Improving Event Detection using Contextual Word and Sentence Embeddings

Mariano Maisonnave[a], Fernando Delbianco[b], Fernando Tohmé[b], Ana Maguitman[a], Evangelos Milios[c]

[a]*Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Instituto de Ciencias e Ingeniería de la Computación (UNS-CONICET), Bahía Blanca, Argentina*
[b]*Departamento de Economía, Universidad Nacional del Sur, Instituto de Matemática de Bahía Blanca (UNS-CONICET), Bahía Blanca, Argentina*
[c]*Faculty of Computer Science, Dalhousie University, Halifax, Canada*

Abstract

The task of Event Detection (ED) is a subfield of Information Extraction (IE) that consists in recognizing event mentions in natural language texts. Several applications can take advantage of an ED system, including alert systems, text summarization, question-answering systems, and any system that needs to extract structured information about events from unstructured texts. ED is a complex task, which is hampered by two main challenges: the lack of a dataset large enough to train and test the developed models and the variety of event type definitions that exist in the literature. These problems make generalization hard to achieve, resulting in poor adaptation to different domains and targets. The main contribution of this paper is the design, implementation and evaluation of a recurrent neural network model for ED that combines several features. In particular, the paper makes the following contributions: (1) it uses BERT embeddings to define contextual word and contextual sentence embeddings as attributes, which to the best of our knowledge were never used before for the ED task; (2) the proposed model has the ability to use its first layer to learn good feature representations; (3) a new public dataset with a general definition of event; (4) an extensive empirical evaluation that includes (i) the exploration of different architectures and hyperparameters, (ii) an ablation test to study the impact of each attribute, and (iii) a comparison with a replication of a state-of-the-art model. The results offer several insights into the importance of contextual embeddings and indicate that the proposed approach is effective in the ED task, outperforming the baseline models.

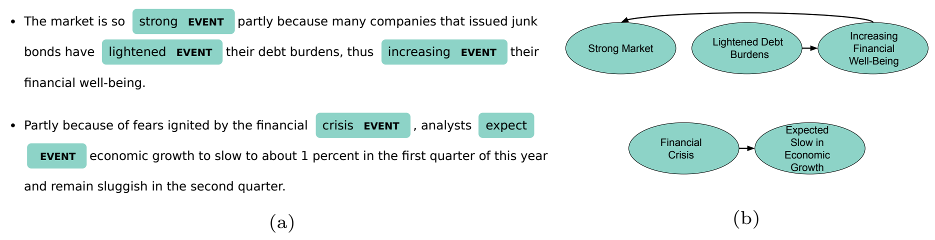
Keywords: Event Detection, Information Extraction, Natural Language Processing, Contextual Embeddings

Email addresses: `mariano.maisonnave@cs.uns.edu.ar` (Mariano Maisonnave),

1. Introduction

The Information Extraction (IE) task consists in extracting structured information from unstructured natural language texts. Event Extraction (EE) is a subtask of IE, in which the goal is to detect and retrieve real-world events from those texts. An EE system performs two different steps to complete the extraction of the events. The first step is to identify the event trigger, which is the word that most clearly expresses the occurrence of an event, and classify it into one of the predefined event types. This step is called Event Detection (ED). The second step is to extract the arguments of the events, which can be a participant (i.e., VICTIM in Injure Event type) or an attribute (i.e., TIME, PLACE). EE and ED systems are crucial for any application or domain that needs structured information and relies on a large corpus of unstructured data. Some examples of this are question-answering systems [41] or text summarization systems [23]. The EE systems are also useful for generating reports of the information available for a domain [1]. These reports can help an expert to make decisions or create policies to address an issue.

Our work aims at extracting real-world events and other relevant variables from news and social media with the ultimate goal of learning causal models. In this paper, we aim to solve the first step, which is the extraction of real-world events. For the extraction of events, in this work, we implemented and tested an RNN model for ED. The two steps involved in the tool we intend to build are depicted in Fig. 1. This work addresses the first step of this tool.


- The market is so strong EVENT partly because many companies that issued junk bonds have lightened EVENT their debt burdens, thus increasing EVENT their financial well-being.
- Partly because of fears ignited by the financial crisis EVENT , analysts expect EVENT economic growth to slow to about 1 percent in the first quarter of this year and remain sluggish in the second quarter.

(a)
Strong Market
Lightened Debt Burdens
Increasing Financial Well-Being
Financial Crisis
Expected Slow in Economic Growth
(b)

Figure 1: Two uses cases for the event detection tool, applied to two sentences containing five events in total (a). A causal graph manually extracted from the examples (b). The main goal of this work is to solve the first step (event detection) for a causal structure extraction tool. The next step will involve automating the second step (causality extraction).

Previous work was limited to a fixed taxonomy of events. In this scenario, a model trained on a set of event types cannot extrapolate a prediction to a new event type. Another limitation of many models is their inability to distinguish very recent or ongoing events reported in the news from historical, future, hypothetical or other forms of events that are neither fresh nor current. To illustrate this, take the word "crisis" in the following sentences:

fernando.delbianco@uns.edu.ar (Fernando Delbianco), ftohme@criba.edu.ar (Fernando Tohmé), agm@cs.uns.edu.ar (Ana Maguitman), eem@cs.dal.ca (Evangelos Milios)

1. The current *crisis* will accelerate the end of paper money.
2. There will not be a *crisis* in the foreseeable future.
3. The same trend could be observed during the Global Financial *Crisis* more than a decade ago.
4. Any financial *crisis* is catastrophic, and safeguards must be drastically improved to mitigate the risks of a future *crisis*.

In our approach, only the reference to a “crisis” in sentence 1 is considered an event trigger, while the other mentions to the same word are not. This is guided by our ultimate goal of detecting causal relations between events reported in the news. However, this context-sensitive definition of event is not handled correctly by most existing proposal.

Motivated by the limitations of previous models, we outlined a more general definition of an event. Guided by this definition, we manually labeled a dataset for training and testing purposes. Using this dataset, we developed an RNN model for event prediction. Since there are no previous studies on this dataset, we also replicated a baseline model from the state-of-the-art for comparison [32].

The contributions of this paper to the ED task can be summarized as follows.

1. Firstly, we design and implement an RNN model with performance comparable with the state-of-the-art. This model includes a new attribute that, to the best of our knowledge, was never used before for the ED task, namely BERT embeddings.
2. Secondly, we present and make available a new dataset with a more general/flexible definition of what an event is. We also elaborate on a baseline model by replicating a state-of-the-art model ([32]) on our data.
3. Finally, we present an extensive empirical evaluation with different architectures and hyperparameters for both the baseline and our model. We performed an ablation test as part of the preliminary studies to measure the impact of each feature on the performance. Guided by these studies, we choose the best set of features for maximizing F1-score. The results in the held-out showed a noticeable improvement by using the features selected during the ablation test.

The code for the proposed and baseline models, as well as the dataset are made available to the research community for reproducibility and data reuse.[1]

The outline for the rest of the paper is as follows. In Section 2 we present background concepts and related work. In Section 3, we introduce our RNN model for event prediction, including a description of the features used, the architecture and hyperparameter settings. In Section 5, we present the results and the discussion. Finally, in Section 6, we discuss the conclusions and outline future work.

[1]The code is available at http://cs.uns.edu.ar/∼mmaisonnave/resources/ED_code/ and the dataset is available at http://cs.uns.edu.ar/∼mmaisonnave/resources/ED_data/.

2. Background and Related Work

The tasks of ED and EE require raw texts labeled with precise information about entities, event triggers, and event arguments. The construction of such a dataset is labor intensive and time consuming. Therefore there are not many publicly available datasets, and many of them are very small (few hundreds of documents) in contrast with available datasets for other classification tasks, such as image classification [9] (with more than 14 millions images) and NLP classification, such as [22] Newsgroup 20 dataset (with nearly 20,000 texts).

As part of competitions and conferences, many datasets become available, and the first models for EE and ED were tested and published. Different approaches and types of models were tested. In this section, we review the existing dataset for the ED/EE task and the different models proposed in the literature. First, in section 2.1, we review some of the existing datasets for the EE/ED task. Next, in section 2.2, we review the existing models for the task.

2.1. Existing Datasets for the EE and ED task.

The most widely used dataset for ED is the ACE 2005 Multilingual Training Corpus [43], which contains the complete set of English, Arabic, and Chinese training data for the 2005 Automatic Content Extraction (ACE) technology evaluation [11]. The corpus consists of data of various types of annotations for entities, relations, and events by the Linguistic Data Consortium (LDC). The objective of the ACE program was to develop technology to support the automatic processing of human language in text form for the task of Information Extraction (IE). The annotation guidelines depict in detail the full taxonomy of valid event types and subtypes (i.e., DIE, ATTACK, MEET).

Although there are other datasets available (e.g., TimeML [38], SentiFM [19]), to the best of our knowledge, all the datasets for the EE task have a fixed taxonomy of valid event types. The main disadvantage of a fixed taxonomy is the problem of extending the model to new event types. A model trained with the ACE Corpus can only detect a predefined set of event types, and it is going to be probably biased towards the most frequent event type.

2.2. Existing Models for the EE and ED Tasks.

The state-of-the-art approach for ED is to generate a neural-based classifier with a class for each possible event type, and an extra class for non-events. These classifiers use a wide variety of features as inputs for the classification task.

