Mariano Maisonnave, Fernando Delbianco, Fernando Tohmé, Ana Maguitman, Evangelos Milios,
Detecting ongoing events using contextual word and sentence embeddings,
Expert Systems with Applications,
Volume 209,
2022,
118257,
ISSN 0957-4174,
https://doi.org/10.1016/j.eswa.2022.118257.
(https://www.sciencedirect.com/science/article/pii/S0957417422013975)

Detecting Ongoing Events Using Contextual Word and Sentence Embeddings

Mariano Maisonnave
Universidad Nacional del Sur
Argentina
mariano.maisonnave@cs.uns.edu.ar

Fernando Delbianco
Universidad Nacional del Sur
Argentina
fernando.delbianco@uns.edu.ar

Fernando Tohmé
Universidad Nacional del Sur
Argentina
ftohme@criba.edu.ar

Ana Maguitman
Universidad Nacional del Sur
Argentina
agm@cs.uns.edu.ar

Evangelos Milios
Dalhousie University
Canada
eem@cs.dal.ca

Abstract

This paper introduces the Ongoing Event Detection (OED) task, which is a specific Event Detection task where the goal is to detect ongoing event mentions only, as opposed to historical, future, hypothetical, or other forms of events that are neither fresh nor current. Any application that needs to extract structured information about ongoing events from unstructured texts can take advantage of an OED system. The main contribution of this paper are the following: (1) it introduces the OED task along with a dataset manually labeled for the task; (2) it presents the design and implementation of an RNN model for the task that 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 detecting ongoing events in news; (3) it presents 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 OED task, outperforming the baseline models.
Keywords: *Ongoing Event Detection; Information Extraction; Contextual Embeddings; BERT; RNN; CNN.*

1 Introduction

The Information Extraction (IE) task consists in extracting structured information from unstructured natural language texts (Grishman, 1997). 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 usually performs two different steps to complete the extraction of the events (Ahn, 2006). The first step is to identify the event trigger (or anchor), which is the word that most clearly expresses the occurrence of an event, and to 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 refer to participants (e.g. people) or attributes (e.g. time and place) associated with the event. In this work, we focus only on a variation of the event detection task and not on the argument extraction task.

The ED task has been addressed in the literature, both as a standalone task (Duan et al., 2017; Jagannatha and Yu, 2016; Liu et al., 2018a; Nguyen et al., 2016b; Nguyen and Grishman, 2015,1,1; Wu et al., 2014) and as a part of an EE system (Ahn, 2006; Chen et al., 2015; Huang et al., 2017; Li et al., 2013; Liu et al., 2018b; Nguyen et al., 2016a; Sha et al., 2018; Zhang et al., 2019). There is a great incentive to study ED systems not only because of their direct use in several applications but also because any improvement in an ED system will impact directly on the performance of any EE system implemented with it. 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 (Sha et al., 2018) and text summarization systems (Lee et al., 2003). These systems are also useful for generating reports of the information available for a domain (Adedoyin-Olowe et al., 2016; Chen et al., 2018; Liu et al., 2009). These reports can help an expert to make decisions or create policies to address an issue.

In this paper, we define and address the Ongoing Event Detection (OED) task as part of a broader project that aims at detecting ongoing real-world events and other relevant variables from news and social media with the

ultimate goal of learning causal models (Maisonnave et al., 2020). In the broader project, we use time series-related techniques from Econometrics to learn such causal models (Granger, 1969). To build those time series, we require (i) all the event mentions in the news and (ii) the time those events took place. These two requirements defined the needs of our event detection task. Because of (i), we require only the detection of event mentions (trigger detection), not the event arguments nor the participants. Because of (ii), we require the event to be ongoing when reported (discarding past, future and hypothetical events). If the event is ongoing at the moment it is being reported, the news article's date can be used as the moment of occurrence of the event. Because we consider an event as ongoing if it was in progress at the time it was reported, an old news article could refer to an ongoing event (if it was happening when the news article was released). Using this definition and the publication date, we can keep track of the beginning and ending of events in time, and we can try to find causal links between them (which is part of the broader project of which we made a pilot study in (Maisonnave et al., 2020)). These two requirements result in the definition of a proposed novel task, to which we refer to as the Ongoing Event Detection (OED) task.

We divided the broader project into two steps, (1) the OED task and (2) the time series construction and causal structure learning from those time series. The two steps involved in building the tool are depicted in Figure 1. The work reported in this paper focuses on the first step only, i.e., on the OED task.

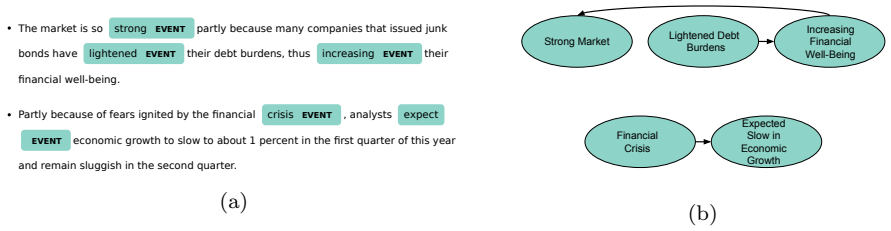


Figure 1: Two use cases for the OED tool, applied to two sentences (a). A causal graph manually extracted from the examples (b).

Motivated by the limitations of previous proposals to address our specific requirements, we outlined a definition of event (described in Section 3) that, on the one hand, is not limited by a fixed taxonomy of events and, on the other hand, is centered on ongoing events only. Guided by this definition, we manually labeled a dataset for training and testing purposes (described in Section 4.1). Using this dataset, we developed an RNN model for event prediction (described in Section 4.2). Since there are no previous studies on this dataset, we also implemented two baselines for comparison. First, we implemented a simple OED model based on a classical approach (SVM model). Second, we replicated a baseline (CNN model) from the state-of-the-art in ED to apply to our task (Nguyen and Grishman, 2015). Both baselines are described in Section 4.3. 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]. Results, discussion and conclusions are presented at the end of the paper in Sections 5.2, 5.3, 6, respectively.

The contributions of this paper can be summarized as follows.

1. Firstly, we define the OED task for the first time, and we present a manually labeled dataset for the task. An active learning tool was developed and used for the construction of the dataset for the task. According to the annotators, this tool (which was fine-tuned as new labels became available) made the annotation process easier, therefore shortening the labeling time. The proposed task proved to be a promising direction for implementing the first step of the framework for causal structure learning depicted in Figure 1.

2. Secondly, we design and implement an RNN model for the OED task that includes BERT embeddings (Devlin et al., 2018) as features. The use of BERT embeddings for the task is extensively studied in this work, and the results demonstrate the usefulness of context-sensitive embeddings for this task. We also elaborate on a baseline by repli-

[1]The code is available at `https://cs.uns.edu.ar/~mmaisonnave/resources/ED_code/`, and the dataset is available at `https://cs.uns.edu.ar/~mmaisonnave/resources/ED_data/`

cating a state-of-the-art model (Nguyen and Grishman, 2015) on our data, as well as a classical SVM model. The proposed RNN model outperforms the evaluated baselines.

3. Finally, we present an extensive empirical evaluation with different architectures and hyperparameters for both the baseline and our model. We also perform an ablation test to measure the impact of each feature on the performance. The full code of the experiments is made available to allow reproducibility.

