_j that has a higher probability, given that l_j is the sub-label of e_i. Eq. (2) formalizes our Disambiguation Mask, i.e., how a given input token x_i is assigned to its final label y_i.

$$y_i = Max\{\, [\, p({l_1}^{e_i}), p({l_2}^{e_i}), \cdots, p({l_C}^{e_i})\,]\,\} \tag{2}$$

5. Experimental Evaluation

In this section, we discuss the experimental evaluation. The main research questions that guide this study are:

RQ1 How effective is the NER component at recognizing the named entities in Police reports into the super-labels of the hierarchical structure present in our fine-grained entity labels?
For this research question, we explore the two possible usage versions of the spaCy NER. One version has the available pre-trained entity recognizer model trained on a large corpus, while the other version concerns the update (fine-tuning) of the pre-trained spaCy NER with our data. We aim at evaluating how these versions perform for the Police domain in the super-labels recognition.

RQ2 What is the best word-level representation for the BLSTM-CRF component that most effectively captures the semantic properties of the Police report's vocabulary?
This research question investigates the best input word representation for the BLSTM-CRF component. We present the performance of the BLSTM-CRF model on different pre-trained word embeddings: FastText, GloVe, Wang2Vec, and Word2Vec. We also generate domain-specific word embeddings using the training set data, mapped through a domain-specific vocabulary with almost 400,000 tokens.

RQ3 Can our ensemble architecture HELD leverage a NER component to solve the Hierarchical Entity-Label Disambiguation problem in Police reports?
With this research question, we want to compare the performance of our proposed model against some baselines. Through the experimental results, it will be possible to have a better clue of the importance of each proposed component in the architecture.

Note that these three research questions, previously defined by domain experts, are complementary to each other and follow the architecture of Figure 1. In the following sections, we discuss the schemes designed to answer them. We first introduce the dataset used in the experiments, report the experiment setups, and finally discuss the results.

5.1. Dataset Construction

To evaluate our proposed method, we use a real dataset with texts from Police reports. In contrast to other domain-specific datasets, to our best knowledge, there is no available annotated dataset for the Police domain.

Data annotation. We use our interactive framework HNERD (Human Named Entity Recognition with Deep Learning) [47] to create a manually annotated dataset. Despite being a complete tool to assist the user in NER classification tasks, HNERD was used only for the annotation process. The annotated corpus contains a total of 3,083 real narrative reports written in Portuguese describing the homicides in the city of Fortaleza (Brazil) from 2014 to 2020.

Data annotation remains time-consuming and expensive. It requires domain experts to perform intensive annotation tasks. A total of nine domain experts were responsible for hand-labeling the data with HNERD. The manually annotated dataset was based on a less fine-grained annotation scheme with five labels:

(i) VICTIM, a named victim;
(ii) PARENTS, the names of the victim's parents;
(iii) INVOLVED, the names of any other person mentioned in the narrative, other than VICTIM or PARENTS;
(iv) LOC_DEATH, the name of victim's death location; and,
(v) LOC_RES, the name of victim's residence location.

The two corresponding super-labels in the two-level hierarchy present in the fine-grained labels are: PERSON, which covers the labels VICTIM, PARENTS, and INVOLVED, and recognizes a named person or family; and LOCATION, the names of politically or geographically defined location (i.e., names or addresses of public or private spaces, cities, provinces, countries, bodies of water, mountains), covering the labels LOC_DEATH and LOC_RES. Also, we use the O (OTHER) label to indicate non-entity tokens.

Labels statistics. The annotated corpus contains a total of 23,837 named entities (per-token basis) with 10,960 mentions to VICTIM, 6,516 to LOC_DEATH, and 5,377 to INVOLVED. These first three fine-grained labels representing the group of the most frequent entities, the majority labels. For the group of the rarest or minority labels, we have 512 mentions to LOC_RES and 472 to PARENTS. As we can observe, the distribution of our dataset faces a data imbalance problem that occurs when some labeled entities appear much more than others.

We follow the iterative stratification technique proposed by [51] to provide the official training (60%), development (20%), and test (20%) sets, using the same seed for the split of our multi-label corpus, which has several target labels per text. Since we are dealing with a sequence labeling task, the iterative stratification manages to output a proportional sample of labels in the three sets to prevent measurement errors for the data imbalance issue. After the split, the training set remained with a total of 1,833 texts, while the development and test sets both with 625 instances each. The per-token named entities distribution for each subset is reported in Table 1.

Table 1
Fine-grained label distributions per token.