Before the advent of the neural-based models, most of these features were handcrafted and more complex. For example, in [24] the authors designed and studied a rich set of attributes with local and global information about each token to boost the performance of the classifier. These richer and complex features were often the result of the application of NLP tools. Although the NLP tools provide a useful source of information, they are not error-free, which means that any error in the initial stages propagates to the model.

With the advent of neural models and the availability of continuous representations for words and sentences learned in an unsupervised fashion from large corpora (such as word and sentence embeddings), the features used in the models changed radically. A lot of the inputs from newer models are unsupervised representations automatically learned from big corpora (e.g., Word2vec [29] and Fasttext [3]), and many of these representations can keep improving during training through gradient descent. According to [4], the approaches for the ED task fall into one of three categories: pattern-based [21, 15, 39, 40, 45], feature-based [14, 7, 42, 20, 35, 25, 18, 16, 24, 5], and neural-based [6, 32, 30, 31, 13]. The first two rely on sophisticated handcrafted rules, patterns, and features, as well as in NLP tools. The third category relies on neural networks for both feature representation and token prediction. In this paper, we focus on the last category, which solves a lot of the problems with the first two categories mentioned above, while achieving state-of-the-art results for both ED and EE.

Several works study the task of ED as a standalone task [34, 26, 12, 33, 31, 32, 44], while others use and analyze ED as a component of an EE system [46, 27, 41, 17, 30, 6, 24, 2]. In the literature, these EE systems follow one of two possible architectures: pipelined [6, 20, 25, 16] or joint [46, 27, 41]. In the pipelined architecture, the ED task is the first step, in which the trigger word is detected and classified. Afterward, the system performs the rest of the EE task by extracting the arguments for those triggers. In the joint architecture, several proposals employ the joint architecture in which argument extraction is part of the trigger extraction phase and vice versa. By carrying out these stages together, they provide information to each other to boost the performance of the overall EE system.

Since, in this work, we aim at solving ED and not EE, we only address the trigger detection task and not the argument extraction. Furthermore, the dataset that we manually labeled does not include information about arguments, so this work is focused solely on ED. Therefore, we evaluate our model against the state-of-the-art in ED, instead of considering approaches that perform ED+EE. We cannot compare our work with the last group because more information is available for making predictions for the ED task (information about the arguments of events). Based on these considerations and because of its simplicity and comparable performance with the state-of-the-art neural models, we selected and replicated the model proposed in [32] as a baseline for comparison.

3. RNN Model

We used a Recurrent Neural Network (RNN) for the task of ED, which consists of classifying each word (or token) into one of two possible categories: event, or non-event. We conducted several experiments combining different hyperparameters, features, and architectures. In this section, we will review the different configurations tested and the intuition behind each selection.

Features. To be able to detect events in natural language text, we hypothesize that syntactic, semantic, and grammatical information is needed. To represent the semantics of each token, we used a well-known continuous representation for each word, Word2Vec [28] (W). Note that we use Word2vec instead of other word embeddings (e.g., FastText [3] and Glove [36]) for the sake of comparison, as Word2vec is the word embedding used in our baseline [32].

To encode the syntactic information, we first used the Spacy library[2] to identify the dependency tree (D). For each token, we extract the dependency relation with the head token in the dependency tree. Using that information, we trained a 10-dimension supervised Keras embedding layer. This layer was initialized with random weights and incrementally improved the representation along with the training of the rest of the network via gradient descent. The grammatical information was encoded using two different embeddings. Both embeddings represent information retrieved through Spacy's Part-Of-Speech tagger. We used both the simplified Part-Of-Speech tag version (P) and the detailed one (T) of the Spacy NLP library. Since this information is categorical it required a continuous representation. As we did in the case of the syntactic information, we used a first embedding layer to transform these two categorical variables into two vectors of 10 dimensions each. In the same way as we did with the D variable, we used the Keras embedding layer, which improves the representation during training in a supervised fashion.

We used the Spacy Named Entity Recognizer tagger to include entity information for the ED task in the form of a separated feature (E). We represented each word as a two-part tag. The first part is the IOB notation (beginning (B), inside (I), outside (O)), and the second is the type of the entity. Since this information is categorical, we transformed this feature into a continuous representation. As we did for other features (T, P, D), we used an embedding layer to transform a one-hot encoding of this categorical input into a 10-dimension vector. This layer is initialised with random weights and is fine tuned while training for the ED task.

The task of detecting an event mention is one of the most challenging IE tasks. For a model to be able to detect event mentions with performance comparable to a human reader, it needs to analyze natural language text with a high level of understanding of grammar, syntax, and semantics. Such a task is inherently complex. When the classification task requires high levels of abstraction and complex reasoning, it is usually solved using more complex and deeper models. However, this is not possible for the ED task due to the small amount of training data available. In this context, a useful approach is to use transfer learning, to exploit the availability of datasets for other tasks, and import the knowledge into the ED task.

We implicitly use transfer learning in the various inputs of the model. Firstly, by using the pre-trained Word2vec vectors trained on the Google News dataset, we are incorporating semantic information extracted from a large dataset. Secondly, by using the Part-of-Speech tagger and the Dependency tagger from the

[2]Spacy NLP library https://spacy.io/

Spacy library, we are including grammar and syntactic information. This information comes from other IE fields, where there are large corpora of text labeled for part-of-speech tagging, and dependency tagging. By using this toolkit, we exploit this information for the ED task.

Although we used Word2vec embeddings for the sake of comparison, we also tested other embeddings to provide a final model with the best possible set of features. We used 96-dimension context-sensitive tensors provided by the Spacy Library (Sp). A Spacy context-sensitive tensor is a 96-dimension vector, which is the internal state of the neural model used for NLP by the Spacy library. We will refer to these vectors as Spacy word embeddings. According to the Spacy documentation, these embeddings do not yield results as good as the pre-trained word embeddings. However, they have the important characteristic of being context-sensitive. To test how important is the semantic and the context of each token, we used both classical word embeddings (Word2vec) and the context-sensitive tensors provided by Spacy as inputs. We use the Spacy tensors as contextual-word embeddings and as an input to the RNN model. Each token in each context is represented by a 96-dimension vector that is not fine-tuned during training. On the other hand, Word2vec embeddings represent each word by means of a 300-dimension vector and differently from the Spacy word embeddings. This 300-dimension representation is fine-tuned as the model is trained for ED.

One limitation of the traditional word embeddings (like Word2vec and FastText) is that there is a single vector to represent each word. Therefore, the term "bank" is associated with only one vector, where all the different semantics are mixed. In this scenario, we will have the same vector for the term "bank" no matter whether the words occurs in the sentence "I went to the park and sit on a bank." or the sentence "I need to go to the bank."

Contextual embeddings, such as ELMo [37] and BERT [10], were proposed to solve this problem of mixed semantics. This new type of embeddings achieves ground-breaking performance on a wide range of natural language processing tasks. We hypothesize that having vector sensitivity to the context is crucial for the ED task, and it will bring a boost in performance. To illustrate this hypothesis, consider the following two sentences: "The firm had to fire employees for ..." and "Fire burns a home near Brainerd airport." They both use the word "fire", written in the same way, but with totally different meanings. Word2vec will only have one vector for this word. In many cases, this difference in semantics can be the reason behind many false-positives and false-negative cases. Motivated by this intuition, we incorporated pre-trained contextual word embeddings, in particular BERT embeddings (B), as an input to the model.

Another problematic situation occurs with a word that preserves its semantics across different sentences but that can refer to an event trigger or not, depending on its context of use. This was illustrated in Section 1 with the word "crisis", which can be used to refer to a recent or ongoing event (event trigger) or to a historical, future, hypothetical or other forms of events that are neither fresh nor current (non-event triggers). Following this intuition, we incorporate for each token a 768-dimensional vector feature representing a contextual embedding for the whole sentence. Note that each token in the same sentence will have the same contextual sentence embedding as input. The intuition behind using a common contextual sentence embedding for each token in a sentence is that sentences usually give information either about past, future and hypothetical situations (which were irrelevant to us), or about current events (which are the ones that we want to detect). It was uncommon for a sentence to mix these two types of information. So, sometimes to determine if the current token was an event, it was necessary to know if the sentence was describing past, hypothetical, future or present circumstances. Guided by this intuition, we hypothesize that a representation of the context of use of the tokens provided by contextual sentence embeddings can boost the performance of the ED task. So the final feature we incorporate is a contextual sentence embedding (S)built by adding up the BERT embeddings for each token in the sentence.

In summary, we use eight input features. In the first place, we use classical Word2vec embeddings to represent the semantics of a word. The dep, tag, and pos embeddings extracted using NLP tools are used to encode syntactic and grammatical information. Another input is represented by entities identified by the Spacy NER tagger. We implemented these five inputs using five Keras layers of size 300, 10, 10, 10, and 10. We initialized the last four layers with random weights, while we use the pre-trained embeddings for initializing the first layer. The sixth feature is the 96-dimension Spacy contextual word embedding. The seventh and eighth features are the BERT word and sentence embeddings, of size 768 each. Since it is not feasible to model these three contextual inputs as a Keras embedding layer, we directly pass the embeddings to the next layer in the model. Since each word has a different context, each word of the corpus has a different contextual embedding. The one-hot encoding would require a length of the total count of tokens in the dataset (∼84K). Although such an encoding is possible, it is not useful and may result in inferior performance given that it may lead to overfitting due to the large number of parameters. Since these inputs are not modeled with a Keras layer, they will not be fine tuned during training for the ED task. A summary of the features used for the RNN models is presented in Table 1.