2 Background and Related Work

The most widely used dataset for EE is the ACE 2005 Multilingual Training Corpus (Walker et al., 2006), which contains the complete set of English, Arabic, and Chinese training data for the 2005 Automatic Content Extraction (ACE) technology evaluation (Doddington et al., 2004). There are other EE and ED datasets available, such as TimeML (Pustejovsky et al., 2003) and SentiFM (Jacobs et al., 2018). However, to the best of our knowledge, all these datasets have a fixed taxonomy of valid event types.

Three main disadvantages arise when using existing EE datasets from the literature. First, as already mentioned, these datasets limit the number of events to a fixed set of possible events (those available in the taxonomy). This problem arises because datasets for the EE task require specific details about arguments for each new event type added to the taxonomy. However, for the ED task in general, and the OED task in particular, there is no need to consider arguments since the goal is to detect event triggers only. Second, when using an existing dataset developed for a particular new task or domain, the event taxonomy and the source of the news articles usually do not suit precisely the needs for the new task or domain. Third, existing datasets do not distinguish ongoing events (which are the focus of our work) from historical, future, hypothetical, or other forms of events. Because of these three disadvantages, we rely on creating our own dataset for our specific task.

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 (Chen et al., 2015; Feng et al., 2018; Huang and Ji, 2020; Lai et al., 2021; Li et al., 2020; Nguyen et al., 2016a; Nguyen and Grishman, 2015,1; Tong et al., 2020b; Yao et al., 2021) (or one binary classification for each possible event type (Nguyen et al., 2016b)). Traditionally, these classifiers use a wide variety of features representing lexical, syntactic or entity information, which are typically the result of applying NLP tools (e.g. (Li et al., 2013)). With the advent of neural network language models and the availability of distributed 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 input features used in state-of-the-art approaches are unsupervised representations automatically learned from big corpora (e.g., Word2vec (Mikolov et al., 2013b) and Fasttext (Bojanowski et al., 2017)), and many of these representations can keep improving during training through gradient descent.

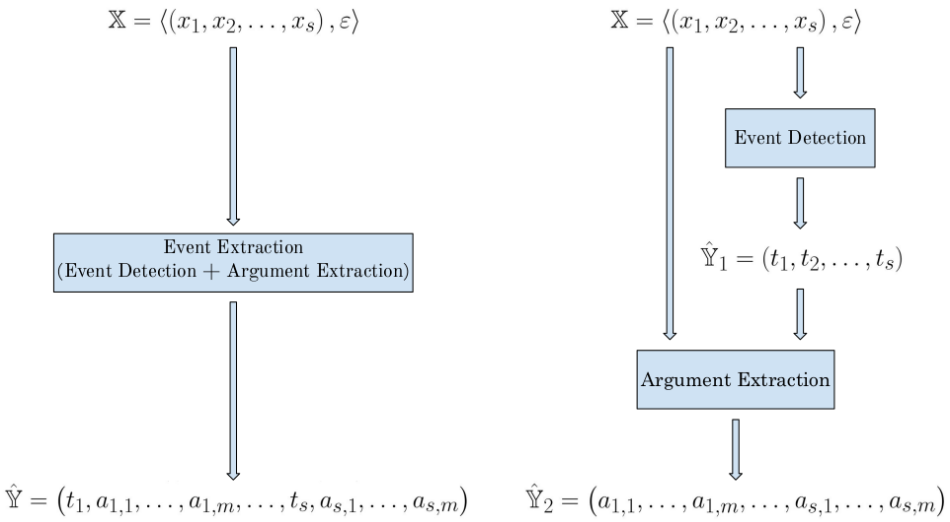
According to (Boroş, 2018), ED approaches fall into one of three categories: pattern-based (Hobbs et al., 1992; Krupka et al., 1991; Riloff, 1996a,9; Yangarber et al., 2000), feature-based (Bronstein et al., 2015; Chieu et al., 2003; Freitag, 1998; Hong et al., 2011; Huang and Riloff, 2011; Ji and Grishman, 2008; Li et al., 2013; Liao and Grishman, 2010; Patwardhan and Riloff, 2009; Surdeanu et al., 2006), and neural-based (Chen et al., 2015; Feng et al., 2018; Nguyen et al., 2016a,1; Nguyen and Grishman, 2015). The first two rely on sophisticated handcrafted rules, patterns, and features, as well as NLP tools. The first two categories present several disadvantages. First, creating the handcrafted rules is very time-consuming and will likely result in high precision and low recall (since it is unlikely to cover all the possible patterns manually). Second, selecting a good set of features is not a trivial task, and building such features relies on sophisticated NLP tools that have a certain level of error associated. The third category relies on neural networks for both feature representation and token prediction (which eliminates the need for handcrafted rules, NLP tools, and – in many cases – feature selection). 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.

EE systems typically follow one of two possible architectures: pipelined (Chen et al., 2015; Hong et al., 2011; Ji and Grishman, 2008; Liao and Grishman, 2010) or joint (Liu et al., 2018b; Sha et al., 2018; Zhang et al., 2019). 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, the argument extraction is part of the event detection (trigger detection) phase and vice versa. A description of this two architectures is provided in Figure 2.

The notion of context information is not new for the ED task. In (Feng et al., 2018), the authors use an RNN model to capture a simple notion of context for each word to improve the performance on the task while using classical embeddings. In (Fan et al., 2020), the authors show how BERT embeddings improve the performance on the adverse drug event (ADE) detection task in the medical domain. In (Tong et al., 2020a) BERT-based text features are integrated with image features to address the ED task in news. Even though these approaches are either applied to domains different from news or incorporate non-textual information, they provide evidence supporting the usefulness of exploiting contextual word representations for ED.

On the other hand, the notion of ongoing events as defined in this paper – to the best of our knowledge – is introduced here for the first time. In (Huang et al., 2016), the authors use document classification to categorize news articles containing events as past, ongoing, future planned, future alert, and future possible. Although they point to the importance of distinguishing ongoing from past and future events, their approach is different from ours. They work at the document level, classifying full news articles, while we work at the level of event triggers. Furthermore, they classify events into different categories (past, ongoing, future), while we only determine if the event is ongoing or not. Additionally, the dataset they introduce, EventStatus corpus, is defined for a different context than ours. Their dataset consists



(a) joint EE architecture

(b) pipelined EE architecture

Figure 2: Depiction of the joint (a) and pipeline (b) architectures used for the EE task as defined in ACE (Walker et al., 2006), notation adapted from (Li et al., 2013). The input, in both cases, is a sentence separated in tokens $(x_1, x_2, \ldots, x_3)$ and a set of m candidate arguments $\varepsilon = \{e_k\}_{k=1}^{m}$ (which can be a gold-standard or predicted with a Named Entity Tagger). In the pipelined architecture, first, the input is used to predict the event trigger type (event detection), the output of this stage is represented with $\hat{\mathbb{Y}}_1 = (t_1, t_2, \ldots, t_s)$. Here each $t_i \in \mathcal{L} \cup \{\perp\}$, where $\mathcal{L}$ is the set of all possible event types and $\perp$ is used for identifying non-events. Secondly, a argument extraction stage is performed using $\mathbb{X}$ and $\hat{\mathbb{Y}}_1$ as input and producing $\hat{\mathbb{Y}}_2 = (a_{1,1}, \ldots, a_{1,m}, \ldots, a_{s,1}, \ldots, a_{s,m})$ as output. Here $a_{i,j} \in \mathcal{R} \cup \perp$, where $\mathcal{R}$ is the set of possible argument roles and $\perp$ denotes that the current candidate argument is not an argument for the current trigger. Therefore $a_{i,j}$ represents the role of the j-th argument candidate for the i-th token. The joint architecture performs both steps at the same time (Event Detection + Argument Extraction), resulting in the $\hat{\mathbb{Y}}$ output which consists exactly of the $\hat{\mathbb{Y}}_1$ and $\hat{\mathbb{Y}}_2$ information combined. In the pipelined architecture the tasks are addressed independently and the error propagates downstream. In the joint architecture a technique is proposed for both tasks at the same time and usually takes advantage of the information of one task to complete the other.

of civil unrest events, such as protests, demonstrations and strikes.