Super-label	Label	Sets			
		Corpus	**Train**	**Test**	**Dev**
PERSON	VICTIM	10,960	0.465	0.453	0.451
	PARENTS	472	0.020	0.019	0.021
	INVOLVED	5,377	0.224	0.228	0.227
LOCATION	LOC_DEATH	6,516	0.269	0.276	0.283
	LOC_RES	512	0.021	0.025	0.019

5.2. *Experiment Setup: Training Process, Evaluation Metrics, and Baseline Models*

NER component training process. We use the following steps to train the NER component in HELD. First, all spaCy models support online learning, so we load an existing pre-trained model available for the Portuguese language[5]. Then, we disable all the other pipeline components to retrain only the spaCy entity recognizer. After defining the labels (in this case, PERSON and LOCATION) and the epochs for training, we loop over our training set annotated with the super-labels[6]. During the loop, we update the model to steps through the words of the input. At each word, the spaCy NER makes a prediction and then consults the annotations to see whether it was right; if it was wrong, it adjusts its weights so that the correct action will score higher next time [48]. For each epoch, we test the retrained model using the development set to make sure that the entities in the training data are correctly recognized.

BLSTM-CRF component training process. We use our training set annotated with the fine-grained labels to train the BLSTM-CRF component in HELD. Similar to the NER component training process, for each epoch, we assess the accuracy of the BLSTM-CRF model using the development set. We also use an early stop strategy to stop the training phase if the accuracy of the model on the development set is not improving. For comparative purposes, we train the model with three distinct loss functions: Class-Balanced (CB) loss [52], Dice loss (DL) [22], and CRF loss [50]. The CB and Dice losses are widely used heuristics that deal with the label imbalance issue, while the CRF loss is often required in models for sequential labeling tasks that have a CRF layer as a tag decoder. Specifically, the CRF loss decides a proper loss (e.g., Categorical Cross-Entropy, Sparse Categorical Cross-Entropy, etc.) for the CRF layer learning mode. By addressing cost-sensitive learning solutions to deal with the data imbalance problem is expected that BLSTM-CRF must be able to achieve a significant increase in performance.

[5]https://v2.spacy.io/models/pt

[6]For the NER component training process, we replace the annotated fine-grained labels in our dataset with only the super-labels. This experimental procedure is explained in Section 5.3.

The CB loss can be applied as a generic loss to a wide range of deep learning models and loss functions. This version considers exactly one label per sample [52], so we had to adapt it to deal with multi-label data. Our adaptation assigns each sentence token to one of the mutually exclusive classes, which corresponds to the Categorical Cross-Entropy (CCE) loss [53]. We replace the loss function $\mathcal{L}$ in the CB loss formula with the CCE loss, leading to the Categorical Cross-Entropy Class Balancing (CB-CCE) loss to substitute the generic CB loss.

Evaluation metrics. The evaluation metrics used in the experiments are Precision (P), Recall (R) and F_1-score (F_1), defined by Eq. (3-5). They are widely used for an exact-match evaluation of NER systems, where a correctly recognized instance requires a model to correctly identify its boundary and label, simultaneously [1]. The numbers of false positives (FP, recognized entities that do not appear in the ground truth), false negatives (FN, not recognized entities that appear in the ground truth), and true positives (TP, recognized entities that also appear in the ground truth) are used to compute Precision, Recall, and F_1-score, given as follows:

$$\mathrm{P} = \frac{TP}{TP + FP} \qquad \mathrm{R} = \frac{TP}{TP + FN} \qquad \mathrm{F}_1 = 2 \cdot \frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}} \tag{3, 4, 5}$$

This work focuses on the token-level method evaluation for all the tested models in this experimental procedure. The metrics P, R, and F_1 are used to compute each label individually, considering that each input token present in the corpus has a label assigned to it. In addition, the macro-averaged of the empirical metrics values considers the performance across multiple entity labels. Macro-averaging treats all the recognized entities as equal to provide a realistic calculation since we do not have a large number of imbalanced entity labels. Consider a metric M chosen to measure the performance of each labeled entity, the macro-averaged M (M_{macro}) independently calculates the M on different entities, then takes the average of the M's. For a labelset L with the size $|L|$, the M_{macro} is given as follows:

$$M_{macro} = \frac{1}{|L|} \sum_{l=1}^{|L|} M_l, \tag{6}$$

Baselines. We compare HELD against two baselines. All the models are trained and evaluated using the same training, development, and test stratified data as used by the HELD model. Notice that these baselines are the isolated components of HELD. Nevertheless, these components, individually, represent state-of-the-art architectures for solving NER tasks. Thus, in the context of fine-grained label disambiguation in Police reports, we want to evaluate how the ensemble architecture of HELD outperforms these two common approaches used in NER.

- *spaCy NER (spaCy)*: We use the pre-trained spaCy NER model [48] available for Portuguese fine-tuned on our training set annotated with the fine-grained labels. The idea here is to compare how a NER solver performs on our Hierarchical Entity-Label Disambiguation problem.
- *BLSTM-CRF*: In [46], we developed a deep learning model called Char-BLSTM-CRF, which achieved high performance on tackle the label or class disambiguation problem in Police reports documents. In the present work, we designed the BLSTM-CRF component of HELD based on the Char-BLSTM-CRF model. Char-BLSTM-CRF incorporates character-based word representations

extracted by an LSTM neural network, so we removed the character-level layer since we deal with OOV words differently. Therefore, we only use word-level embeddings as input for the BLSTM-CRF recognition baseline model. This model was implemented using the Keras library [54] with TensorFlow in the backend.

