HANKE: Hierarchical Attention Networks for Knowledge Extraction in Political Science Domain

Erick Skorupa Parolin*, Latifur Khan*, Javier Osorio‡, Vito D'Orazio†, Patrick T. Brandt†, Jennifer Holmes†
Department of Computer Science, School of Economic, Political and Policy Sciences†*
The University of Texas at Dallas, Richardson, Texas
{erick.skorupaparolin, lkhan, dorazio, pbrandt, jholmes}@utdallas.edu
School of Government and Public Policy‡, *University of Arizona*, Tucson, Arizona
josorio1@arizona.edu

***Abstract*—Extracting structured metadata from unstructured text in different domains is gaining strong attention from multiple research communities. In Political Science, these metadata play a significant role on studying intra and inter-state interactions between political entities. The process of extracting such metadata usually relies on domain specific ontologies and knowledge-based repositories. In particular, Political Scientists regularly use the well-defined ontology CAMEO, which is designed for capturing conflict and mediation relations. Since CAMEO repositories are currently human maintained, the high cost and extensive human effort associated with updating them makes it difficult to include new entries on a regular basis. This paper introduces HANKE: an innovative framework for automatically extracting knowledge representations from unstructured sources, in order to extend CAMEO ontology both in the same domain and towards other related domains in political science. HANKE combines Hierarchical Attention Networks as engine for identifying relevant structures in raw-text and the novel Frequency-Based Ranker approach to obtain a collection of candidate entries for CAMEO's repositories. To show the efficiency of the proposed framework, we evaluate its performance on capturing existing CAMEO representations in a soft-labelled dataset. We also empirically demonstrate the versatility and superiority of HANKE method by applying it to two case studies related to CAMEO extension on its actual domain and towards organized crime domain.**

***Index Terms*—hierarchical attention networks, political events, knowledge extraction, CAMEO**

I. Introduction

The generation of structured metadata and Knowledge Extraction techniques play an important role in several academic domains and research areas. Specifically, conflict scholars in Political Science are often interested in studying the interactions between political entities around the world, which requires putting together considerable amounts of *structured* data for analyzing conflict processes [1]. Building these datasets commonly relies on automated data coder systems, which identify, extract, and categorize the political interactions from unstructured text data, converting them into computer friendly structured representations. Such automated coder systems generally require a well defined ontology as knowledge base for capturing critical information from the text, like *who* are the entities or actors involved in *what* type of actions or interactions are they conducting.

The existing standard ontology for political event data is CAMEO (Conflict and Mediation Event Observations) [2], which incorporates a knowledge-base composed by actor dictionaries comprising names of political entities and action-pattern dictionaries for classifying political interactions or actions. CAMEO classifies political interactions in four primary categories (called QuadClass): Verbal Cooperation, Material Cooperation, Verbal Conflict and Material Conflict. Each one of these are further disaggregated, generating 20 more granular event categories, called root codes (shown in Table I).

Political event automated coder systems like PETRARCH, PETRARCH2 [3] and Universal PETRARCH [4] rely on CAMEO dictionaries for coding political events from text. Fig. 1 illustrates the operation of a generic automated coding mechanism using the CAMEO ontology.

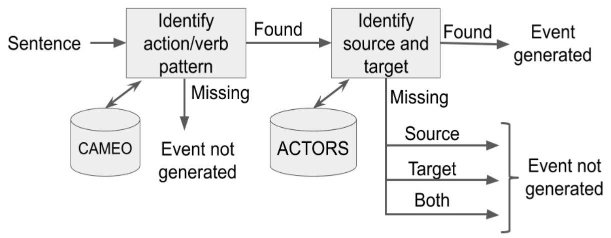


Fig. 1: Basic Mechanism of Automated Coding

Essentially, the coding mechanism receives a sentence in raw-text format as input and searches for a matching pattern in the CAMEO action-patterns dictionary. These patterns resemble verbal phrases and are usually composed of a main verb and some surrounding keywords. Altogether, each set of terms (verb plus surrounding words) relate to an event code, which in turn corresponds to a certain political event. For example, below there are two entries from CAMEO dictionary.

```
* DIPLOMATIC OFFENSIVE TO DEFEND [053] # LAUNCH
* AIRSTRIKE                      [195] # LAUNCH
```

The verb "LAUNCH," in the first entry, followed by the words "DIPLOMATIC OFFENSIVE TO DEFEND" form a pattern that receives the event code 053, which corresponds to the political event "ENGAGE IN DIPLOMATIC COOPERATION." In contrast, if the verb "LAUNCH" is found in the text followed by the token "AIRSTRIKE", this will form another pattern that receives the event code 195 and corresponds to the political interaction "FIGHT".