3 The Ongoing Event Detection Task

The OED task is a specific ED task whose goal is to detect ongoing event mentions only, as opposed to historical, future, hypothetical, or other forms of events that are neither fresh nor current. Current states of affairs re-

ported in the news are also considered ongoing events. Since we need to establish the date on which each real-world event took place, we use the best approximation available, the publication date. For example, if a crisis took place in 1994 and we can detect that an article from 1994 mentions that crisis as current, we can use the publication date as the date on which the event happened. On the other hand, if an article published years later describes that crisis as a past one, we can ignore that mention (because it is not ongoing). Using each real-world event and the time at which it occurred, we can build a time series dataset to perform causal discovery (which is the broader goal of our work (Maisonnave et al., 2020)).

In this section, we present all the definitions required for the OED task, and we present several key examples to facilitate the understanding of the task. All these definitions, the examples for the task, and the annotators' guidelines are available on the dataset website.[2]

Definition 3.1 (Ongoing Event)**.** An ongoing event is any text fragment in a news article reporting a real-world event that meets any of the following conditions: (i) it is a fresh event, (ii) it happened a time ago and is still ongoing, or (iii) it is the current state of affairs for a given entity.

An example of (i) is when an earthquake just took place, and a news article fragment covers that event. An example of (ii) is when a riot started in a city some days ago and is again reported in the news while it is still happening (because there is some new information or is a recapitulation of what happened). Lastly, an example of (iii) is when a news article reports a crisis or a recession that is taking place in some country or region. It is important to notice that the same fragment could contain more than one ongoing event. For example, some news article fragments could be reporting an event caused by another. For instance, an ongoing crisis (an example of (iii)) could cause a riot (an example of (ii)).

Ongoing Event Trigger and Ongoing Event Detection Task are defined analogously to Event Trigger and Event Detection Task (Walker et al., 2006), respectively, as follows:

[2] `https://cs.uns.edu.ar/~mmaisonnave/resources/ED_data/`.

Definition 3.2 (Ongoing Event Trigger)**.** An Ongoing Event Trigger is the word (verb, noun, or any other part of speech) that most clearly states the occurrence of an ongoing event (Definition 3.1).

Definition 3.3 (Ongoing Event Detection Task)**.** The Ongoing Event Detection (OED) Task is the task of detecting the ongoing event trigger (Definition 3.2).

In the OED task, the context of a word is crucial to determine if it refers to an ongoing event or not. For example, take the word "crisis" in the following sentences:

1. The current *crisis* will accelerate digital technology.
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 we must mitigate the risks of a future *crisis*.

Only the reference to a "crisis" in sentence 1 is considered an event trigger, while the other mentions of the same word are not. This is guided by the need to distinguish ongoing events from those that are not, which is a requirement of the ultimate goal of our broader project of detecting causal relations between events reported in the news. Most existing approaches do not adopt this context-sensitive definition of event. Furthermore, our definition of ongoing event also accounts for current states of affairs, which are not taken into consideration by existing proposals. To address this context-sensitive definition of event, we include BERT-based features. The context-sensitive nature of the BERT model allows finding different representations of the same word (for example, the word "crisis") according to the context. We believe these different representations will allow the model to correctly distinguish ongoing events from non-events, as well as from hypothetical, future or past events.

As another example, consider 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 ongoing event trigger in this example is "deficit" because it is the only ongoing event referred to in the news. The word "devaluation" is not an ongoing event trigger as a devaluation may not take place. Similarly, the word "weakening" is not an ongoing event trigger as it is a hypothetical event. Finally, the word "crisis" is not considered an ongoing event trigger as the news refers to a crisis from the past. Note that the words "devaluation", "weakening" and "crisis" could be labeled as ongoing event triggers in other news extracts, where the context of use of these words is different, but not in the given example.

As stated in Definition 3.3 and illustrated with the provided examples, the OED task can be addressed using a binary classification model, which predicts for each token in a text fragment if it is an ongoing event trigger or not. We adopt this approach and propose a binary classification model for the task along with several baselines for comparison. The proposed model is introduced in Section 4.2, and a high-level depiction of the proposal is presented in Figure 5.

We provide additional details on the OED dataset building process and the active learning tool in the next section (Section 4.1). A document with the annotation guidelines is available at the dataset website (`https://cs.uns.edu.ar/~mmaisonnave/resources/ED_data/`). In that document a brief introduction to the topic is given, and the OED task is presented (as presented in this section), and the active learning tool if fully described to the annotators.

4 Experimental Setup

4.1 Dataset

To build our dataset we tokenized the full New York Times (NYT) archive (1987-2007) (Sandhaus, 2008) using the Spacy NLP library and divided the news into sentences. From the full set of sentences extracted from the corpus (∼64 million), we selected a subset for labeling. We chose 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 selected financial episodes because, in the broader project (Maisonnave et al., 2020), we work with economic experts who have an interest in applying these tools to the economic domain. 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", and "devaluation". From the obtained results, we randomly selected 2,000 sentences. Also, we randomly selected from these results a separate set of 200 sentences for testing purposes.

We chose to use sentences instead of full texts to favor diversity in our dataset. Because the annotation task is labor-intensive, by labeling sentences instead of full texts we covered different events from different news articles rather than repeated mentions of the same event discussed in various parts of the same article. We handle each sentence as an independent instance (independently of the article it came from). Therefore, the Lucene search engine could return zero, one, or more sentences from each given article. However, given that there are approximately 64 million sentences in 2 million full-length articles, and we only selected 2,200 sentences in total, it is likely that many of the sentences are drawn from different articles. We believe the decision of using sentences instead of full texts does not simplify the task. Instead, it changed the task to a different one, with different advantages and disadvantages. In general terms, and for the text classification

[3] `https://lucene.apache.org/`

task in particular (Cekik and Uysal, 2020), dealing with short-text presents several problems like incomplete, inconsistent and sparse data, together with high dimensionality. On the other hand, there are several advantages of using short texts. For example, since usually short-texts require less effort to write, we can expect short-text datasets to have a higher number of instances. For instance, while the New York Times corpus has roughly 2 million full-length articles, on Twitter, 500 million tweets are generated in a single day (İnanç Arın et al., 2018). Another advantage of the short-text context is that simpler models can be used without compromising performance. Usually, this is not the case for full-length articles when the model needs to keep track of long dependencies between words in the text. In this context, we have short texts (sentences), which means we could use simpler models. However, we have the problem that context was missing for some instances (because important information was not part of the sentence being labeled). In that sense, this task is similar to short-text classification.

Since in the proposed OED task we are not limited to a fixed set of events, any ongoing real-world event or situation reported in the news articles was considered an event. Consequently, each word in our dataset was labeled as *event trigger* or *non-event trigger*, transforming the OED task into a binary classification task. As mentioned earlier, 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 describe real-world events (like crisis, recession, earthquake, flood, etc.) are labeled as *non-event triggers* if they do not refer to ongoing events at the time the analyzed news is released. This is illustrated by the example presented in Section 3 with different mentions of the word "crisis".