Computational resources. All models listed in our experiments were implemented and run on Colab[7], a product from Google Research. Colab is a hosted Jupyter Notebook service that requires no setup to use. In particular, we use TPUs to run the BLSTM-CRF models, one of Colab's free computing resources. For spaCy models, we ran using Colab's CPU. SpaCy v2.0 does not support TPUs, and the GPU runtimes, at least in our experiments, did not provide faster training.

Model notation. For the experimental procedure and results, we use the two following notations: (i) $MODEL^{<loss>}_{<embedding>}$, where *MODEL* refers to the deep learning approach used (i.e., BLSTM-CRF or HELD), and the indexes *embedding* and *loss*, corresponding to the word-level embedding representation and the loss function used for training, respectively; (ii) $spaCy_{<method>}$, where the index *method* identifies for the spaCy NER tool whether we are using its *pre-trained* or *fine-tuned* (with our training data) version.

5.3. RQ1: Evaluating the Performance of the NER Component

Our experimental results for the RQ1 over the test set are summarized in Table 2.

Usage setups. We assessed the two possible usage setups of the spaCy NER: $spaCy_{pre\text{-}trained}$ and $spaCy_{fine\text{-}tuned}$. The first one refers to the pre-trained available NER model that recognizes the following entities: PER (PERSON), LOC (LOCATION), ORG (ORGANIZATION), and MISC (MISCELLANEOUS), trained with the Universal Dependencies (UD) Portuguese treebank and WikiNER corpora [48]. In turn, $spaCy_{fine\text{-}tuned}$ refers to the fine-tuning of $spaCy_{pre\text{-}trained}$ with our training data annotated with the labels PERSON and LOCATION. Both usage setups took about 2 to 3 hours of training.

Data annotation format. To retrain the $spaCy_{pre\text{-}trained}$, we replace the fine-grained labels annotated in our stratified subsets (train, test, and development) to be tagged only with the super-labels. Therefore, the mentions of VICTIM, PARENTS, and INVOLVED were replaced by PERSON, and the mentions of LOC_RES and LOC_DEATH became LOCATION.

Table 2

The performances of *spaCy NER* pre-trained and fine-tuned models for the super-labels.

Super-label	$spaCy_{pre\text{-}trained}$			$spaCy_{fine\text{-}tuned}$		
	P	**R**	$\mathbf{F_1}$	**P**	**R**	$\mathbf{F_1}$
PERSON	0.1819	0.1289	0.1509	0.9467	0.9382	**0.9424**
LOCATION	0.1399	0.1634	0.1507	0.6275	0.6719	**0.6489**
Macro-averaged	*0.1609*	*0.1462*	*0.1508*	*0.7871*	*0.8051*	***0.7957***

Discussion. $SpaCy_{fine\text{-}tuned}$ shows better performances in all three metrics for the two super-labels, reaching a macro F_1-score equal to 0.7957. Interestingly, $spaCy_{pre\text{-}trained}$ achieves a limited performance in this experiment, with similar F_1-score values for PERSON and LOCATION (slightly difference of $+0.0002$), and a macro F_1-score of 0.1508. We believe that the spaCy model only trained on UD Portuguese treebank and Wikipedia corpora perform inconsistently due to the specificities of our Police domain. Also,

[7]https://colab.research.google.com/

by fine-tuning the pre-trained NER model only for the super-labels PERSON and LOCATION, skipping ORG and MISC, the model seems more effective in learning to identify and classify named persons and locations. The F_1 values for *spaCy*$_{\textit{fine-tuned}}$ results are consistent with the total number of samples for each super-label in our dataset since we have more mentions for PERSON (higher F_1 of 0.9424, recognizing almost all mentions for this entity) than LOCATION (F_1 of 0.6489).

We observed by addressing RQ1 that the *spaCy*$_{\textit{fine-tuned}}$ is more effective in recognizing the named entities present in Police reports into the super-labels of the hierarchical structure present in our fine-grained entity labels. Our experiments showed that this fine-tuned version significantly outperformed the *spaCy*$_{\textit{pre-trained}}$ model for the Police domain. To conclude, *spaCy*$_{\textit{fine-tuned}}$ is the best setup for the NER component in HELD.

5.4. RQ2: Evaluating the Best Word-Level Representation for the BLSTM-CRF Component

With the most effective spaCy NER model setup identified for the NER component, following the architecture of HELD, we focus here on the RQ2 to investigate the best word-level representation for the BLSTM-CRF component. We make the following observations before discussing the results of Table 3:

Input setups. Since our dataset contains documents written in Portuguese, to train the domain-specific word embeddings we use the same corpus vocabulary used in our previous work [55], which has 393,550 tokens collected from Police reports from Fortaleza (Brazil) from 2010 to 2018. Concerning the pre-trained word-level embeddings, we use the vectors generated by FastText, GloVe, Wang2Vec, and Word2Vec models that were trained on seventeen linguistic corpora of Brazilian and European Portuguese, from different sources and genres, corresponding to more than one trillion tokens [56]. These pre-trained versions are available on the NILC repository[8], and we selected the ones that were trained with the Skip-Gram architecture since it gives a better representation of rare words and is more indicated to identifies patterns [57]. Both pre-trained and domain-specific word embeddings have an embedding size of 50 dimensions.