Architecture. We chose an RNN architecture based on Long-Short Term Memory (LSTM) cells to exploit the dependencies between previous and subsequent tokens for the classification of the current word. The inputs for each word, which are seven embeddings, are all concatenated together to form a vector of 1962 dimensions. We added a Dropout layer after the concatenated embeddings. Following the dropout layer, we added a Bidirectional LSTM (Bi-LSTM) layer with fifteen hidden units. A final dense layer with only one hidden unit was attached to the Bi-LSTM Layer. The output of this final layer is the prediction. Although this architecture is the one used during the experiments on the data held-out for testing, we tested other architectures, with different numbers of Bi-LSTM layers, with different numbers of hidden units. The results of these experiments, with varying numbers of Bi-LSTM layers, can be found in Appendix Appendix B.

Hyperparameters. For the embedding layers we use the default configuration, only changing the random initialization for the Word2vec embeddings. We use a probability of 0.1 for the Dropout layer. We tested different numbers of Bi-LSTM layers and hidden units, including one three Bi-LSTM layers model, two with two Bi-LSTM layers, and five single Bi-LSTM layer models. We set up every Bi-LSTM using the default configuration, adding only L1L2 regularization with values of 0.001 for both L1 and L2. For the final experiments on the data held-out for testing, we used a single Bi-LSTM layer architecture with fifteen hidden units, which was the architecture with the best performance during the preliminary studies. Finally, we set up the dense layer using the sigmoid activation function. The number of layers and hidden units for the Bi-LSTM layers were selected based on several pilot studies that are described in appendix Appendix B.

4. Experimental Setup

4.1. Dataset

In our work, we are not interested in pre-defining all the possible event types. Instead, we only label each word as *event trigger* or *non-event trigger*, transforming the ED task into a binary classification task. We consider as *event trigger* any ongoing real-world event or situation reported in the news articles. It is important to distinguish those events and situations that are in progress (or are reported as fresh events) at the moment the news is delivered from past events that are simply brought back, future events, hypothetical events, or events that will not take place. In our dataset we only labeled as *event trigger* the first type of event. Based on this criterion, some words that are typically considered as events are labeled as *non-event triggers* if they do not refer to ongoing events at the time the analyzed news is released. Take for instance the following news extract: "devaluation is not a realistic option to the current account deficit since it would only contribute to weakening the credibility of economic policies as it did during the last crisis." The only word that is labeled as *event trigger* in this example is "deficit" because it is the only ongoing event refereed in the news. The word "devaluation" is labeled as *non-event trigger* as a devaluation may not take place. Similarly, the word "weakening" is a *non-event trigger* as it is a hypothetical event. Finally, the word "crisis" is considered a *non-event trigger* as the news refers to a crisis from the past. Note that the words "devaluation", "weakening" and "crisis" could be labeled as *event triggers* in other news extracts, where the context of use of these words is different, but not in the given example.

The rationale for the adopted criterion is that the ultimate goal of our work is to detect causal relations from digital media based on the identified events. As a consequence, the focus is on ongoing events and situations only. The event detection model proposed and evaluated in this work has the ability to capture both context-independent and context-dependent cues and, as we will see later, it is able to effectively identify ongoing events from those that are not in the most common situations.

Features		
Abbr.	Size	Description
W	300	Pre-trained word embeddings: Word2vec.
P	10	Part-Of-Speech tag embeddings, simplified version.
T	10	Part-Of-Speech tag embeddings, detailed version.
D	10	Dependency parser tag embeddings.
E	10	Entity tag embeddings. Computed using the Spacy NER tagger.
Sp	96	Spacy contextual word embeddings.
B	768	Pre-trained contextual word embeddings: BERT embeddings
S	768	Contextual sentence embeddings. Computed adding the B embeddings.

Table 1: Description of all the features used by the RNN models. Each of the RNN models used in this work uses these features, or a subset of them, as input. The first two are fixed pre-trained vectors. The second two are pre-trained vectors that are fine-tuned during the training. The last three are randomly initialized vectors that are updated during the training.

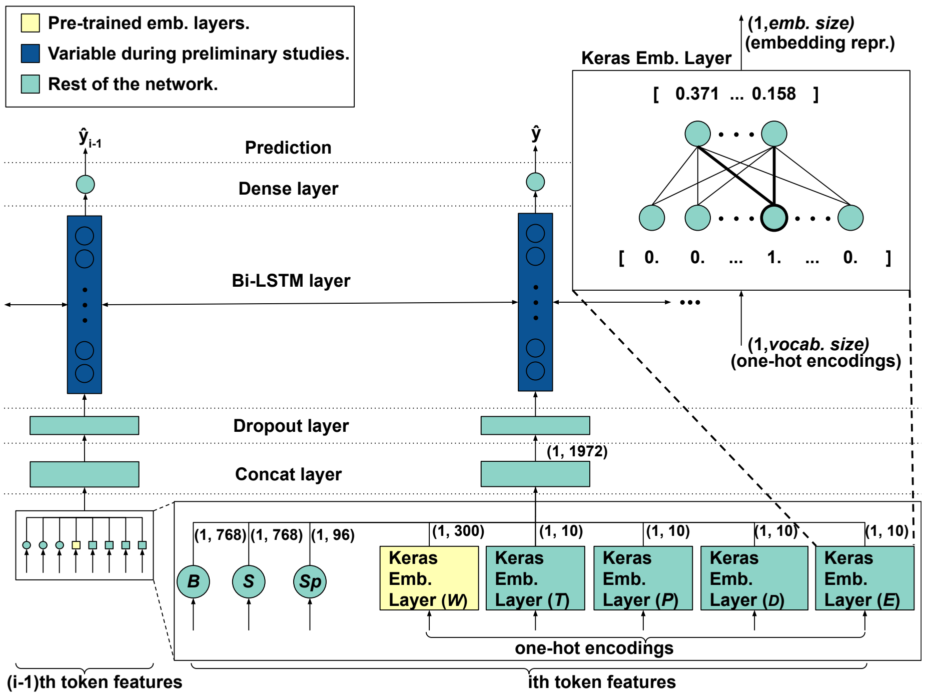


Figure 2: **Outline of the proposed architecture (RNN)** used in the experiments on the held-out data. From bottom to top. A representation of the eight inputs, for the ith and the $(i-1)$th token. For each token, three of the inputs are vector representations (B, S, Sp); the other five are one-hot encodings that enter five Keras embedding layers. Each of thee dense layers is depicted in the top-right corner diagram. One of these embedding layers start with pre-trained weights (W), the others with random weights. After the embedding layer, the eight vectors are concatenated together in a single 1972-dimensional vector. This vector enters a dropout layer. The output of the dropout layer enters a Bi-LSTM layer with fifteen hidden units. During the preliminary studies, we tested different numbers of Bi-LSTM layers and hidden units. For the held-out data, we use the configuration depicted in this Figure. The output of the Bi-LSTM layer is the input to a last dense layer, which makes the prediction.

We tokenized the full New York Times (NYT) archive using the Spacy NLP library and divided the dataset into sentences. Afterward, we selected a subset of those sentences for labeling. We choose three episodes of real-world crises: the Mexican peso crisis of 1994, the Russian financial crisis of 1998, and the Asian financial crisis of 1997. We set up the search engine Lucene[3] (with the default configuration) to search sentences related to these three episodes. We performed a search using keywords manually selected by experts. Examples of these keywords include "Mexico", "Crisis", "Debt", "capital flight", "devaluation". From the obtained results, we randomly selected two thousand sentences. Also, we randomly selected from these results a separate set of two hundred sentences that were held out for testing purposes.

We developed a simple active learning tool to assist the labeling process of the training and validation data. This tool used an early prototype of the RNN model used for event prediction (described in section 3) to suggest labels. The tool marked each possible event trigger candidate in the text while the user was in charge of correcting the suggestions by removing candidate event triggers proposed by the tool or adding events that were not suggested. We employed a consensus-based approach to minimize errors during the labeling process. Each sentence was presented to four users for labeling, along with the corresponding suggestion generated by the model. Then the four users had to agree on keeping a suggested label, removing it or adding a new one. The whole process took a total of fifteen sessions of approximately two hours each.

To avoid biasing the users' decisions when tagging events in the held-out set used for testing, we had to adopt a different approach from the one used for labeling the training and validation set. Therefore, the process of labeling the testing set was not assisted by the RNN model. Instead, the four users were presented with sentences with no suggestions provided by the tool and had to reach a consensus on which words had to be marked as event triggers.

4.2. Baseline Model

As detailed in section 2, existing datasets for ED have several drawbacks, and those drawbacks make them unsuitable for our work. Therefore, as explained in section 4.1, we designed and built a dataset specifically for our work. Since no previous model was validated using our dataset, existing metrics, and results from the ED field are not directly comparable to ours. For this reason, to be able to compare our work with other works from the literature of ED, we had to choose and replicate an ED model from the state-of-the-art, to be used as a baseline.