We developed a simple active learning tool to assist the labeling process of the training and validation data (but not the test data). In Figure 3, we present a screenshot of the active learning tool used by the annotators. This tool used an early prototype of the RNN model used for event prediction

(described in Section 4.2) to suggest labels. Each sentence was presented to a group of four annotators, along with the corresponding suggestions generated by the model. Then the four annotators had to agree on keeping the suggested labels, removing some of them, or adding new ones. The whole process took a total of fifteen sessions of approximately two hours each. Since the labeling process was carried out following a consensus-based approach, no inner-agreement could be calculated. Therefore, such a metric is not reported. Because we initially do not have any labeled instances for training, we had a *cold start*. As new instances were labeled, the model was re-trained. The re-training process took place each time 50 news instanced were labeled, and each re-training process was performed using the whole set of instances labeled at that moment (including the newly labeled 50 instances). It is important to notice that to avoid biasing the users' decisions when tagging events in the test set, 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. Because the annotation process is so labor-intensive, the size of the test set is relatively small. However, for the results to be reliable, statistical tests were used on the results reported in Section 5 to guarantee that the hypotheses tested in this work are valid.

The statistics of the dataset are presented in Tables 1 and 2. Additional details on how the dataset was built can be found on the dataset website (`https://cs.uns.edu.ar/~mmaisonnave/resources/ED_data/`).

4.2 RNN Model

We designed and trained a Recurrent Neural Network (RNN) for the OED task. 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 ongoing events in natural language text, we hypothesize that syntactic, semantic, and grammatical information is needed. To represent the semantics of each token we use Word2vec (Mikolov et al., 2013a) (W). Note that we use Word2vec instead of other word em-

Figure 3: Overview of the active learning tool running for the first time. There are 2,000 sentences to be labeled, and 0 labeled. The system sorts the unlabeled list and presents it to the annotators. Because the system has not been trained yet (as there is no label information yet), the system's suggestions have a performance no better than a random generator.

beddings (e.g., FastText (Bojanowski et al., 2017)) for the sake of comparison, as Word2vec is used in our baseline (Nguyen and Grishman, 2015). Word2vec embeddings represent each word by means of a 300-dimension vector that is fine-tuned as the model is trained for OED. We use the pre-

Full Dataset (training/validation + test)	
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 1: Statistics about the OED dataset vocabulary.

trained Word2vec[4] word embeddings to initialize the weights of a Keras embedding layer. Fine-tuning is performed to update those weights as we train the model for the OED task. We also tested 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[5]. This feature was not fine-tuned during training.

To encode the syntactic information, we first used the Spacy library to identify the dependency tree (*D*). For each token, we extracted 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 the representation was incrementally fine-tuned along with the training of the rest of the network. The grammatical information was encoded using two different embeddings

[4] `https://code.google.com/archive/p/word2vec/`
[5] `https://spacy.io/`

Metric	Training/Validation		Test	
	Total	Avg. per Sent	Total	Avg. per Sent
Token Count	76,629	38.31	7,382	36.91
Word Count	67,032	33.52	6,442	32.21
Entity Count	11,502	5.75	950	4.75
Event Count	5,119	2.56	416	2.08

Table 2: Total number of tokens, words, entities, and events found in the dataset.

representing information retrieved through Spacy's Part-Of-Speech (POS) tagger. We used both the simplified POS tag version (P) and the detailed one (T) of the Spacy NLP library. 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 and fine-tuned the representation during training.

We used the Spacy Named Entity Recognizer tagger to include entity information for the OED task in the form of a separate feature (E). We represented each word as a two-part tag. The first part is the IOB notation and the second is the type of the entity. 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 initialized with random weights and is fine-tuned during training.

Contextual embeddings, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018), were proposed to solve the problem of mixed semantics when the same word has a context-sensitive meaning. These embeddings achieve ground-breaking performance on a wide range of NLP tasks. We hypothesize that having vector sensitivity to the context is crucial for the OED task, and it will bring a boost in performance. To illustrate this hypothesis, consider the following two sentences: "The firm had to fire employees" and "A 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, which can limit the performance of an ED model for identifying these cases. While Word2vec has the same representation for the two mentions of the word "fire" in these two sentences, the BERT model will produce different vectors for the two different uses of the word "fire" (because it uses the context to build the representation). Motivated by the intuition that the context is crucial for detecting ongoing events, we incorporated pre-trained contextual word embeddings, in particular BERT embeddings (B), as an input to the model. We expect BERT embeddings to have a favorable impact on the performance for instances where the same word could be or not be an ongoing event depending on context (like the example of the word "crisis" in Section 3).

Another problematic situation occurs with a word that preserves its semantics across different sentences but that can refer to an ongoing 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. Guided by this intuition, we incorporate a contextual sentence embedding (S) built by adding up the BERT embeddings for each token in the sentence. The B and S inputs are not modeled with a Keras layer and therefore they are fine-tuned during training for the OED task. A summary of the eight above-described features used for the RNN models is presented in Table 3. Since we explore different subsets of features, not all the features are present in all the RNN models. The subset of features used in each model is concatenated in one vector. For example, for the model with all the features, the resulting concatenation is a vector of 1962 dimensions.

We implicitly use transfer learning in the various inputs of the model. Firstly, by using the pre-trained Word2vec, Spacy and BERT embeddings, we are incorporating semantic information extracted from large corpora of unlabeled text. Secondly, by using the POS tagger and the Dependency tagger from the Spacy library, we are including grammar and syntactic information coming from other IE fields, where there are large labeled corpora.

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 eight embeddings, are all concatenated to form a

Abbr.	Size	Description
W	300	Pre-trained word embeddings.
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.
Sp	96	Spacy contextual word embeddings.
B	768	Pre-trained contextual word embeddings.
S	768	Contextual sentence embeddings.

Table 3: **Features** used by the RNN models. The subset of features used for each model is concatenated into one embedding. For example, if a model uses all the features, it will have a 1962-dimension input vector consisting of all the embeddings concatenated.

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 15 hidden units. A final dense layer with only one hidden unit was added to the Bi-LSTM Layer. The output of this final layer is the prediction. The RNN architecture is outlined in Figure 4. Although this architecture is the one used during the experiments on the data held-out for testing, we tested other architectures (different numbers of Bi-LSTM layers and hidden units). The results of these additional experiments with various architectures can be found in the supplementary materials.

Hyperparameters. For the embedding layers we use the default configuration, only changing the random initialization for the Word2vec embeddings. We use $p = 0.1$ for the Dropout layer and 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 15 hidden units, which was the architecture with the best performance during the preliminary studies reported in the supplementary materials. Finally, we set up the dense layer using the sigmoid activation function.