Specifically, we evaluate three different input setups for the word embedding layer: *domain-specific*, *pre-trained*, and *concat*. The *domain-specific* version learns word embeddings for our Police domain jointly with the BLSTM-CRF model training, while the *pre-trained* version uses the original pre-trained vectors (from FastText, GloVe, Wang2Vec, and Word2Vec) fixed during training. Lastly, the *concat* (i.e., a concatenation layer) concatenates both the *domain-specific* and the *pre-trained* vectors.

Missing words. For what concerns the statistics of our dataset, the number of tokens in a sentence varies from 6 to 537. We observe that 50% of the sentences have 32 tokens or less, while 75% of the dataset have 49 tokens or less. By looking at the effectiveness of *domain-specific* version, in 50% of sentences *domain-specific* embedded 31 tokens per sentence or less. While for 75% of the dataset, *domain-specific* word embeddings represented 49 tokens per sentence or less. When we used the *pre-trained* version as a word embedding layer, 30 tokens per sentence or less were embedded for 50% of sentences. While for 75% of the dataset, *pre-trained* word embeddings represented 47 tokens per sentence or less. The differences between the percentiles are slight, which indicates that the level of missing tokens for sentences embedded by *domain-specific* and *pre-trained* are minimal.

Discussion. In more detail, each row in Table 3 represents a word-level embedding representation and a loss function. We evaluate the performance of the BLSTM-CRF model by training it with nine word-level representations (i.e., one *domain-specific*, four *pre-trained*, and four *concat* versions) and three loss

[8] http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc

Table 3

The general performances of the *BLSTM-CRF* variations.

<word-level embedding>	*BLSTM-CRF* $^{CB\text{-}CCE}$			*BLSTM-CRF* ^{CRF}			*BLSTM-CRF* ^{DL}		
	$\mathbf{P}_{macro}$	$\mathbf{R}_{macro}$	$\mathbf{F}_{1\,macro}$	$\mathbf{P}_{macro}$	$\mathbf{R}_{macro}$	$\mathbf{F}_{1\,macro}$	$\mathbf{P}_{macro}$	$\mathbf{R}_{macro}$	$\mathbf{F}_{1\,macro}$
Domain-specific	0.411	0.580	0.455	0.471	0.473	0.469	0.462	0.440	0.447
FastText	0.530	0.778	0.611	0.709	0.556	0.603	0.598	0.522	0.554
GloVe	0.557	0.763	**0.629**	0.712	0.565	0.598	0.475	0.436	0.453
Wang2Vec	0.561	0.739	0.619	0.744	0.580	**0.638**	0.491	0.425	0.452
Word2Vec	0.486	0.767	0.574	0.733	0.574	0.614	0.473	0.460	0.465
ConcatFastText	0.431	0.686	0.491	0.565	0.602	0.581	0.591	0.574	**0.580**
ConcatGloVe	0.469	0.638	0.521	0.560	0.595	0.576	0.486	0.501	0.492
ConcatWang2Vec	0.458	0.625	0.509	0.574	0.551	0.556	0.500	0.470	0.483
ConcatWord2Vec	0.457	0.654	0.508	0.575	0.554	0.562	0.524	0.510	0.508

functions (i.e., CRF, CB-CCE, and DL). All the BLSTM-CRF model variations were trained in 4 to 5 hours.

The *BLSTM-CRF* $^{CRF}_{Wang2Vec}$ model had the best performance with a macro F_1-score equal to 0.638. In general, the performance of the models using the *pre-trained* setup for the word embedding layer outperform the two other versions: *domain-specific* and *concat*. In this case, the *domain-specific* word embeddings had the worse average performance (-0.191 macro F_1 vs. *BLSTM-CRF* $^{CRF}_{Wang2Vec}$). The results suggest that the corpus is not large enough for representing the relationship between the words used in the Police reports. Besides, the fact that the *concat* representation under-performs the *pre-trained* one suggests that the *domain-specific* layer in the *concat* setup added noise to the model. *Domain-specific* and *pre-trained* have different representations and, therefore, different vector spaces. When we concatenate different vectors, the scales of the word embeddings may not be the same and neither the dimensional training space. We similarly experiment with the BLSTM-CRF models by *fine-tuning* the pre-trained word embeddings but performed poorly.

Two main observations are obtained from this experiment. First, the performance of the BLSTM-CRF model does not necessarily have just one specific best word embedding representation for all the loss functions. In fact, even though that *BLSTM-CRF* $^{CRF}_{Wang2Vec}$ shows the best overall performance, this model achieves minor improvements ($+0.009$ macro F_1) compared to *BLSTM-CRF* $^{CB\text{-}CCE}_{GloVe}$. As a result, Wang2Vec pre-trained vectors yielded good performance across our evaluations, particularly for the CRF and CB-CCE losses. Second, the use of loss functions that take into account the imbalance characteristics of the dataset had no relevant impact on the performance of the BLSTM-CRF model.