Baseline selection. The Convolutional Neural Network (CNN) presented in [32] was used as the baseline. Despite being proposed several years ago, this model is still competitive with the available state-of-the-art methods. For example, it presents an F1-score of only 4.1 percentage points lower than more

[3]Apache Lucene https://lucene.apache.org/

Full Dataset (training/validation + testing)	
Sentence Count	2,200
Word Vocab. Size	8,647
Entity (E) Vocab. Size	34
Part-Of-Speech Simplified (P) Tag Vocab. Size	16
Dependency Parser (D) Tag Vocab. Size	47
Part-Of-Speech Detailed (T) Tag Vocab. Size	47

Table 2: Statistics about the vocabularies found in the full manually labeled dataset for ED (training/validation + testing). This dataset consists of 2000+200 sentences extracted randomly from a subset of sentences of the full NYT dataset (1987-2007). The subset consists of sentences related to three relevant economic events. We manually labeled each word of each sentence either as an event or as a non-event trigger. For the count of unique words, the only NLP tool required was a tokenizer. We included the punctuation marks found in the texts as part of the word vocabulary. We used a Named Entity Recognition (NER) tagger to build the vocabulary of unique entities, and a Part-Of-Speech and Dependency tagger to build the following three vocabularies (T, P, D). We used the taggers and tokenizer from the Spacy NLP library.

Metric	Training/Validation		Testing		Full Dataset	
	Total	Avg. per Sent	Total	Avg. per Sent	Total	Avg. per Sent
Token Count	76,629	38.31	7,382	36.91	84,011	38.19
Word Count	67,032	33.52	6,442	32.21	73,474	33.40
Entity Count	11,502	5.75	950	4.75	12,452	5.66
Event Count	5,119	2.56	416	2.08	5,535	2.52

Table 3: Total number of tokens, words, entities, and events found in the two different partitions of the dataset (and in the sum of both). The training/validation portion is the one we used for building the different training and validation sets randomly. The testing portion is the data held out for testing the proposed and baseline model. Therefore, we do not use that portion for training nor for choosing the hyperparameters. The active learning system was not used to assist the labeling of the data held out for testing to avoid introducing bias on this set. We report token count (words+punctuation), words, entities (extracted with the Spacy NER), and Events (manually generated). For each metric and dataset portion, we report the total and average per sentence.

recent models, such as [34]. The selected baseline reports an F1-score of 69% for the ED task using gold-standard entity annotations, while the model from [34] reports an F1-score of 73.1% on the same dataset (ACE 2005 Corpus) and using the same gold-standard entity annotations. The authors do not report in [34] the result of the more recent neural model for predicted entities. However, they do report in [32] the F1-score for predicted entities for the model we use as our baseline. For this setting, the model achieves an F1-score of 67.6%. Several other models are not considered as baselines because they perform ED as part of the EE task. Consequently, they are not directly comparable with our model as they make use of information that is not used during the ED task.

Although we tried to replicate the model exactly as it was in the original paper, to adapt it to our dataset, we had to make some minor changes. Furthermore, since the code was not available, some minor implementation details may not be the same as in the original model. We had to make some decisions about some aspects and configuration of the models which were not explicit in the original paper. For example, we had to decide which NLP tools to use for entity extraction. When facing these decisions, we always tried to choose the option that yields the best performance based on results reported in the literature. For example, when we had to choose an NLP tool, we chose for both models the Spacy library which, as far as we know, is one of the best tools available for NLP[4] [8]. In the remainder of this section, we describe in detail the baseline model used. We also describe some minor changes and decisions that we had to make, explaining the rationale behind them.

Features. As in the original paper, we use three input features for the baseline model. First, we use the Word2vec embeddings as in our RNN model (W). Second, we use entities embeddings (E). For building this feature, we use the Spacy Named Entity Recognition (NER) tagger and a Keras embedding layer for building the embeddings, using the former to get categorical variables (tags) and the latter for transforming these variables into vectors. We chose Spacy as our NER tagger, as in the original paper there is no mention of a specific tool, and because of the state-of-the-art performance of the Spacy library in several NLP tools. The last feature employed for the baseline model is a position embedding (Po), which represents the relative position of each word with respect to the current token under classification. The word embedding layer starts with a representation learned from the pre-trained Word2vec vectors, while the other two start with random weights, as in [32]. The three layers are updated while training for the ED task.

Architecture. The architecture used in [32] is a one layer CNN, followed by a max-pooling layer and lastly a dropout layer followed by a dense layer for the prediction. The inputs of this network are three lookup tables, with the three types of embeddings. The representation in theses lookup tables improves along with the training for the ED task. We replicated the same architecture and behavior, by replacing the lookup tables with embedding layers that meet the

[4]https://spacy.io/usage/facts-figures

same purpose: storing and providing a vectorized representation, and improving the representation while training for the ED task. The remaining of the network follows the same architecture as in [32].

Hyperparameters. While many of the hyperparameters remain the same, since our data is different from the one used in [32], the window size used had to be changed to better suit the data. Since in our dataset, each data item is a sentence and not a whole document like in ACE 2005 (the dataset used in [32]), the window sizes had to be adjusted. We tried with smaller window sizes, 1, 3, 5, 11, 21, and also for comparison sake we tested with window size 31. We use the same number of filters (150), and the same size for the filters (2, 3, 4, 5), as in the original paper. We used the sigmoid activation function for the final dense layer to use the same performance metrics as in our RNN model. We use, as in the original paper, batch size 50, a probability for the dropout layer of 0.5, and we set the hyperparameter for the l2 norms to 3. We used the binary Cross-Entropy loss function and Adam's optimizer.

5. Results and Discussion

In this section, we present the results and discussions for seven different variations of the proposed model and four different variations of the baseline model. For each variation and each model, we randomly split the training/validation part of the dataset into different training and validation subsets. For each variation, we trained the model until the model did not improve the F1-score in the validation set for 400 epochs (Early Stopping with patience 400). We used consecutive seeds from 1 to 10 to guarantee the replicability. The model with the best performance in the validation set was selected and used to make the predictions in the data held-out for testing. It is important to notice that we used the same held-out data to test all the models and this data was never used during the training stage. In this section we present and discuss the average performance of each model variation on the validation splits and testing data. Because we have seven variations of the proposed model and four variations of the baseline model, we report results for a total of eleven different models. For selecting each of these model variations, we run several preliminary studies. The preliminary studies and their discussion are presented in the appendices.

For each model variation, we report the average metrics for the ten trials, which include the average sensitivity, specificity, and harmonic mean between these two metrics, namely F1-score. We do not report the average accuracy for the model, because, in the presence of highly unbalanced data (93.41% are non-events), the accuracy is a misleading metric. Measuring the accuracy is not useful, considering that a model that always predicts *non-event triggers* has more than 90% accuracy. For these reasons, we focus our analysis on the more balanced F1-score metric. To thoroughly analyze the F1-score metric on the testing set we also report the confidence intervals (CI) at 95% level of confidence and the p-value for a t-test between the model considered and the best model of each table. For example, in the first row of Table 4, we report the p-value of a single-tail t-test between the F1-score values achieved by the

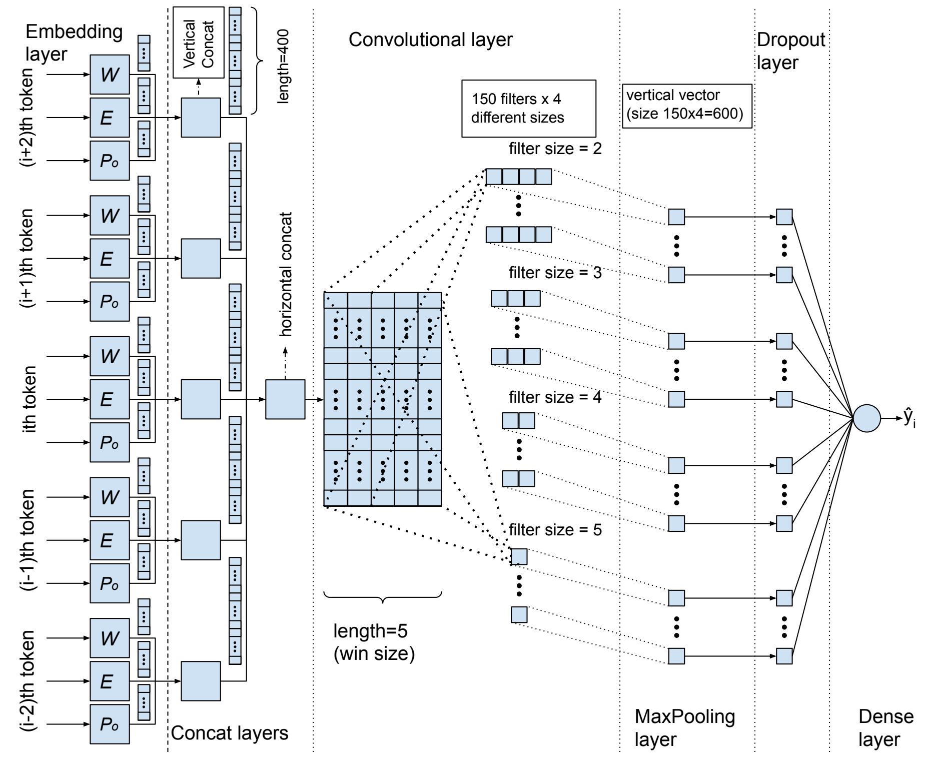


Figure 3: **Outline of the baseline model (CNN)** with window size 5. Descriptions are given from left to right. (a) Inputs. A representation of the three different inputs (Word embedding W, Entity embeddig E and Position Embeddig Po) for the five tokens in the window. Each input is a one-hot encoding. (b) Embedding layer. Input enters a Keras embedding layer (a depiction of this layer can be found in Fig. 2). (c) Concatenation layers. We concatenated the resulting embeddings in a single 400-dimension vector that represents each token. Another concatenation layer stacks these five vectors into one matrix. (d) A convolutional layer applies 600 filters of four different sizes (2, 3, 4, and 5) with 150 filters for each size. (e) MaxPooling layer. A max-pooling layer is applied afterward. (f) Dropout and Dense layers. Finally, the last two layers are a dropout and a dense layer. The dense layer is the last one of the network and is the one that makes the prediction.

the first model (model 1) and the best model of the table (model 7). We used the F1-score metric on the testing data for the t-test.