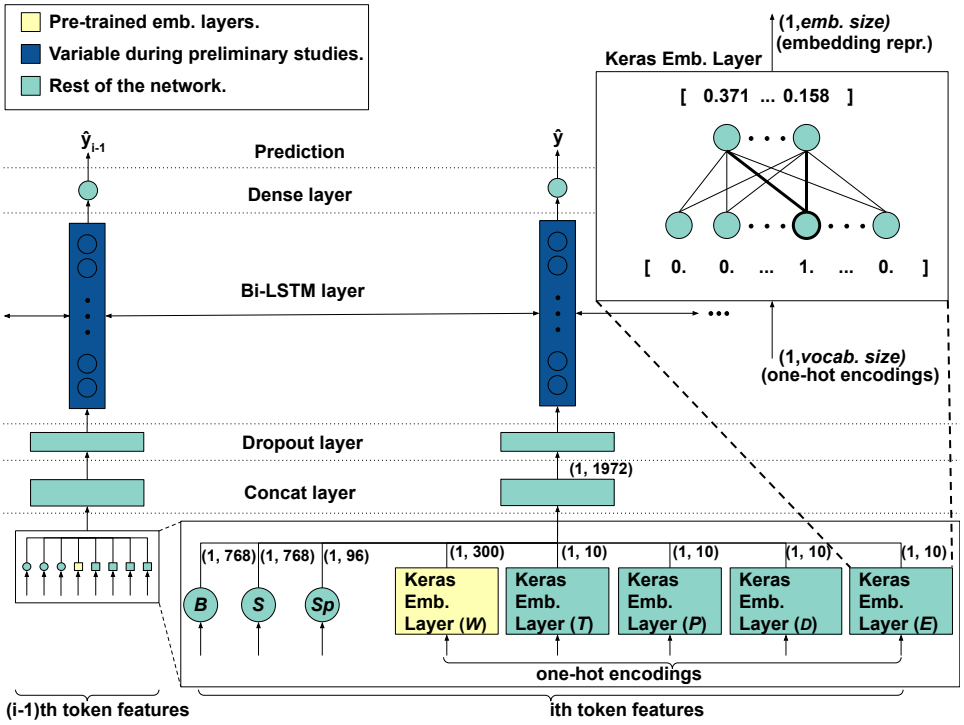


Figure 4: **Outline of the proposed RNN architecture**. From bottom to top. A representation of the eight inputs, for the ith and the $(i-1)$th token. For each token, three of the inputs are vector representations (B, S, Sp); the other five are one-hot encodings that enter five Keras embedding layers. Each of the dense layers is depicted in the top-right corner diagram. One of these embedding layers starts 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.

4.3 Baseline Models

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, we replicated an ED model from the state-of-the-art to use as a baseline, as well as a classical SVM model. Since the focus of our work is ED (and more specifically OED), we evaluated our model against state-

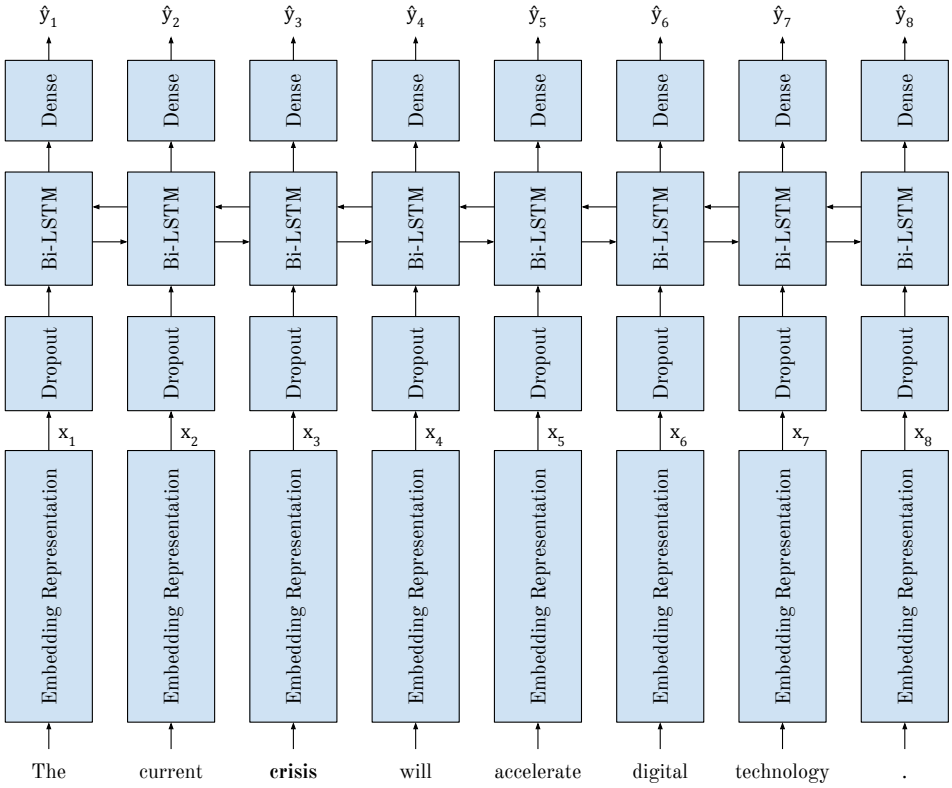


Figure 5: High-level description of the proposed methodology to solve the OED task. At the bottom of the Figure, the complete sentence is fed as input to an embedding representation layer. This first layer transforms each token into a 1962-dimension vector representation (x_i). Each of these 1962-dimension vectors is built concatenating eight embedding representations from different sources (Word2vec, BERT, and embeddings built using NLP tools (NER, dependency parser)). See Table 3 for a description of the eight embeddings used. A dropout layer is applied to x_i for regularization, followed by a layer of Bi-LSTM cells. Finally, a single-hidden-unit dense layer produces the prediction (y_i). In this example sentence, we expect to have the word "crisis" labeled as an event trigger ($y_3 = 1$) and all the other words predicted as non-event-trigger ($y_i = 0$ for $i \neq 3$).

of-the-art in ED, instead of considering approaches that perform ED+EE. Based on these considerations and because of its simplicity and comparable performance with the state-of-the-art in ED, we selected and replicated the Convolutional Neural Network (CNN) model proposed in (Nguyen and Grishman, 2015) as a baseline for comparison (in addition to the SVM baseline model). Despite being proposed several years ago, this model is still compet-

itive with the available state-of-the-art methods. For example, it presents an F1-score of only 4.1 percentage points lower than more recent models, such as (Nguyen and Grishman, 2018). The selected baseline reports an F1-score of 69% for the ED task using gold-standard entity annotations, while the model from (Nguyen and Grishman, 2018) 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 (Nguyen and Grishman, 2018) the result of the more recent neural model for predicted entities. However, they do report in (Nguyen and Grishman, 2015) 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%.

As mentioned earlier, in order to compare our work with a classical approach, we also implemented a SVM-based ED model using the Scikit-learn library. The SVM approach used Word2vec embeddings as features (W) and it was tested with four different kernels and default parameters.

The CNN model from (Nguyen and Grishman, 2015) was replicated as faithfully as possible. However, although we tried to replicate the model exactly as it is 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. In our work, we use the Spacy library for all the NLP related tasks. In the remainder of this section, we describe in detail the CNN model used. We also describe some minor changes and decisions that we had to make, explaining the rationale behind them.

CNN 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 these features, 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 (Nguyen and Grishman, 2015). The three layers are updated while training for the ED task.

CNN architecture. The architecture used in (Nguyen and Grishman, 2015) 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 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 (Nguyen and Grishman, 2015). An outline of the architecture of the CNN model is presented in Figure 6.

CNN hyperparameters. While many of the hyperparameters remain the same, since our data is different from the one used in (Nguyen and Grishman, 2015), 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 (Nguyen and Grishman, 2015)), 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 the model 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, a batch size of 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.