A closer examination of the *BLSTM-CRF* $^{CRF}_{Wang2Vec}$ performance is reported in Table 4 for each label individually. The label that the model performs best is VICTIM (F_1 of 0.875), which is the label with the largest number of samples in the dataset. The most difficult label for recognition is LOC_RES (F_1 of 0.329). For the PARENTS label, even with the smallest proportion of samples, the model achieves an F_1 value of 0.599, a higher value than that obtained for LOC_RES. Most likely due to common expressions found in the text that help to recognize PARENTS, such as "*son of...*", "*daughter of...*", or even "*having as a father [...] and mother [...]*".

In conclusion, the pre-trained word-level representation generated by the Wang2Vec Skip-Gram model most effectively captures the semantic properties of the Police report's vocabulary. We also observe that the use of domain-specific embeddings from our training data has not improved consistently the performance of the models. It may suggest the need of a larger corpus for the domain. By addressing

Table 4
The performance of $BLSTM\text{-}CRF^{CRF}_{Wang2Vec}$ for each label independently.

Super-label	Label	$BLSTM\text{-}CRF^{CRF}_{Wang2Vec}$		
		P	**R**	**F1**
PERSON	VICTIM	0.868	0.883	0.875
	PARENTS	0.658	0.549	0.599
	INVOLVED	0.769	0.624	0.689
LOCATION	LOC_DEATH	0.798	0.618	0.696
	LOC_RES	0.628	0.223	0.329

the RQ2, the best setup for the BLSTM-CRF component in HELD is the $BLSTM\text{-}CRF^{CRF}_{Wang2Vec}$ model. This model performance demonstrates the effectiveness in recognizing the fine-grained entity labels on Police narrative reports.

5.5. RQ3: Assessing the Improvement of HELD

We continue our experiments focusing on the last research question. To address RQ3, we finally investigate the performance of our proposed ensemble architecture: HELD. Our experimental results are reported in Table 5.

Best setups. To compose the architecture of HELD, we use the best setups of each component. We use $BLSTM\text{-}CRF^{CRF}_{Wang2Vec}$ for the disambiguation of the fine-grained labels, and the $spaCy_{fine\text{-}tuned}$ for the super-labels recognition. We compared the performance of our ensemble architecture against each component working individually to solve the Hierarchical Entity-Label Disambiguation problem. In particular, we want to evaluate how HELD performs in confront to $spaCy_{fine\text{-}tuned}$ and $BLSTM\text{-}CRF^{CRF}_{Wang2Vec}$ for the recognition of fine-grained entities in Police reports. The $spaCy_{fine\text{-}tuned}$ model that recognizes fine-grained entity labels took about 2-3 hours to train.

Table 5
The performances of *HELD* and the baseline models.

Super-label	Label	$spaCy_{fine\text{-}tuned}$			$BLSTM\text{-}CRF^{CRF}_{Wang2Vec}$			$HELD^{CRF}_{Wang2Vec}$		
		P	**R**	$\mathbf{F_1}$	**P**	**R**	$\mathbf{F_1}$	**P**	**R**	$\mathbf{F_1}$
PERSON	VICTIM	0.431	0.697	0.535	0.868	0.883	**0.875**	0.849	0.909	0.871
	PARENTS	0.405	0.498	0.449	0.799	0.618	**0.696**	0.613	0.789	0.694
	INVOLVED	0.103	0.273	0.152	0.769	0.624	0.689	0.723	0.697	**0.708**
LOCATION	LOC_DEATH	0.333	0.333	0.333	0.628	0.223	0.329	0.448	0.421	**0.436**
	LOC_RES	0.122	0.158	0.115	0.658	0.549	**0.598**	0.644	0.516	0.573
Macro-averaged		*0.279*	*0.392*	*0.317*	*0.744*	*0.580*	*0.638*	*0.656*	*0.666*	***0.656***

Discussion. HELD shows the best performance overall, as demonstrated by its highest macro F_1-score equal to 0.656. Compared with $spaCy_{fine\text{-}tuned}$, HELD shows better performances for all labels, which demonstrates that this classic NER tool alone is not capable of performing efficient disambiguation on the fine-grained entity labels. Compared with the $BLSTM\text{-}CRF^{CRF}_{Wang2Vec}$ deep learning model, HELD shows the higher F_1 for INVOLVED and LOC_DEATH (F_1 values of 0.708 and 0.436). The $BLSTM\text{-}CRF^{CRF}_{Wang2Vec}$ model is the more challenging baseline, with a macro F_1-score of 0.638, and reaching the best F_1 for VICTIM (0.875) and the rarer entity labels (PARENTS with an F_1 of 0.696 and LOC_RES with 0.598). However, it is also worth noting that the difference between the F_1 values for VICTIM, PARENTS, and LOC_RES is very low for HELD and $\mathit{BLSTM\text{-}CRF}^{CRF}_{Wang2Vec}$ (−0.004, −0.002 and −0.025, respectively).