5.1. Experiments on the Test Set

For the proposed model (RNN), we run ten trials for seven different sets of features (models 1 to 7). The first model is the model with the full set of features (as depicted in Table 4). The following models are models with the full set of features removing some particular set of features for each experiment. Although we tried different numbers of Bi-LSTM layers and hidden units, we only evaluated on the testing set the architecture that achieved the best performance during the preliminary studies, i.e., the architecture with a single Bi-LSTM layer with 15 hidden units. The architecture is depicted in Figure 2 and the results are reported in Table 4.

We conducted these experiments for two main reasons. First, to test the hypothesis that the contextual word embeddings (B) are having a significant positive impact on the overall performance. We tested this hypothesis during the preliminary studies, and we found further evidence during the experiments on the testing dataset. The significant drop in performance (of 11.7 percentage points) from including or excluding the contextual word embeddings (model 1 vs. model 2) is indicating the importance of these features for the ED task.

Second, we carried out the rest of the experiments (models 3 to 7) to asses the impact of several of the proposed features to the overall performance. We measure the contribution of each feature during the preliminary studies. During the experiments on the testing dataset, we found further evidence that several inputs are not contributing to the performance in the presence of the other features. These results allowed us to simplify the model and obtain even better results.

Model 3 allows to analyze the impact of the grammatical information, extracted with the Part-Of-Speech (T, P) and Dependency Parser (D) taggers. The impact of the model excluding these three features showed a negligible drop in performance compared to the model with all the features (model 1). We believe that these results can be the consequence of three influencing factors. Firstly, an inadequate embedding representation due to the small amount of data. Note that, unlike the pre-trained embedding Sp and W, the T, P, D embeddings are randomly initialized and improve during training in a supervised fashion using the available dataset consisting of 2000 sentences. Secondly, the grammatical information can already be available in the other features (i.e., B). Finally, the NLP technologies used for extracting the tags have an inherent error that could be harming the performance. For the rest of the models, models 4 to 7, we exclude the T, P, and D features to test our hypothesis that the good performance is mainly due to the contextual embeddings. To test this hypothesis, besides excluding the T, P and D features, we exclude the Word2vec word embeddings (model 4), and the Contextual Spacy word embeddings (model 5), the entity embeddings (model 6), and all the previously mentioned features (model 7).

Models 4 to 6 showed a negligible variation in performance compared to the model with all the features (model 1). These variations were of +1.4, +0.7, and -0.1, respectively. These results provide evidence to support our hypothesis that the contextual embeddings are crucial for the ED task and are the features that most contribute to the good results. Model 7, which excludes all the attributes except the contextual word and sentence embeddings, is the model with better performance. This result shows that the other features are not relevant in the presence of these two other features, and they were adding noise and complexity to the model. This simpler model, with only the contextual information, performed better than all the previous models. The statistical analysis between model 7 and the others shows that all the differences are significant at the level of 90%, namely all the p-values are smaller than 0.1. Furthermore, except for model 4, all the p-values are smaller than 0.05, showing a statistical difference at the level of 95%.

The baseline model (CNN) was run for ten trials with two different window sizes (1 and 11), and for each window size, we trained models with two different sets of features. For both window sizes, we run experiments with and without the entity embedding, but always including the word embeddings. For windows size 1, we excluded the position embeddings. The results of these experiments are presented in Table 5.

The results for both window sizes show a negligible impact of the entity embedding. Furthermore, in both cases, the impact is negative, showing better performance using the word embeddings alone. For window size 11, the results suggest a tendency for both sets of features to overfit the data, showing a drop in performance when we use the model for making predictions on the testing data. The best baseline model shows an F1-score of 56.9% on the testing data, which is significantly lower than the performance of all the proposed models.

5.2. Discussion

In this section, we discuss the conclusions we derive from the experiments with the variations of the proposed model (models 1 to 7) and the models based on the selected baseline (models 8 to 11). In particular, we focus the discussion on the best of the proposed models (model 7), the model without the contextual word embeddings (model 2), and the two best baseline models (models 8 and 9). To compare the proposed and baseline models, we present in Table 6 the p-values of single-tail t-tests between the average F1-score for each model against each other on the testing data.

In the rest of this section we present observations on the behaviour of the best performing models 2, 7, 8 and 9.

Model 2. We hypothesize that the contextual embeddings are a crucial factor for boosting the performance of the proposed models. Therefore, in model 2, we exclude only the contextual word embeddings to asses the impact of them

[5]We describe the features used by each model by indicating which features are removed from the full set of features (all), where the full set of features is $\{B, S, W, Sp, P, T, D\}$.

RNNs on the Testing Dataset										
Model	Architecture	Features	validation			testing				
			$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	F1 CI	p-value
1	⟨15⟩	all [5]	0.696	0.945	0.712	0.667	0.931	0.676	± 0.026	0.024
2	⟨15⟩	all-$\{B\}$	0.545	0.957	0.591	0.522	0.948	0.559	± 0.046	0.000
3	⟨15⟩	all - $\{T, P, D\}$	0.677	0.949	0.697	0.654	0.935	0.666	± 0.031	0.012
4	⟨15⟩	all-$\{T, P, D, W\}$	0.703	0.948	0.718	0.686	0.935	0.690	± 0.018	0.076
5	⟨15⟩	all-$\{T, P, D, Sp\}$	0.689	0.945	0.706	0.672	0.931	0.683	± 0.013	0.007
6	⟨15⟩	all- $\{T,P,D, E\}$	0.686	0.949	0.706	0.662	0.936	0.675	± 0.023	0.012
7	⟨15⟩	all - $\{T, P, D, Sp, W, E\}$	0.726	0.943	**0.734**	0.706	0.928	**0.704**	± 0.012	—

Table 4: **Average performance of the proposed model (RNN)** for seven different sets of features using an architecture with a single single Bi-LSTM layer and 15 hidden units (indicated as ⟨15⟩ with this notation explained in Appendix B). For each set of features, we run ten trials and report the average sensitivity, specificity, and F1-score for both the validation and testing data. For the testing data, we also report the confidence intervals at a 95% level of significance for the F1-score. We also report the p-value of a single-tail t-test for each model against the best model of the table. The test is performed between the F1-score on the testing data for each of the two models. A small p-value gives evidence to reject the hypothesis that the models have the same F1-score, indicating that the best model is statistically significantly better.

In this experiment, we tested two different hypotheses. First, we examine our intuition that some features are not useful for prediction on the presence of others. As the preliminary studies show several features are not useful for the ED task as long as the other features are present in the model. Furthermore, removing such features improves the performance by eliminating the noise they add to the task. We tested this hypothesis with the experiments for models 3 to 7 using model 1 as the baseline. Where model 7, with all the features excluded except the contextual word and sentence embeddings, showed the best performance. Secondly, we tested the hypothesis that the exclusion of contextual embeddings significantly impacts the overall performance, even more than any other feature. The second experiment provides evidence to support our second hypothesis.

CNNs on the Testing Dataset										
Model	Win Size	Features	validation			testing				
			$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	F1 CI	p-value
8	1	$\{W,E\}$	0.477	0.971	0.570	0.499	0.968	0.575	± 0.031	—
9	1	$\{W\}$	0.472	0.972	0.565	0.493	0.969	**0.569**	± 0.031	0.384
10	11	$\{W,E,Po\}$	0.507	0.963	0.596	0.326	0.960	0.394	± 0.028	0.000
11	11	$\{W,Po\}$	0.499	0.965	0.589	0.345	0.942	0.406	± 0.035	0.000

Table 5: **Average performance of the baseline model (CNN)** for two different window sizes, including and excluding the entity embedding input, and excluding the position embedding input when appropriate (for window size one). We run ten trials and report the average sensitivity, specificity, and F1-score on both the validation and testing data. We also report the confidence intervals at a 95% level of significance for the F1-score on the testing data and the p-values of a single-tail t-test of each model against the best model of the table. The test is performed to compare the F1-score of each of the two models on the testing data. A small p-value gives evidence to reject the hypothesis that the models have the same F1-score, indicating that the best model is statistically significantly better.