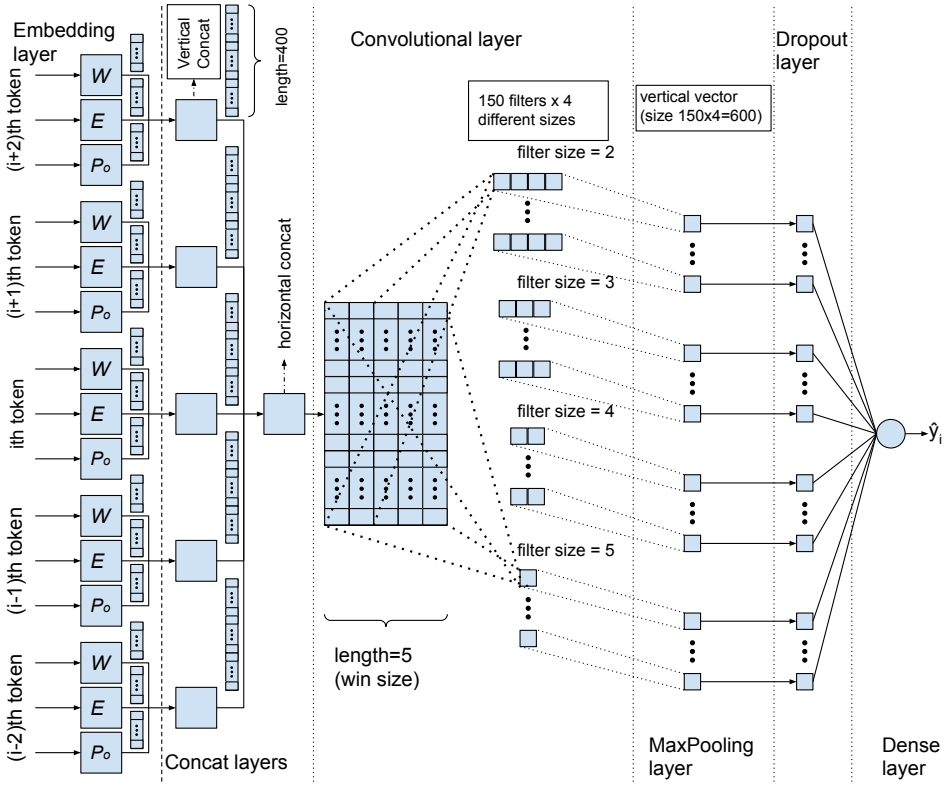


Figure 6: **Outline of the baseline model (CNN)** with window size 5. Descriptions are given from left to right. (a) Inputs. A representation of the three different inputs (Word embedding W, Entity embedding E and Position Embedding Po) for the five tokens in the window. Each input is a one-hot encoding. (b) Embedding layer. Input enters a Keras embedding layer. (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.

5 Results and Discussion

5.1 Experiments Configuration and Reported Metrics

In this section, we present the results and discussions for different variations of the proposed (RNNs) and baseline models (CNNs and SVM). For each variation and each model (excluding the SVM models), we randomly split the training/validation part of the dataset into different training and validation subsets. For each model variation, we trained the model until it did not improve the F1-score on the validation set for 400 epochs (early stopping with patience 400). Although we set a hard limit on the number of epochs (2,000), the models were never trained for that many epochs. All the models stopped training due to the early stopping criterion (when they do not improve for 400 epochs, they stop automatically). We used consecutive seeds from 1 to 10 to guarantee replicability. The model with the best performance on the validation set was selected and used to make the predictions on the test set. In the case of the SVM models, since early stopping was not necessary, we use the training and validation sets together (training+validation) for training.

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 performance of each model on the validation splits and test data. We report results for a total of fifteen different model variations. For selecting these model variations, we run several preliminary studies. The preliminary studies and their discussion are presented in the supplementary material. For each variation of the neural-network models, 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. To thoroughly analyze the F1-score metric on the test 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 first model (model RNN_1) and the best model of the table (model RNN_7). We used the F1-score metric on the test data for the t-test. Since the SVM model is not stochastic, only one run on the training+validation and test data was executed for each of the four kernels. Consequently the results reported are from a single run (not averages), and therefore no t-test could be performed.

Additionally, we conducted Tukey's tests between the performances (in terms of F1-score in the validation set) for all pairs of RNN models (Figure 7a). We also conducted Tukey's tests between all pairs of CNN models and also on all CNN models against the best RNN model (in terms of average F1-score in the validation test), model RNN_7 (Figure 7b). Since Tukey's tests are used here to reduce the probability of joint errors in the analysis of differences among different means in an ANOVA analysis, they allowed us to obtain more robust results.

5.2 Experiments on the Test Set

For the proposed RNN model, we run ten trials for seven different sets of features (models RNN_1 to RNN_7) as depicted in Table 4. Although we tried different numbers of Bi-LSTM layers and hidden units, we only evaluated on the test 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 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 test dataset. The significant drop in performance (of 11.7 percentage points) from including or excluding the contextual word embeddings (model RNN_1 vs. model RNN_2) is indicating the importance of these features for the OED task. Second, we carried out the rest of the experiments (models RNN_3 to RNN_7) as an ablation study to assess the

impact of the proposed features to the overall performance. We measure the contribution of each feature during the preliminary studies. During the experiments on the test dataset, we found further evidence that several inputs are not contributing to the performance in the presence of the other features. Model RNN_7, which excludes all the attributes except the contextual word and sentence embeddings, is the model with the best performance. This result shows that the other features are not relevant in the presence of these two other features, and they were adding unnecessary complexity to the model. The statistical analysis between model RNN_7 and models RNN_1, RNN_2, RNN_3, RNN_5, and RNN_6 shows that all the differences are significant at the level of 95% (p-value$<$0.05). On the other hand, there is less statistical evidence to support that the models RNN_7 and RNN_4 have different performances (significant at the level of 90%, p-value$<$0.1).

To further analyze the differences in performance between the models, we conducted an ANOVA analysis. The results of the ANOVA analysis are presented in Figures 7a and 7b. The main conclusions (analyzed in the caption of the figures) are that all models that contain contextual BERT embeddings are statistically superior to model RNN_2 (the only model without BERT embeddings) and that model RNN_7 (the best RNN model) is statistically superior to all the CNN models.

The baseline CNN model was run for ten trials with different window sizes. Here, we analyze windows of size 1 and 11 (other window sizes are examined in the supplementary material). For both window sizes, we run experiments with and without the entity embedding, but always including the word embeddings. For window size 1, we excluded the position embeddings. The results of these experiments are presented in Table 5. The best CNN baseline (model CNN_1) shows an F1-score of 0.575 on the test data, which is significantly lower than the performance of all the proposed RNN models, except for the one that excludes the contextual word embeddings (model RNN_2).