In summary, in RQ3, we have assessed that by combining the BLSTM-CRF and spaCy NER models in an ensemble architecture, it allows HELD to exploit the best of each model to improve the fine-grained classification in Police reports of the named entities recognized as the super-labels of our two-level hierarchical structure. The important conclusion here is that our approach can leverage the NER component (i.e., $\mathit{spaCy}_{\mathit{fine\text{-}tuned}}$ updated with the super-labels) to disambiguate the entity mentions based on the context.

6. Conclusions and Future Work

This paper has presented HELD, an ensemble model for the Hierarchical Entity-Label Disambiguation problem in Police reports. This problem is very challenging since the fine-grained entities must be recognized and disambiguated in the narrative reports based on the sentence context. To the best of our knowledge, we believe that our work is the first to tackle such a problem. Our ensemble approach is domain-independent, which ensures that it can be applied in various domains as long as it has texts with ambiguous entities arranged in a hierarchical structure. HELD is free from knowledge bases, language-specific resources, and hand-crafted features, and includes two main components for sequence labeling in NER: a NER component represented by the spaCy NER tool and a BLSTM-CRF component. Each component has a specific task: spaCy NER identifies and classifies only super-labels from a two-level label hierarchy, and the BLSTM-CRF model recognizes and disambiguates fine-grained entity labels. HELD combines the predictions of the spaCy NER and BLSTM-CRF models via Disambiguation Mask to assign the final fine-grained labels for the input texts.

We train HELD and the baseline models using a real corpus of Police reports human-annotated in the HNERD framework. To guide our experimental evaluation, we address three research questions. The first research question explored the most effective spaCy NER model setup for the NER component. The $\mathit{spaCy}_{\mathit{fine\text{-}tuned}}$ model, which is the retrained $\mathit{spaCy}_{\mathit{pre\text{-}trained}}$ model with our annotated data only with super-labels, can correctly recognized about 80% more super-labels according to our experimental results. The second research question focused on the discussion on the best word-level representation for the BLSTM-CRF component. We evaluated three different input setups for the word embedding layer of this component (*domain-specific*, *fine-tuned*, and *concat*), and we also test different training loss functions (CRF, CB-CCE and DL). The most effective deep learning model identified in evaluation experiments was the $\mathit{BLSTM\text{-}CRF}^{CRF}_{Wang2Vec}$ model. Finally, for the third research question, we compared the performance of our proposed model with two baselines that are the isolated components of HELD. Compared with the $\mathit{spaCy}_{\mathit{fine\text{-}tuned}}$ and $\mathit{BLSTM\text{-}CRF}^{CRF}_{Wang2Vec}$, a major advantage of HELD is the ability to improve the fine-grained classification in Police reports of the named entities recognized as the super-labels. Evaluation experiments demonstrated that our proposal outperforms the baselines in terms of quality.

Several directions could be pursued to expand this research. First, a research direction would be to explore active learning techniques to reduce the effort of the data annotation process. Second, we can further enhance the performance of HELD by testing other state-of-the-art approaches. Our main interest is in the recent language models, such as BERT, GPT, ELMo, and Flair Embeddings, to improve the

BLSTM-CRF component. We can attach the BLSTM-CRF model on top of the pre-trained language model to predict the labels of each token independently by fine-tuning the model for our domain. Thus, for the NER task, the contextualized language model will have the role of context encoder, besides given contextualized word-level representations. However, to do this, we have to generate a larger annotated Police reports corpus. Also, with more data available, we can increase the number of labels for our fine-grained annotation scheme. Third, another future direction is to explore different loss functions, such as the variations of the Dice Loss [22], or data level strategies for our data-imbalanced dataset. Further, we can explore data-augmentation techniques, such as word-embeddings substitution [58, 59] and masked language model applied by Transformer models, to generate additional synthetic data using the Police reports corpus. Augmentation methods are widely used approaches in computer vision applications, and they are just as powerful for NLP. For our context, they can help generate more labeled data and deal with the data imbalance problem.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This work has been partially supported by FUNCAP, under project 04772551/2020, and by UFC-FASTEF, under project 31/2019.

References

[1] J. Li, A. Sun, J. Han and C. Li, A Survey on Deep Learning for Named Entity Recognition, CoRR abs/1812.09449 (2020), *arXiv preprint arXiv:1812.09449* (2020).

[2] X. Ma and F. Xia, Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1, 2014, pp. 1337–1348.

[3] X. Ling and D.S. Weld, Fine-Grained Entity Recognition, in: *AAAI*, Vol. 12, 2012, pp. 94–100.

[4] D. Ye, Z. Xing, C.Y. Foo, Z.Q. Ang, J. Li and N. Kapre, Software-specific named entity recognition in software engineering social content, in: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1, IEEE, 2016, pp. 90–101.