In these experiments, we try to determine the best configuration for the baseline chosen from the literature. The experiments showed a tendency to overfit the data for those models with window size 11. By using early stopping, we selected the best model for the training and validation data, but the results show a lack of generality in what the model learned for window size 11. On the other hand, for windows size 1, the evidence suggests that there was no overfitting, and the models were statistically significantly better than the other two.

		proposed model	baseline models	
		model 7 (all-$\{T,P,D,Sp,W, E\}$)	model 8 ($\{W,E\}$)	model 9 ($\{W\}$)
proposed models	model 2 (all - $\{B\}$)	0.000	0.264	0.344
	model 7 (all - $\{T,P,D,Sp,W, E\}$)	—	0.000	0.000
baseline models	model 8 ($\{W,E\}$)	—	—	0.384

Table 6: P-values of single-tail t-tests between the best two baseline models (models 8 and 9) and two of the proposed models (models 2 and 7). For each model, we compute the p-values of the t-test against each of all the other models. A high p-value implies that there is not enough evidence to reject the null hypothesis that the models have the same performance. Here we measure performance based on the F1-score on the testing data. For example, the t-test between models 8 and 9 has a p-value of 0.384. This high p-value indicates that the difference in performance of these two models is not statistically significant.
In other words, the model without the contextual word embeddings is close in performance to the baseline model. In contrast, the model with contextual word embeddings is statistically significantly better than all the others (having a p-value of 0.0 for all the t-tests). This result highlights the significant impact that the contextual embeddings have on the performance of the ED task.

on the performance. Since this model lacks the contextual word embedding, we expected a poor performance in comparison to model 7 (which has the contextual word and sentence embeddings). The poor performance of model 2, which is closer to the baseline than to model 7, provides evidence to support our hypothesis. The first row of Table 6 gives additional information supporting our conclusion since model 2 has a higher p-value for the baseline model and a smaller one for the other proposed model (model 7). These p-values indicate that we have more statistical evidence to assume model 7 and 2 are different, but less evidence to tell apart model 2 and the baselines. Moreover, the inclusion of the other features (W, Sp, T, P, D, E) are not useful for the ED task once the contextual word embeddings are included. We derive this conclusion from the fact that the best model only has B, S as features, indicating that once the contextual word (B) and sentence (S) embeddings are present in the model, other features can be removed with a positive effect on performance. In conclusion, we found evidence to support the hypothesis that contextual embeddings play a key role in the overall performance of the proposed models.

Model 7. We hypothesize that the models using contextual embeddings are superior in performance to the baseline and the proposed model without contextual embeddings. The experiments confirm our hypothesis. A p-value of 0.0 for the three t-tests involving model 7 provides evidence to support the fact that its performance is statistically significantly better than the performance achieved by models 2, 8 and 9.

Models 8 and 9. The high p-values in the t-tests of these two models suggest that there is little evidence to conclude that the models have different performances. Since the only difference between the two models are the entity embeddings, these results are indicating that the entity embeddings have a negligent impact on performance.

6. Conclusions

The first contribution of this work is the presentation of an extensive study on the use of RNN models for the ED task, as well as a baseline CNN model for the same task. We studied eight different types of features and their impact on the overall performance.

The ultimate goal of our research work is to identify causal relations from digital media based on the detected events. Since we define a notion of event that suits this goal and created our own dataset for the ED task it was not possible to rely on existing benchmarks for assessing the performance of the proposed models. Therefore, we also replicated and performed an extensive analysis of a state-of-the-art CNN model [32], which was used as baseline for comparison purposes.

The analysis of the proposed model showed promising results. The best of the proposed models based on an RNN architecture shows an improvement of 13.3% in F1-score with respect to the best baseline model on the testing data. Considering that some more recent approaches for ED are only a couple of percentual points ahead of our baseline (4.1% for [34]), we have enough evidence to believe that our model achieves a performance competitive to the state-of-the-art, even outperforming some of the most advanced models. It is also worth mentioning that, differently from some of the state-of-the-art models, the code for replicating our model is fully available.

By analysing the performance achieved by different combinations of features in our proposed model, we reach three conclusions. First, among the proposed models, the one with the worst performance was the model that includes all the features except for the the contextual word embeddings. Although superior to the baseline, this model was relatively close to it with a difference of only 4 percentage points. This result provides evidence to support our hypothesis that the contextual embeddings are better suited for the ED task than the other analyzed embeddings. This is an important result that highlights the usefulness of contextual embeddings for capturing contextual cues, which are crucial in the ED task. Further studies are required to evaluate the impact of contextual embeddings on other ED settings (using other datasets and other models).

Second, some features proved to have a negligible impact on performance, in the presence of other features. For example, grammar information captured using Part-Of-Speech and Dependency Parser taggers and represented using embeddings (T, P, and D) showed to have no positive impact on performance as long as contextual embeddings are accounted in the model. Furthermore, the model with the best performance was the model with three types of features only (B and S). We believe that this result is due to two possible reasons. The first reason is that the information contributed by some features is already captured by other features. For example, the non-contextual word embeddings (W) offer no improvement in performance when added to the model. The absence of a positive impact is not because these features do not carry useful information, but probably because this information is already provided by contextual embeddings. The second reason may be that the embedding representations for the poorly-performing features (T, P, D, E) were not sufficiently robust since these embeddings were learned from a small dataset. Note that the absence of a large dataset for learning embeddings does not affect the pre-trained embedding (W) as much as the other embeddings (T, P, D, and E), which were randomly initialized and trained in a supervised fashion using the training data available for the ED task. A third reason why some embeddings do not have a favorable impact on performance may be the inherent error of the NLP tools used. This issue arises for the T, P, D, and E embeddings, which we constructed using different NLP taggers (NER, POS, and Dependency Parser).

The analysis of the baseline model showed some limitations of the CNN models when compared to the RNN models. In particular, a detailed study of the window size for the context of the CNN models showed that the CNN models do not handle the context of each word properly. The fact that a larger window size decreased the performance of the model supports this conclusion. In particular, for window size 11, the results show a tendency to overfit the data.

The third conclusion we derive from the results is that although RNN models showed more flexibility and suitability for the ED task than CNN models, the major difference in performance between the baseline and the proposed models relied on the features used, in particular the contextual embeddings. In fact, the performance achieved by the RNN models that lack contextual embeddings was only slightly superior to that achieved by the baseline models. This indicates that although the improvements are due in part to the use of an RNN architecture, the major boost in performance relies on the features being used.

The second important contribution of this work is the construction of a dataset for the ED task using an active learning approach and the public release of the dataset. A major obstacle in the ED field is the limited amount of available labeled data. In this work, we implemented a simple active learning system for assisting the user in the process of labeling event data. As part of our work we have created a dataset of news extracts with event triggers manually labeled using a consensus-based approach. To the best of our knowledge, this is the first time Active Learning is applied to create an ED dataset and we consider this to be a promising approach to partially overcome the data scarcity problem in the ED field.

References

[1] Adedoyin-Olowe, M., Gaber, M. M., Dancausa, C. M., Stahl, F., and Gomes, J. B. (2016). A rule dynamics approach to event detection in twitter with its application to sports and politics. *Expert Systems with Applications*, 55:351 – 360.

[2] Ahn, D. (2006). The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

[3] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[4] Boroş, E. (2018). *Neural Methods for Event Extraction.* PhD thesis, Université Paris-Saclay.

[5] Bronstein, O., Dagan, I., Li, Q., Ji, H., and Frank, A. (2015). Seed-based event trigger labeling: How far can event descriptions get us? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 372–376.

[6] Chen, Y., Xu, L., Liu, K., Zeng, D., and Zhao, J. (2015). Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

[7] Chieu, H. L., Ng, H. T., and Lee, Y. K. (2003). Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 216–223. Association for Computational Linguistics.

[8] Choi, J. D., Tetreault, J., and Stent, A. (2015). It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 387–396.

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[10] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[11] Doddington, G. R., Mitchell, A., Przybocki, M. A., Ramshaw, L. A., Strassel, S. M., and Weischedel, R. M. (2004). The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, page 1. Lisbon.

[12] Duan, S., He, R., and Zhao, W. (2017). Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 352–361.

[13] Feng, X., Qin, B., and Liu, T. (2018). A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

[14] Freitag, D. (1998). Information extraction from html: Application of a general machine learning approach. In *AAAI/IAAI*, pages 517–523.

[15] Hobbs, J. R., Appelt, D., Tyson, M., Bear, J., and Israel, D. (1992). Sri international: Description of the fastus system used for muc-4. Technical report, SRI INTERNATIONAL MENLO PARK CA.

[16] Hong, Y., Zhang, J., Ma, B., Yao, J., Zhou, G., and Zhu, Q. (2011). Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

[17] Huang, L., Ji, H., Cho, K., and Voss, C. R. (2017). Zero-shot transfer learning for event extraction. *arXiv preprint arXiv:1707.01066*.

[18] Huang, R. and Riloff, E. (2011). Peeling back the layers: detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1137–1147. Association for Computational Linguistics.

[19] Jacobs, G., Lefever, E., and Hoste, V. (2018). Economic event detection in company-specific news text. In *Proceedings of the First Workshop on Economics and Natural Language Processing*, pages 1–10.

[20] Ji, H. and Grishman, R. (2008). Refining event extraction through cross-document inference. In *Proceedings of ACL-08: Hlt*, pages 254–262.

[21] Krupka, G., Jacobs, P. S., Rau, L., and Iwanska, L. (1991). Ge: Description of the nltoolset system as used for muc-3. In *THIRD MESSAGE UNDERSTANDING CONFERENCE (MUC-3): Proceedings of a Conference Held in San Diego, California, May 21-23, 1991*.

[22] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

[23] Lee, C.-S., Chen, Y.-J., and Jian, Z.-W. (2003). Ontology-based fuzzy event extraction agent for chinese e-news summarization. *Expert Systems with Applications*, 25(3):431 – 447.

[24] Li, Q., Ji, H., and Huang, L. (2013). Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82.

[25] Liao, S. and Grishman, R. (2010). Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

[26] Liu, J., Chen, Y., Liu, K., and Zhao, J. (2018a). Event detection via gated multilingual attention mechanism. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[27] Liu, X., Luo, Z., and Huang, H. (2018b). Jointly multiple events extraction via attention-based graph information aggregation. *arXiv preprint arXiv:1809.09078*.