After detecting an action pattern, the automated event coder uses the actor dictionaries looking for the political entities that are involved in this event. After finding all the pieces of information, the system finally outputs a *political event*, which is a structured way of representing interactions between the political entities expressed in the input sentence. In summary, this structured output consists of a triplet SAT (Source-Action-Target), where the Source corresponds to the acting entity and the Target, to entity being acted upon. Note that, if there is any missing (non-matched) information, the event coder will output no political event.

As an illustration of such a processing, suppose we have the following sentence from a news article:

```
PM Theresa May has struck a last-minute
deal with the EU in a bid to move Brexit
talks on to the next phase.
```

Using this sentence as input for the automated coding process depicted in Fig. 1 will produce the following event:

```
Source - GBRGOV
Target - IGOEUREEC
Action - 057 (Sign formal agreement)
```

PM Theresa May and EU are coded in standard format used for event generation. Another well known jargon used to define this event coded structure is *who-did-what-to-whom*.

Although CAMEO is the dominant ontology for the political science domain, we consider that are still opportunities to improve it in terms of expanding its action-patterns representations. According to a preliminary experiment performed on a corpora of $1,000,000$ news articles scrapped from 400 different news agencies spread around the world, running Universal PETRARCH using CAMEO was able to code events for less than 15% of all the sentences, which corresponds to only 55% of those sentences whose both actors (Source and Target) were matched. These figures not only indicate a low capturing rate in terms of political events but also suggests that a considerable share of potentially relevant sentences are overlooked because there are no matching action-pattern in the CAMEO dictionary.

Currently, CAMEO repositories are human maintained and, therefore, practically static. The high cost and extensive human effort required for manually identifying new representations and updating the dictionaries make unfeasible the inclusion of new political actors and action-patterns in CAMEO on a regular basis. In this way, extending CAMEO repositories either on its actual domain or towards other political science related domains turns an impracticable task.

To overcome this issue, we propose an innovative framework called Hierarchical Attention Networks for Knowledge Extraction (HANKE). Such framework combines two main components: (i) *Hierarchical Attention Networks*, which is used in an inventive manner for identifying relevant structures from raw-text and (ii) the novel *Frequency-Based Ranker* algorithm, which forms new representations based on the relevance of the terms and employs a frequency based strategy as pruning method. As result, HANKE extracts from raw-text the most relevant action-patterns that serve as candidate entries for CAMEO's knowledge base extension. We evaluate the efficiency of HANKE on patterns acquisition task and present two case studies to demonstrate its applicability.

II. Related Works

Although ontology-based classification and image annotation works have shown remarkable results [5]–[9], the topic related to knowledge extraction and ontology extension in the political science domain still is to be explored.

Extracting semantic relationships among entities from unstructured data is usually a task recurrently explored in Information Extraction [10]–[13]. Here, we review relevant approaches that address similar problems as the one concerning this research. Yangarber et. al [14] combine NLP techniques with frequent sequential pattern mining algorithm to mine domain-specific patterns from text documents, that later can be used for document summarizing. Yangarber et al. [15] and Riloff et al. [16] implement bootstrap based procedures, which identify a set of event patterns from non-annotated text, starting from a small set of seed patterns. Huang et al. [17] introduce a bootstrapping based technique to learn event phrases as well as agent terms and purpose phrases associated with civil unrest events.

Other studies focus on identifying and properly coding political and protest events as a classification task. Those approaches largely rely on classical machine learning and deep learning techniques to approach such tasks. Hanna [18] propose a framework combining Support Vector Machines for classification task with Named Entity Recognizer for identifying and coding protest events. Beieler [19] resorts to Convolutional Neural Networks to classify pre-selected sentences into four political event classes. Radford et al. [20] train a Recurrent Neural Networks to identify indicators of protest events in English text data.