[5] A. Lal et al., SANE 2.0: System for Fine Grained Named Entity Typing on Textual Data, *Engineering Applications of Artificial Intelligence* **84** (2019), 11–17.

[6] A. Barrena, A. Soroa and E. Agirre, Combining mention context and hyperlinks from wikipedia for named entity disambiguation, in: *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, 2015, pp. 101–105.

[7] S. Cucerzan, Large-scale named entity disambiguation based on Wikipedia data, in: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[8] D.B. Nguyen, M. Theobald and G. Weikum, J-NERD: joint named entity recognition and disambiguation with rich linguistic features, *Transactions of the Association for Computational Linguistics* **4** (2016), 215–229.

[9] S. Hakimov, S.A. Oto and E. Dogdu, Named entity recognition and disambiguation using linked data and graph-based centrality scoring, in: *Proceedings of the 4th international workshop on semantic web information management*, ACM, 2012, p. 4.

[10] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami and C. Dyer, Neural architectures for named entity recognition, *arXiv preprint arXiv:1603.01360* (2016).

[11] J. Hammerton, Named entity recognition with long short-term memory, in: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics, 2003, pp. 172–175.

[12] Z. Huang, W. Xu and K. Yu, Bidirectional LSTM-CRF models for sequence tagging, *arXiv preprint arXiv:1508.01991* (2015).

[13] J.P. Chiu and E. Nichols, Named entity recognition with bidirectional LSTM-CNNs, *arXiv preprint arXiv:1511.08308* (2015).

[14] A. Raganato and R. Navigli, *New Frontiers in Supervised Word Sense Disambiguation: Building Multilingual Resources and Neural Models on a large scale (07a Tesi di Dottorato)*, Ph.D. thesis. Sapienza – University of Rome, 2017.

[15] X. Ren, W. He, M. Qu, L. Huang, H. Ji and J. Han, Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding, in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1369–1378.

[16] A. Abhishek, A. Anand and A. Awekar, Fine-grained entity type classification by jointly learning representations and label embeddings, in: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 2017, pp. 797–807.

[17] L. Del Corro, A. Abujabal, R. Gemulla and G. Weikum, Finet: Context-aware fine-grained named entity typing, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 868–878.

[18] C. Dogan, A. Dutra, A. Gara, A. Gemma, L. Shi, M. Sigamani and E. Walters, Fine-grained named entity recognition using elmo and wikidata, *arXiv preprint arXiv:1904.10503* (2019).

[19] M.A. Yosef, S. Bauer, J. Hoffart, M. Spaniol and G. Weikum, Hyena: Hierarchical type classification for entity names, in: *Proceedings of COLING 2012: Posters*, 2012, pp. 1361–1370.

[20] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT) (1)*, 2019.

[21] X. Li, J. Feng, Y. Meng, Q. Han, F. Wu and J. Li, A unified mrc framework for named entity recognition, *arXiv preprint arXiv:1910.11476* (2019).

[22] X. Li, X. Sun, Y. Meng, J. Liang, F. Wu and J. Li, Dice Loss for Data-imbalanced NLP Tasks, *arXiv preprint arXiv:1911.02855* (2019).

[23] Y. Liu, F. Meng, J. Zhang, J. Xu, Y. Chen and J. Zhou, GCDT: A Global Context Enhanced Deep Transition Architecture for Sequence Labeling, in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2431–2441.

[24] A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer and M. Auli, Cloze-driven Pretraining of Self-attention Networks, 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)*, 2019, pp. 5363–5372.

[25] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, Deep contextualized word representations, *arXiv preprint arXiv:1802.05365* (2018).

[26] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, Improving Language Understanding by Generative Pre-Training, *Technical Report, OpenAI* (2018).

[27] A. Akbik, D. Blythe and R. Vollgraf, Contextual string embeddings for sequence labeling, in: *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1638–1649.

[28] L. Ratinov and D. Roth, Design challenges and misconceptions in named entity recognition, in: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, Association for Computational Linguistics, 2009, pp. 147–155.

[29] A. Ritter, S. Clark, O. Etzioni et al., Named entity recognition in tweets: an experimental study, in: *Proceedings of the conference on empirical methods in natural language processing*, Association for Computational Linguistics, 2011, pp. 1524–1534.

[30] D. Nadeau, P.D. Turney and S. Matwin, Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity, in: *Conference of the Canadian Society for Computational Studies of Intelligence*, Springer, 2006, pp. 266–277.

[31] G. Luo, X. Huang, C.-Y. Lin and Z. Nie, Joint entity recognition and disambiguation, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 879–888.

[32] Y. Wu, M. Jiang, J. Lei and H. Xu, Named entity recognition in Chinese clinical text using deep neural network, *Studies in health technology and informatics* **216** (2015), 624.

[33] Q. Wang, Y. Xia, Y. Zhou, T. Ruan, D. Gao and P. He, Incorporating dictionaries into deep neural networks for the Chinese clinical named entity recognition, *arXiv preprint arXiv:1804.05017* (2018).