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

[29] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[30] Nguyen, T. H., Cho, K., and Grishman, R. (2016a). Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

[31] Nguyen, T. H., Fu, L., Cho, K., and Grishman, R. (2016b). A two-stage approach for extending event detection to new types via neural networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 158–165.

[32] Nguyen, T. H. and Grishman, R. (2015). Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371.

[33] Nguyen, T. H. and Grishman, R. (2016). Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 886–891.

[34] Nguyen, T. H. and Grishman, R. (2018). Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-second AAAI conference on artificial intelligence*.

[35] Patwardhan, S. and Riloff, E. (2009). A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 151–160. Association for Computational Linguistics.

[36] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

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

[38] Pustejovsky, J., Castano, J. M., Ingria, R., Sauri, R., Gaizauskas, R. J., Setzer, A., Katz, G., and Radev, D. R. (2003). Timeml: Robust specification of event and temporal expressions in text. *New directions in question answering*, 3:28–34.

[39] Riloff, E. (1996a). Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049.

[40] Riloff, E. (1996b). An empirical study of automated dictionary construction for information extraction in three domains. *Artificial intelligence*, 85(1-2):101–134.

[41] Sha, L., Qian, F., Chang, B., and Sui, Z. (2018). Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence.*

[42] Surdeanu, M., Turmo, J., and Ageno, A. (2006). A hybrid approach for the acquisition of information extraction patterns. In *Proceedings of the Workshop on Adaptive Text Extraction and Mining (ATEM 2006).*

[43] Walker, C., Strassel, S., Medero, J., and Maeda, K. (2006). Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

[44] Wu, S., Bondugula, S., Luisier, F., Zhuang, X., and Natarajan, P. (2014). Zero-shot event detection using multi-modal fusion of weakly supervised concepts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2665–2672.

[45] Yangarber, R., Grishman, R., Tapanainen, P., and Huttunen, S. (2000). Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 940–946. Association for Computational Linguistics.

[46] Zhang, T., Ji, H., and Sil, A. (2019). Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 1(2):99–120.

Appendices

Appendix A. Preliminary Studies for the Baseline Model

In this appendix, we describe the preliminary study for the baseline model. Based on this study, we chose the final four variants of the baseline model that were evaluated on the testing data. We conducted this preliminary study for determining the best window size to be used in the baseline model.

For each of the models in the preliminary study, we run five different trials. We used consecutive seeds from 1 to 5 for each model to guarantee replicability. For each model we report the average sensitivity, specificity, and F1-score of the five trials in both the training and validation data. We do not report accuracy because we are in the presence of highly unbalanced data. To thoroughly analyze the F1-score on the validation data, we compute two additional metrics. Firstly, we calculate the confidence intervals (CI) at the 95% level of confidence. Secondly, we compute the p-value for a t-test between each model and the best model reported in each table. We select the best model in terms of the F1-score in the validation data. For example, the p-value of 0.410 in the fourth row of Table A.7 is indicating that there is no enough evidence to reject the null hypothesis that model 4 has a statistically significantly different F1-score from the best model of the table (model 1) in the validation data.

Appendix A.1. Results of the Preliminary Study for the Baseline Model

We run the baseline model with six different window sizes. Although in the original paper, the authors used a fixed window of size 31, we run experiments to determine if this was the best window size for our setting. We do not use the original window size of 31 because our data was different from the data used in the original paper. While they used full news articles, we only used text fragments (sentences). Therefore, our instances were much smaller. Hence, we required smaller window sizes to avoid using too much padding. Given our setting, a window of size 31 would result in a large amount of padding for many tokens. For example, in a sentence of length 31, only the middle word will not have padding while every other word will have. This extra unnecessary padding will add noise and increase the computational cost of the model with little gain in performance. Guided by this intuition, we conducted preliminary studies to explore different window sizes.

In table A.7, we present the result of the models with the six analyzed window sizes. Since the position embedding is used for representing the relative position inside the window, a window of size 1 does not require a position embeddings. Therefore, we excluded the position embedding from this model. All the other features were used in the reported experiments.

Appendix A.2. Discussion

We derive the following conclusions from these preliminary experiments. As we hypothesized, long window sizes were not suitable for our setting. We find evidence to support this intuition in the results from models 5 and 6. For these models, the specificity reached 1.0, and the F1-score dropped to almost 0. For this reason, we do not select this model for the final experiments on the testing dataset.

The model with window size 1 (model 1), achieves the best performance with an F1-score of 59.4% in the validation data. Therefore, we selected this model to use it in the testing data. We also selected model 4 because of its high performance in the validation data. Furthermore, the high p-value provides no evidence to reject the null hypothesis that model 1 has a performance statistically different from this model. Although models 2 and 3 have a decent performance (56.2% and 55.8%, respectively), the statistical analysis shows that we can reject the null hypothesis. Therefore, we have evidence to believe that model 1 is statistically better than models 2 and 3.

In summary, the best model of the table is model 1, with a window of size 1. Model 4, with a window of size 11, has comparable performance, and the analysis showed that there is no statistical difference between them. Therefore, we choose these two window sizes for the experiments on the testing data.

CNNs on the Validation Dataset: Assessing Window Sizes										
Model	Win Size	Features	train			validation				
			$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	F1 CI	p-value
1	1	$\{W,E\}$	0.608	0.980	0.692	0.502	0.970	**0.594**	± 0.019	—
2	3	$\{W,E,Po\}$	0.749	0.986	0.808	0.466	0.975	0.562	± 0.015	0.003
3	5	$\{W,E,Po\}$	0.816	0.988	0.858	0.464	0.975	0.558	± 0.023	0.006
4	11	$\{W,E,Po\}$	0.811	0.979	0.852	0.500	0.965	0.590	± 0.043	0.410
5	21	$\{W,E,Po\}$	0.015	1.000	0.022	0.002	0.999	0.002	± 0.002	0.000
6	31	$\{W,E,Po\}$	0.001	1.000	0.001	0.001	1.000	0.001	± 0.002	0.000

Table A.7: Average performance of the CNN model for six different window sizes. For each window size, we run five trials and report the average sensitivity, specificity, and F1-score for both the training and validation data. For the validation data, we also report the confidence intervals at a 95% level of significance for the F1-score. We also report for each model the p-value of a single-tail t-test against the best model of the table. The test is performed between the F1-score of each of the two models in the validation data. A small p-value gives evidence to reject the hypothesis that the models have the same F1-score, indicating that the best model of the table is statistically better.
In these experiments, we try six different window sizes to determine the best one for using on the testing dataset. The baseline achieved the best performance with window size 1. We found evidence that suggests that the model is statistically better than the ones with window sizes 3, 5, 21 and 31. Although the model with window size 1 performs slightly better than the one with window size 11, we could not find statistical evidence to suggest that one is better than the other. Therefore, the experiment on the testing dataset were carried out with these two window sizes.

Appendix B. Preliminary Studies for the Proposed Model

In this appendix, we describe the preliminary studies carried out to evaluate the proposed model (RNN). Based on these studies, we chose the seven final variant models to use on the testing data. We conducted two preliminary studies. One for determining the number of Bi-LSTM layers and hidden units, and another for studying the features used.

For each of the models in the preliminary study, we run five different trials. We report the average of the five trials for each model. We used consecutive seeds from 1 to 5 for each model to guarantee replicability. We report for each model, the average sensitivity, specificity, and harmonic mean between these two metrics, namely F1-score, for both the training and validation data. We exclude the accuracy of the analysis because we are in the presence of highly unbalanced data. To thoroughly analyze the F1-score on the validation data, we compute two additional metrics. Firstly, we calculate the confidence interval (CI) at the 95% level of confidence for this metric. Secondly, we compute for each model the p-value for a t-test between the model considered and the best model of each table. We define the best model in terms of the F1-score in the validation data. For example, the p-value of 0.160 in the fifth row of Table B.8 is indicating that there is no enough evidence to reject the null hypothesis that model 5 has a statistically different F1-score from the best model of the table (model 6) in the validation data.

Appendix B.1. Results of the First Preliminary Study for the RNN Model

We run the proposed model with eight different architectures (by varying the number of Bi-LSTM layers and hidden units). We configured the number of hidden units to be in descending order, with the first layer having the largest number and the last layer having a smallest number of units. This configuration follows the intuition that each layer should take the input from the previous one and build less but more elaborate features with higher levels of abstraction. For the remaining of this paper, we will describe each architecture as a sorted list of hidden units, where the first number is the number of hidden units of the first BLSM layer, and so on. For example, the architecture $\langle 100{,}15{,}5 \rangle$ is a three-layer architecture, with 100 hidden units in the first layer, 15 in the next one, and 5 in the last one. Except for the Bi-LSTM layers, the remaining of the network is the same for the eight models. The general description of the proposed RNN model is in section 3.

In table B.8, we present the result of the models for the eight different configurations for the Bi-LSTM layers. The first model has three-layers of Bi-LSTM, with 100, 15, and 5 hidden units in the first, second, and third layers, respectively. The second and third models have two layers of Bi-LTSM each, with varying numbers of hidden units. For the second model, we use 15 and 5 hidden units, while for the third, we use 5 and 2. The remaining five modes are all single-layer, with different numbers of hidden units. We present the models from the most complex to the simplest. Models 4 to 8 have 200, 50, 15, 7, and 1 hidden units, respectively. As we previously mentioned, the remaining of the network is the same for the eight models.