Works applying unsupervised and weakly-supervised learning for extracting event patterns present remarkable results. Krause et al. [21] propose a method based on a neural network architecture that only relies on the weak supervision signal that comes from the news published on the same day and mention the same real-world entities. Ideally, such a network explores the co-occurrence of event patterns in news with similar entities to map them into an embedding space. O'Connor et al. [22] describe an unsupervised model for extracting events between major political actors from news corpora and analyze the performance of the model on detecting real-world conflict.

Despite the prevalence of CAMEO in studying conflict processes, most machine learning-based approaches have paid little attention to expanding the CAMEO ontology. Solaimani et al. [23] propose a system to recommend new political actors for expanding the CAMEO's actor dictionaries, but do not extend the system for new action-patterns extraction functionality. Parolin et al. [24] introduce a word-embedding based framework to extract new action-patterns and recommend labels for them based on the existing interactions covered in CAMEO. Makarov [25] presents a framework that first applies dependency parsing task into an input corpora to extract political event candidate patterns and then label propagation for inferring labels to them. Radford [26] et al. design a word2vec based framework to explore the vector-space representation of a small sample of phrases from a pre-defined ontology and extract synonymous event patterns from the existing ones.

HANKE presents valuable advantages over these approaches [24]–[26]. First, it supports CAMEO extension towards any given domain in political and social science sphere. Second, it extracts action-patterns more comprehensively by exploring semantics, not simply syntactic parsing. Finally, HANKE assigns root-codes to extracted action-patterns independently of existing CAMEO patterns.

III. Data Description and Problem Definition

Because most of the experiments conducted in this paper deal with domain-specific data, we dedicate Subsections III-A and III-B to briefly describe their format and sources. After discussing the essence of the data in Subsection III-C, we formally introduce the problem addressed in this paper.

A. *Political Events Data*

To generate the data used on experiments related to CAMEO extension on its actual domain (Subsection V-D), we collected $1,000,000$ news articles using URLs from the RSS Feed of around 400 different news agencies spread around the world. After gathering all the data in regular HTML form, we pre-process the text by cleaning and extracting the main stories. We then use this pre-processed corpora as input for the automated coding mechanism Universal Petrarch [4] connected to the CAMEO ontology.

As discussed in Section I, Universal Petrarch [4] codes events for those sentences which a triplet (Source-Action-Target) was entirely matched on CAMEO's dictionaries. We collect such coded sentences to form the **Coded Events (CE)** dataset. On the other hand, if a SAT triplet is not entirely matched in a sentence, there is still a chance that the coding mechanism recognizes Source and Target as political actors on this sentence (and an event code is not generated just because no action-pattern is matched). We then form the **Not Coded events with Political Actors (NCPA)** dataset by gathering these sentences such that only Source and Target (but not Action) were matched on CAMEO dictionaries. The flow depicted in Fig. 2 illustrates this whole process for obtaining both *CE* and *NCPA* datasets.

Note that each record in *CE* is composed by the text containing the captured SAT political event, the CAMEO root-code corresponding to the Action of this political event (one of the 20 codes shown in Table I), and the existing action-pattern matched on CAMEO dictionary. Further, in Subsection V-C,

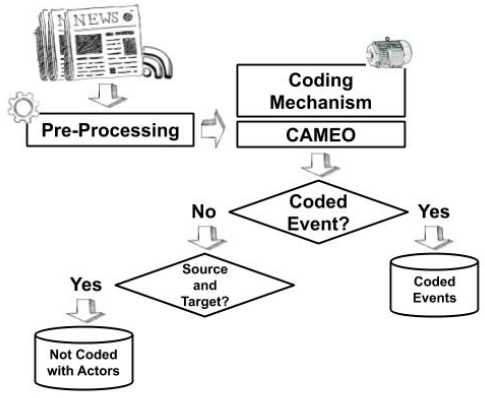


Fig. 2: Political Science Dataset.

CE data is used to evaluate how well our proposed method performs on capturing the existing CAMEO action-patterns.

Overall, *CE* data is composed by $1,920,174$ sentences and Table I presents their frequency by CAMEO root-code.

TABLE I: Frequency of Sentences by CAMEO Political Events.