[34] C. dos Santos, V. Guimaraes, R. Niterói and R. de Janeiro, Boosting Named Entity Recognition with Neural Character Embeddings, in: *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, 2015, p. 25.

[35] K. Yano, Neural Disease Named Entity Extraction with Character-based BiLSTM+ CRF in Japanese Medical Text, *arXiv preprint arXiv:1806.03648* (2018).

[36] A. Bharadwaj, D. Mortensen, C. Dyer and J. Carbonell, Phonologically aware neural model for named entity recognition in low resource transfer settings, in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1462–1472.

[37] J. Xie, Z. Yang, G. Neubig, N.A. Smith and J. Carbonell, Neural cross-lingual named entity recognition with minimal resources, *arXiv preprint arXiv:1808.09861* (2018).

[38] Z. Yang, R. Salakhutdinov and W.W. Cohen, Transfer learning for sequence tagging with hierarchical recurrent networks, *arXiv preprint arXiv:1703.06345* (2017).

[39] X. Ma and E. Hovy, End-to-end sequence labeling via bi-directional lstm-cnns-crf, *arXiv preprint arXiv:1603.01354* (2016).

[40] K. Mai, T.-H. Pham, M.T. Nguyen, T.D. Nguyen, D. Bollegala, R. Sasano and S. Sekine, An empirical study on fine-grained named entity recognition, in: *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 711–722.

[41] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, Natural language processing (almost) from scratch, *Journal of Machine Learning Research* **12**(Aug) (2011), 2493–2537.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser and I. Polosukhin, Attention is all you need, in: *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[43] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, RoBERTa: A Robustly Optimized BERT Pretraining Approach, *arXiv preprint arXiv:1907.11692* (2019).

[44] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma and R. Soricut, Albert: A lite bert for self-supervised learning of language representations, *arXiv preprint arXiv:1909.11942* (2019).

[45] A. Adhikari, A. Ram, R. Tang and J. Lin, Docbert: Bert for document classification, *arXiv preprint arXiv:1904.08398* (2019).

[46] T.L.C. da Silva, N. da Silva Araujú, J.A.F. de Macêdo, D. Araújo, F.M. Soares, P.A. Rego and A.V.L. Neto, Novel approach for Label Disambiguation via Deep Learning., in: *Machine Learning and Data Mining (MLDM) (2)*, 2019, pp. 431–442.

[47] T.L.C. da Silva, R.P. Magalhães, J.A. de Macêdo, D. Araújo, N. Araújo, V. de Melo, P. Olímpio, P.A. Rego and A.V.L. Neto, Improving Named Entity Recognition using Deep Learning with Human in the Loop, in: *Extending Database Technology (EDBT)*, 2019, pp. 594–597.

[48] M. Honnibal, I. Montani, S. Van Landeghem and A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python, Zenodo, 2020. doi:10.5281/zenodo.1212303.

[49] A. McCallum, Efficiently inducing features of conditional random fields, in: *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc., 2002, pp. 403–410.

[50] J. Lafferty, A. McCallum and F.C. Pereira, *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 282–289–.

[51] K. Sechidis, G. Tsoumakas and I. Vlahavas, On the Stratification of Multi-label Data, in: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2011, pp. 145–158.

[52] Y. Cui, M. Jia, T.-Y. Lin, Y. Song and S. Belongie, Class-Balanced Loss Based on Effective Number of Samples, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9268–9277.

[53] F. Chollet, *Deep learning with Python*, Manning Publications Co., 2017.

[54] F. Chollet et al., Keras resources, Accessed: 2021-04-27. Software available from https://keras.io/. https://github.com/fchollet/keras-resources.

[55] T.L.C. da Silva, N. da Silva Araújo, J.A.F. de Macêdo, D. Araújo, F.M. Soares, P.A.L. Rego and A.V.L. Neto, Novel approach for Label Disambiguation via Deep Learning, in: *Machine Learning and Data Mining in Pattern Recognition, 15th International Conference on Machine Learning and Data Mining, MLDM 2019, New York, NY, USA, July 20-25, 2019, Proceedings, Volume II*, P. Perner, ed., ibai publishing, 2019, pp. 431–442. https://dblp.org/rec/conf/mldm/SilvaAMASRN19.bib.

[56] N. Hartmann, E. Fonseca, C. Shulby, M. Treviso, J. Rodrigues and S. Aluisio, Portuguese word embeddings: Evaluating on word analogies and natural language tasks, *arXiv preprint arXiv:1708.06025* (2017).

[57] T. Mikolov, K. Chen, G. Corrado and J. Dean, Efficient estimation of word representations in vector space, *arXiv preprint arXiv:1301.3781* (2013).

[58] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang and Q. Liu, Tinybert: Distilling bert for natural language understanding, *arXiv preprint arXiv:1909.10351* (2019).

[59] W.Y. Wang and D. Yang, That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets, in: *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 2557–2563.