To compare the RNN models and the best CNN baselines, we evaluated the p-values of single-tail t-tests between the average F1-score for the worst and the best RNN models (models RNN_2 and RNN_7, respectively) against

Model	Features	validation			test				
		$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	F1 CI	p-value
RNN_1	all	0.696	0.945	0.712	0.667	0.931	0.676	± 0.026	0.024
RNN_2	all-$\{B\}$	0.545	0.957	0.591	0.522	0.948	0.559	± 0.046	0.000
RNN_3	all-$\{T,P,D\}$	0.677	0.949	0.697	0.654	0.935	0.666	± 0.031	0.012
RNN_4	all-$\{T,P,D,W\}$	0.703	0.948	0.718	0.686	0.935	0.690	± 0.018	0.076
RNN_5	all-$\{T,P,D,Sp\}$	0.689	0.945	0.706	0.672	0.931	0.683	± 0.013	0.007
RNN_6	all-$\{T,P,D,E\}$	0.686	0.949	0.706	0.662	0.936	0.675	± 0.023	0.012
RNN_7	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 **RNN** model in the validation and test set for seven different sets of features using an architecture with a single Bi-LSTM layer and 15 hidden units. 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 $\{T,P,D,Sp,W,E,B,S\}$. We adopted this notation to simplify our experiments' interpretation as an ablation study (comparing the model with all the features against the models with some features removed). The training set is used for training and the validation set for early stopping, while the test set is held out for testing. The resulting average performance during training is omitted from the reported analysis.

the two best CNN models (models CNN_1 and CNN_2). (1) Model RNN_7 is statistically significantly better than all the others (RNN_2, CNN_1, CNN_2) having a p-value of 0.0 for all the t-tests. The t-test between models CNN_1 and CNN_2 has a p-value of 0.384. (2) This high p-value indicates that the difference in performance of these two models is not statistically significant, which points to the fact the entity embeddings (E) have a negligent impact on performance. (3) The difference in performance of the worst RNN (model RNN_2) with respect to the best CNNs (models CNN_1 and CNN_2) is not statistically significant either (p-value of 0.264 and 0.344, respectively).

Model	Win Size	Features	validation			test				
			$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	$\overline{sens}$	$\overline{spec}$	$\overline{F1}$	F1 CI	p-value
CNN_1	1	$\{W,E\}$	0.477	0.971	0.570	0.499	0.968	**0.575**	± 0.031	—
CNN_2	1	$\{W\}$	0.472	0.972	0.565	0.493	0.969	0.569	± 0.031	0.384
CNN_3	11	$\{W,E,Po\}$	0.507	0.963	0.596	0.326	0.960	0.394	± 0.028	0.000
CNN_4	11	$\{W,Po\}$	0.499	0.965	0.589	0.345	0.942	0.406	± 0.035	0.000

Table 5: Average performance of variations of the **CNN** model on the validation and test sets. The training set is used for training and the validation for early stopping, while the test set is held out for testing. The average performance during training is omitted from the reported analysis.

Conclusions (1), (2) and (3) are confirmed not only by t-tests but also by Tukey's tests.

The baseline SVM model was run on the training and test data with linear, polynomial, RBF and sigmoid kernels. The results of these experiments are presented in Table 6. The best SVM baseline (model SVM_4) shows an F1-score of 0.564 on the test data, which is significantly lower than the performance of all the RNN models except for the one that excludes the contextual word embeddings (model RNN_2). On the other hand, its performance is only slightly inferior to that achieved by the two best CNN baselines (models CNN_1 and CNN_2).

5.3 Discussion

In this section, we review and discuss the behavior of five representative models. The models selected for this discussion are model RNN_2 (RNN with all the features except for the contextual word embeddings), model RNN_7 (best RNN model in terms of average F1-score in the validation set), models CNN_1 (best CNN model for the same metric in the same set), and model SVM_4 (best SVM model for the same metric in the same set).

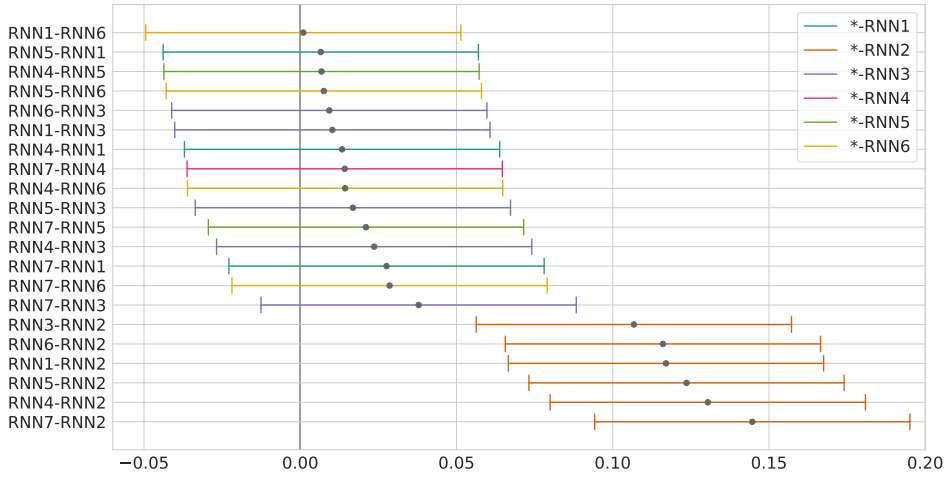
We hypothesize that the contextual embeddings are a crucial factor for boosting the performance of the proposed models. Therefore, in model RNN_2, we exclude only the contextual word embeddings to assess their impact on the performance. Since this model lacks the contextual word embeddings, we expected a poor performance in comparison to model RNN_7 (which has the contextual word and sentence embeddings). The poor performance of model RNN_2, which is lower than the best baseline models, provides evidence to support our hypothesis. Moreover, the superior performance achieved by model RNN_7 (containing only the contextual embeddings B and S as features) indicates that the inclusion of the other features (W, Sp, T, P, D, E) are not useful for the OED task once the contextual embeddings are included. It is also interesting to observe that model SVM_4, which has a *sigmoid* kernel, obtained an F1-score of 0.564 on the test set, outperforming some of the neural-based models (models RNN_2, CNN_3 and

Model	Kernel	training + validation			test		
		sens	*spec*	$F1$	*sens*	*spec*	$F1$
SVM_1	linear	0.247	0.995	0.395	0.231	0.994	0.375
SVM_2	polynomial	0.494	0.993	0.660	0.380	0.980	0.548
SVM_3	RBF	0.459	0.993	0.628	0.346	0.984	0.512
SVM_4	sigmoid	0.372	0.957	0.536	0.401	0.949	**0.564**

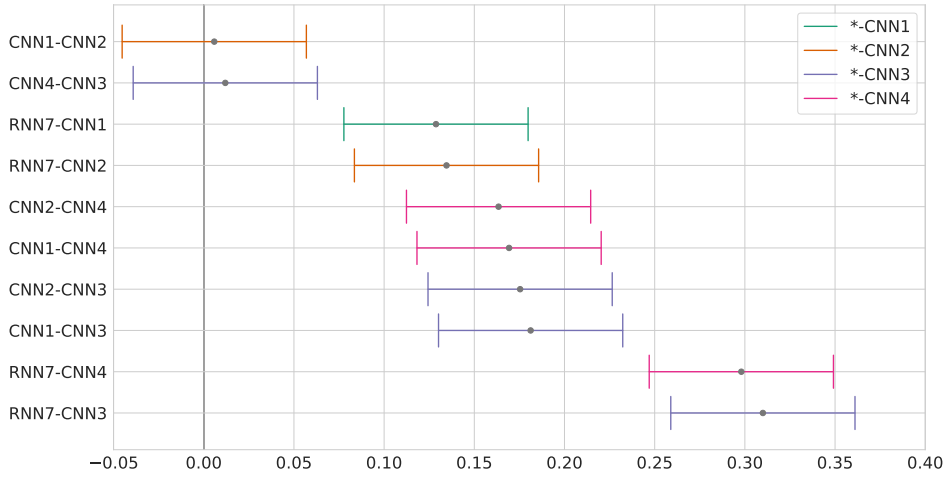
Table 6: Performance of variations of the **SVM** model on the training+validation and test sets. Since SVM has a deterministic output and there is no need for early stopping, there was no need to use the training and validation tests separately. Therefore, unlike the RNN and CNN models, the SVM models were trained using the training and validation sets together (training+validation set).