Root Code	Event Category	Freq. (#)
01	Make Public Statement	558,109
02	Appeal	55,487
03	Express Intent to Cooperate	90,520
04	Consult	228,225
05	Engage in Diplomatic Cooperation	120,742
06	Engage in Material Cooperation	214,553
07	Provide Aid	46,451
08	Yield	79,917
09	Investigate	27,068
10	Demand	29,797
11	Disapprove	87,714
12	Reject	17,288
13	Threaten	19,035
14	Protest	14,113
15	Exhibit Military Posture	4,328
16	Reduce Relations	11,907
17	Coerce	140,273
18	Assault	29,361
19	Fight	143,278
20	Engage in Unconventional Mass Violence	2,008

In addition, we have *NCPA* data containing sentences for which no SAT event was coded because the system could not match an action-pattern. Overall, the *NCPA* data is composed by 1,571,086 records. Since *NCPA* records already include the Source and Target matched from the CAMEO dictionaries, the context of most of their sentences are likely related to political affairs. Therefore, we make use of this data to focus on mining and discovering new action-patterns for CAMEO extension of agent-patterns in Subsection V-D.

B. *Organized Crime Data*

In this subsection, we detail the data used for the experiment presented in Subsection V-E, which demonstrates the usage of HANKE for extending CAMEO towards organized crime domain. For collecting the corpora in this specific domain, we scrapped news articles from July 2004 to March 2020 from

InsightCrime, an online news outlet specialized on reporting organized crime activity in Latin America and the Caribbean [27].

An advantage of using InsightCrime as data source is that some of their stories include tags for specific topics or criminal groups. Conventionally, these tags are used as a way of representing the main features, topics or entities mentioned in the text. Because we are interested in studying the representations of the relations between concepts and entities in this specific domain (organized crime), such tags provide a valuable piece of additional information for our analysis.

Specifically for InsightCrime, the tags express the locality (city, country or state), public person (politics or even famous criminals), organizations (gangs, parties, cartels, etc.) and *crime categories* discussed along the publications.

After scraping the content available in website, we were able to capture $3,134$ news articles in English having tags corresponding to a *crime category*. The 14 crime related tags with their corresponding frequency in the news articles are: Elites and Organized Crime (23.36%), Homicides (12.76%), Human Trafficking or Smuggling (13.31%), Micro-trafficking (9.41%), Arms Trafficking (7.37%), Extortion (6.83%), Money Laundering (6.48%), Contraband or Counterfeit (4.44%), Criminal Migration (4.31%), Kidnapping (4.18%), Illegal Mining (4.15%), Eco-Trafficking (3.54%), Oil Theft (1.63%) and Cyber Crime (1.24%). For the purpose of our experiments, we consider these crime tags as document labels, assuming they represent the crime category reported in each news article.

C. *Problem Definition*

As aforementioned, our goal is to provide a new method for automatically identifying and extracting new action-patterns from unstructured data so we can extend CAMEO repository both in the actual domain and towards other political science domains.

Generalizing, this work focuses on *Knowledge Extraction*, by creating knowledge from unstructured sources and representing it in such a way that facilitates coding previously unidentified political event data. In particular, we seek to identify and extract action-patterns from a domain specific corpus (Political Science and Organized Crime), so the derived knowledge representation can be further reused to capture political interactions in other input documents.

IV. Proposed Framework

The flow in Fig. 3 depicts the steps involved in the proposed framework called Hierarchical Attention Networks for Knowledge Extraction (HANKE). As further detailed in this section, HANKE was designed for knowledge extraction process, more precisely, for action-patterns acquisition task on political science domain.

As shown in the flow, the *Training* step basically consists of training a Hierarchical Attention Network (HAN) model, taking a *Domain Specific Training Dataset*, which consists of a labeled corpora in the domain we want to explore (or to obtain new action-patterns). In the *Knowledge Extraction* step, the previously trained HAN model serves as basis mechanism for extracting knowledge representations from raw-text, by providing both the predictions and the relevance weights for the contents in the documents given as input in *Exploring Dataset*. Finally, the *Frequency-Based Ranker* algorithm is designed to properly generate action-patterns based on the provided weights and prune out those low-frequent formed representations. Note that *Exploring Dataset* input in *Knowledge Extraction* step must be composed by corpus in the same domain as the HAN was trained in *Training* step. In other words, *Domain Specific Training Dataset* and *Exploring Dataset* must be composed by corpus belonging to the same domain (the domain to be explored).