Appendix B.2. Discussion

The results achieved by the multiple-layer models 1 and 2 are similar to (69.2% and 68.2%, respectively) and considerably worse than those achieved by model 6 (72.9%), which is a much simpler model. Model 3 and 4 are the worst two models, with a performance of 66.8% and 66.0%, respectively. These results show the importance of finding the right balance between complexity and simplicity. A three-layer model (model 1) achieves good performance, but not as a good as that achieved by the considerably simpler single-layer models (models 5 to 8). However, too simple models, such as model 3, with a small number of hidden units, also perform poorly.

The best model is model 6, which is a single-layer model with 15 hidden units, and hence has a good balance between complexity and simplicity. Furthermore, the four best models are all single-layer models, with 50 hidden units or less (50, 15, 7 and 1). From the results achieved by these models, we observe an inverse relation between complexity and performance. Simpler models achieve better performances. These results confirm the intuition that big and complex models require more data to learn general patterns and avoid overfitting. Therefore, in the context of ED, where the datasets are in the order of hundreds of documents, complex models are prone to harm the performance. Guided by this intuition, which is confirmed by the results, we selected the architecture of model 6 for the second preliminary study and the experiments on the testing set.

RNNs on the Validation Dataset: Assessing Different Architectures										
Model	Architecture	Features	train			validation				
			$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	F1 CI	p-value
1	⟨100,15,5⟩	all	0.722	0.988	0.742	0.654	0.962	0.692	± 0.037	0.026
2	⟨15,5⟩	all	0.732	0.984	0.748	0.648	0.958	0.682	± 0.019	0.002
3	⟨5,2⟩	all	0.723	0.981	0.741	0.628	0.964	0.668	± 0.053	0.015
4	⟨200⟩	all	0.741	0.982	0.758	0.630	0.956	0.660	± 0.043	0.004
5	⟨50⟩	all	0.729	0.982	0.747	0.685	0.950	0.704	± 0.059	0.160
6	⟨15⟩	all	0.757	0.977	0.765	0.709	0.952	**0.729**	± 0.026	—
7	⟨7⟩	all	0.749	0.983	0.763	0.681	0.948	0.698	± 0.039	0.052
8	⟨1⟩	all	0.763	0.977	0.771	0.694	0.945	0.708	± 0.026	0.070

Table B.8: Average performance of the proposed model (RNN) for eight different architectures. The architecture is depicted with a list that represents the number of hidden units of each layer. The architecture ⟨15,5⟩ is a network with two Bi-LSTM layers with 15 and 5 hidden units in the first and second layers, respectively. The layers before and after the Bi-LSTM layers do not vary and are as described in section 3. For each architecture, we run five trials and report the average sensitivity, specificity, and the harmonic mean between these two metrics, namely F1-score, on both the training and validation data. For the validation data, we also report the confidence interval at a 95% level of significance for the F1-score. We also report the p-value of a single-tail t-test of each model against the best model of the table. The test is performed between the F1-score for each of the two models on the validation data. A small p-value gives evidence to reject the hypothesis that the models have the same F1-score, indicating that the best model of the table is statistically better.
In these experiments, we tried eight different number of architectures (number of Bi-LSMT layers and hidden units) for our proposed model. Model 6, the single-layer model with 15 hidden units, is the one with the best performance. The four models with the best results are single-layer, indicating that in this context of small datasets, simpler models perform better. Models that are too complex, such as model 4, with 200 hidden units, and models that are too simple, such as model 3, with only two small Bi-LSTM layers, perform poorly. The results provide evidence to believe model 6 achieves the best balance between complexity and simplicity. Therefore, we use the architecture of model 6 for the subsequent experiments.

Appendix B.3. Results of the Second Preliminary Study for the RNN Model

We performed a second preliminary study by varying the features used in the model to study the impact of each feature on the overall performance. We conducted these experiments in a similar way to an ablation study. We first measure the performance of the model with all the features, and for each model or hypothesis we wanted to test, we run a new model with different features removed and compared its performance with that of the model that maintained all the features.

The goal of this preliminary experiment was to assess the impact of each feature on the overall performance. This allows us to remove those features that are not useful for the task or that add noise and therefore harm the performance. We performed five trials for each set of features using the best architecture found in the previous preliminary study. The results of these experiments are in Table B.9, where the first row reports the performance of the model with all the features, and the following are the same model with one or more features removed.

We tested nine different feature configurations, including model 1, which contained all the features (all). To assess the impact of contextual embeddings we evaluated three different models (models 2 to 4). Model 2 contains all the features except for the contextual embeddings (all-$\{B\}$). Model 3 includes all but the contextual sentence embeddings (all-$\{S\}$), which are features constructed by adding up the contextual embeddings (B) for all the words in the sentence. Finally, model 4 has both the contextual word and contextual sentence embeddings removed (all-$\{B,S\}$).

We also evaluated two different models to measure the impact of the use of grammatical information. First, we evaluated model 5, where the Part-Of-Speech tags are removed, both the simplified version and the detailed ones (all-$\{P,T\}$). Second, we evaluated model 6, where the dependency tags were removed (all-$\{T\}$).

Finally, we evaluated three additional models to test the impact of the three remaining features: the entity embeddings (E), the Word2Vec embeddings (W), and Spacy contextual word embeddings (Sp). These are models 7, 8, and 9, respectively.

Appendix B.4. Discussion

We observe that the performance drops considerably (a drop of 15.1% in F1-score) for the model that does not contain the contextual word embeddings (model 2) in comparison to the model that contains all the features (model 1). Similarly, the model without the contextual sentence embeddings (model 3) has a drop in performance of 10.3%. On the other hand, the performance of the model without both features (model 4) has a performance similar to that of the model where only the contextual sentence embeddings are removed (model 3). A statistical analysis shows that for the three models, the hypothesis that they are equal to the best model can be rejected with a confidence level of over 95%. The two individual analyses of the contextual embeddings show the significant

RNNs on the Validation Dataset: Assessing Feature Contribution										
Models	Architecture	Features	train			validation				
			$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	F1 CI	p-value
1	$\langle 15 \rangle$	all	0.757	0.977	0.765	0.709	0.952	0.729	± 0.026	0.326
2	$\langle 15 \rangle$	all-$\{B\}$	0.707	0.980	0.728	0.528	0.963	0.578	± 0.072	0.001
3	$\langle 15 \rangle$	all-$\{S\}$	0.671	0.995	0.710	0.571	0.971	0.626	± 0.061	0.003
4	$\langle 15 \rangle$	all-$\{B,S\}$	0.781	0.997	0.790	0.594	0.935	0.637	± 0.023	0.000
5	$\langle 15 \rangle$	all-$\{T,P\}$	0.752	0.976	0.761	0.667	0.957	0.698	± 0.037	0.026
6	$\langle 15 \rangle$	all-$\{D\}$	0.747	0.980	0.758	0.718	0.944	0.733	± 0.017	0.430
7	$\langle 15 \rangle$	all-$\{E\}$	0.761	0.977	0.769	0.697	0.950	0.721	± 0.034	0.182
8	$\langle 15 \rangle$	all-$\{W\}$	0.629	0.973	0.644	0.707	0.954	0.727	± 0.049	0.341
9	$\langle 15 \rangle$	all-$\{Sp\}$	0.748	0.979	0.759	0.721	0.946	**0.735**	± 0.018	—

Table B.9: Average performance of the proposed model (RNN) for nine different sets of features using the best architecture (single Bi-LSTM layer with fifteen hidden units). For each set of features, we run five trials and report the average sensitivity, specificity, and F1-score in both the training and validation data. For the validation data, we also report the confidence intervals at a 95% level of significance for the F1-score. We also report for each model the p-value of a single-tail t-test against the best model of the table. The test is performed between the F1-score in the validation data for each of the two models. A small p-value gives evidence to reject the hypothesis that the models have the same F1-score, indicating that the best model of the table is statistically significantly better.

In these experiments, we examine nine different sets of features. We use the complete set of features for comparison, and remove each different input to measure its impact. We removed T and P together because they are semantically the same input (i.e., the simplified and the detailed versions of Part-Of-Speech). And we remove B and S inputs together because those are the two contextual-embedding-related inputs. The contextual-embedding-related features show the most significant impact. The results show that features D, E, W, and Sp have a negligible impact on the performance. We can conclude from these results that, in the presence of all the other features, we can remove each of these features without harming the performance. On the other hand, the T and P features have a small but statistically significant effect when removed. We conducted further experiments to find additional evidence for these findings.

impact of each in the overall performance and indicate a slightly higher impact of the contextual word embeddings over the contextual sentence embeddings.

The results for the model without the dependency information (D) (model 6) are similar to those obtained by the model that preserves all the features (model 1). These results provide evidence that suggests that the model could not take advantage of the information of this input. Three factors can be influencing these results, which are discussed in section 5.1. By removing the Part-Of-Speech information (T, P) (model 5), we have a small drop in performance (3.1%). This result suggests that the feature can be useful for the ED task. We further explore the inclusion and exclusion of all this grammatical information (T, P, D) in the testing set.

Rows 7 to 9 of table B.9, corresponding to models 7 to 9, show the models with all the features excluding the entity embeddings (E), the Word2vec word embedding (W), and the Spacy contextual embeddings (Sp), respectively. Model 9 is the one with the best performance. However, the high p-values show that there is no statistical difference between the best model and the other two. Furthermore, there is no statistical difference between the best model and the model with all the features (model 1). These results show that removing any of these three features (E, W, Sp) has a negligible impact on the overall performance. We further study the effect of including and excluding these features in the testing set. However, based on these results, it is expected that, in the presence of the other attributes, excluding these three features improves performance.