CNN_4).

We draw two major conclusions from the discussed results. First, in an OED task trained on a small labeled datasets, the use of pre-trained features is crucial. This follows from the fact that BERT-based contextual embeddings (B and S) and Word2vec embeddings (W) had a major impact on the performance of models RNN_7 and SVM_4, respectively. This provides further evidence of the well-known fact that the use of transfer learning in small data settings contributes significantly to improving performance. Second, in the presence of a small labeled dataset, the choice of features, and in particular the use of transfer learning, is more important than having a large or complex model. This follows from the fact that a classical approach (model SVM_4) that relies on pre-trained features outperformed or had a performance comparable to that of some of the more complex neural-network models.



(a)



(b)

Figure 7: The figure shows the difference in performance between each pair of models along with the confidence interval (CI) obtained using Tukey's test at the level of significance of 95%. For each pair of models, if the CI contains the value zero, then there is not enough statistical evidence to conclude that the models' performances are different. On the other hand, if the CI does not contain the value zero, then we can conclude that the difference in performance is significant. Unlike the pairwise test in Table 4 and 5, Tukey's tests take into consideration the number of different means under an ANOVA analysis, i.e., the CIs are larger than those that could be obtained with the t-tests. Therefore, some pairs of models that we found statistically different using the t-test are probably found not statistically different due to larger confidence intervals. However, we still found evidence to support the main claim about the models (that the BERT embeddings provide the greatest improvement in performance for the task). This claim is supported by the fact that all the RNN models are significantly better than model RNN_2 (the only model without BERT embeddings) (Figure 7a) and that model RNN_7 (with BERT) has a significant leap in performance compared with all CNN models (Figure 7b).

6 Conclusions

The main contribution of this work is the definition of the OED task and an extensive evaluation of an OED framework that combines several features. The best of the proposed models, based on an RNN architecture and BERT-based contextual word and sentence embeddings, shows an improvement of 13.3% in F1-score with respect to the best baseline model on the test data. Considering that some more recent approaches for ED are slightly above our baseline (4.1% for (Nguyen and Grishman, 2018)), 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 the code and data for replicating our model are fully available.

Two main conclusions follow from our extensive analysis. First, contextual embeddings are better suited for the OED task than the other analyzed embeddings and features. In particular, some features proved to have a negligible impact on performance, in the presence of other features. For example, grammar information (captured using POS and Dependency Parser taggers), entity tags and non-contextual word embeddings showed to have no positive impact on performance as long as contextual embeddings are accounted for in 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. Also, the analysis of a classical baseline model showed that, in this context of small data, it is sometimes more important to have useful features than to have a complex or large model.

The second conclusion we derive from these results is that although the proposed RNN models (which are best suited for text processing) show more flexibility and effectiveness for the OED task than the analyzed CNN models (which are best suited for computer vision tasks), the major difference in performance between the evaluated models relies on the features used, in particular the BERT-based contextual embeddings.

Another important contribution is the construction of a public dataset

for the OED task. The labeling of the dataset was assisted by an active learning tool. The resulting dataset is particularly useful for the OED task as it focuses on ongoing events only. It also differs from other ED and EE datasets in being independent of a fixed event type taxonomy.

As part of our future work, we plan to evaluate the impact of contextual embeddings on other ED settings (using other datasets and other models).

Acknowledgment

This work was enabled by support provided by CONICET, Universidad Nacional del Sur (PGI-UNS 24/N051 and PGI-UNS 24/E145), ANPCyT (PICT 2019-01640, PICT 2019-02302, and PICT 2019-03944), a LARA project (Google Research Award for Latin America 2019-2021), Compute Canada (`https://www.computecanada.ca`), a New Frontiers in Research Fund Exploration Grant, an ELAP scholarship by the Department of Foreign Affairs, Trade and Development Canada and ACENET (`https://www.ace-net.ca/`).

References

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.

Ahn, D. (2006). The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

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.

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

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.

Cekik, R. and Uysal, A. K. (2020). A novel filter feature selection method using rough set for short text data. *Expert Systems with Applications*, 160:113691.

Chen, X., Zhou, X., Sellis, T., and Li, X. (2018). Social event detection with retweeting behavior correlation. *Expert Systems with Applications*, 114:516–523.

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.

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.

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*.

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.

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.

Fan, B., Fan, W., Smith, C., and Garner, H. (2020). Adverse drug event detection and extraction from open data: A deep learning approach. *Information Processing & Management*, 57(1):102131.

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

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

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438.

Grishman, R. (1997). Information extraction: Techniques and challenges. In Pazienza, M. T., editor, *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*, pages 10–27, Berlin, Heidelberg. Springer Berlin Heidelberg.

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.

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.

Huang, L. and Ji, H. (2020). Semi-supervised new event type induction and event detection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 718–724, Online. Association for Computational Linguistics.

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

Huang, R., Cases, I., Jurafsky, D., Condoravdi, C., and Riloff, E. (2016). Distinguishing past, on-going, and future events: The eventstatus corpus. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 44–54.

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.

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.

Jagannatha, A. N. and Yu, H. (2016). Bidirectional rnn for medical event detection in electronic health records. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting*, volume 2016, page 473. NIH Public Access.

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

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.*

Lai, V. D., Nguyen, M. V., Nguyen, T. H., and Dernoncourt, F. (2021). Graph learning regularization and transfer learning for few-shot event detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2172–2176, New York, NY, USA. Association for Computing Machinery.

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.

Li, L., Jin, L., Zhang, Z., Liu, Q., Sun, X., and Wang, H. (2020). Graph convolution over multiple latent context-aware graph structures for event detection. *IEEE Access*, 8:171435–171446.

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.

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.

Liu, D.-R., Shih, M.-J., Liau, C.-J., and Lai, C.-H. (2009). Mining the change of event trends for decision support in environmental scanning. *Expert Systems with Applications*, 36(2, Part 1):972–984.

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*.

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

Maisonnave, M., Delbianco, F., Tohmé, F., Maguitman, A. G., and Milios, E. E. (2020). Assessing causality structures learned from digital text media. In *Proceedings of the ACM Symposium on Document Engineering 2020*, pages 1–4.

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

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.

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.

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.

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.

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.

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*.

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.

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*.

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.

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

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

Sandhaus, E. (2008). The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

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*.

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)*.

Tong, M., Wang, S., Cao, Y., Xu, B., Li, J., Hou, L., and Chua, T.-S. (2020a). Image enhanced event detection in news articles. In *AAAI*, pages 9040–9047.

Tong, M., Xu, B., Wang, S., Cao, Y., Hou, L., Li, J., and Xie, J. (2020b). Improving event detection via open-domain trigger knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5887–5897, Online. Association for Computational Linguistics.

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

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.

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.

Yao, S., Shuang, K., Li, R., and Su, S. (2021). Fgcan: Filter-based gated contextual attention network for event detection. *Knowledge-Based Systems*, 228:107294.

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

İnanç Arın, Erpam, M. K., and Saygın, Y. (2018). I-twec: Interactive clustering tool for twitter. *Expert Systems with Applications*, 96:1–13.