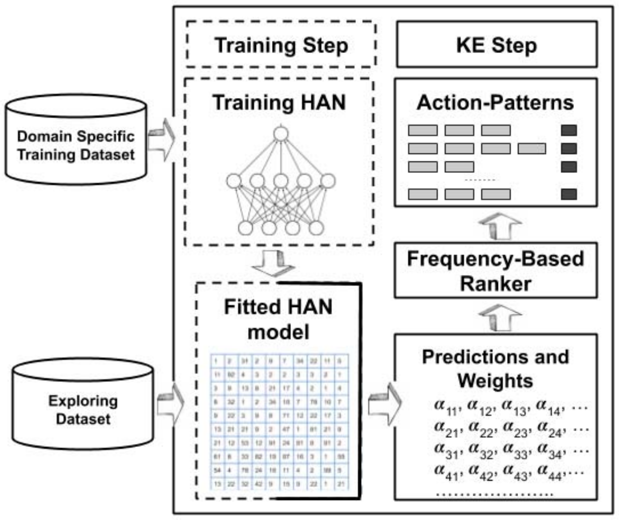

Training Step
KE Step
Training HAN
Action-Patterns
Domain Specific Training Dataset
Frequency-Based Ranker
Fitted HAN model
Predictions and Weights
Exploring Dataset
$\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}, \cdots$
$\alpha_{21}, \alpha_{22}, \alpha_{23}, \alpha_{24}, \cdots$
$\alpha_{31}, \alpha_{32}, \alpha_{33}, \alpha_{34}, \cdots$
$\alpha_{41}, \alpha_{42}, \alpha_{43}, \alpha_{44}, \cdots$

Fig. 3: HANKE for Action-Patterns Extraction.

Subsections IV-A and IV-B will provide enough background for the reader to understand the *Training* step and how the relevance weights are generated in *Knowledge Extraction* step. Subsection IV-C details the *Frequency-Based Ranker* algorithm that finally allows HANKE to extract action-patterns.

A. *The Hierarchical Attention Network Component*

Since the Hierarchical Attention Network (HAN) [28] is an essential component on our framework, we dedicate this subsection to present some important aspects of this architecture, which inspired us on adopting it.

Originally, HAN was designed for document classification task and according to Yang et al. [28], it outperforms previous deep learning based methods by a substantial margin on this task. Its original architecture is composed by the following components connected in bottom-up fashion respectively: *word encoder*, *word attention*, *sentence encoder* and *sentence attention*.

Technically, given an input text, HAN first embeds the words into vectors, through an embedding layer composed by a word2vec model [29]. Following, a bidirectional GRU [30] will provide the annotations of words by summarizing information in both forward and backward directions on sentences.

Therefore, for obtaining the annotation of a specific word t, it concatenates the forward and backward GRU hidden states to obtain the information of the whole sentence centered around t. Both the embedding layer and the bi-GRU together work as the *word encoder* component in HAN, which outputs the annotations of the words.

Next, there is an one-layer FFN to output the hidden representations of the annotations of words, which will then feed the word-level attention layer. These layers form the *word attention* component, which calculates the relevance of the words and output the sentence vectors. The Equations 1 to 3 summarize the math behind *word attention* component:

$$u_{it} = tanh(W_w h_{it} + b_w) \quad (1)$$

$$\alpha_{it} = \frac{exp(u_{it}^T u_w)}{\sum_t exp(u_{it}^T u_w)} \quad (2)$$

$$s_i = \sum_t \alpha_{it} h_{it} \quad (3)$$

Note that, h_{it} denotes the word annotation of a given word t in a sentence i, obtained as output of *word encoder*. Such annotation feeds the FFN layer, which outputs u_{it} as its hidden representation. Equation 2 computes the importance of the word t by measuring the similarity of its hidden representation u_{it} with a word level context vector u_w, which is randomly initialized and gradually learned as the training process advances. The similarity measure is then normalized through a *softmax* function, resulting in α_{it}. In Equation 3, s_i expresses the sentence vector, which summarizes the word annotations weighted by their corresponding word importance.

Analogously to *word encoder* and *word attention* respectively, *sentence encoder* will take the sentence vectors (output from *word attention*) and convert into sentence annotations through bi-GRU, which will feed the *sentence attention* to compute α_i corresponding to the relevance weight of sentence i and output the document representation as a vector v. Finally, on the top of this hierarchical architecture, a *feed-forward network* connected to *softmax* take v as input features to calculate the document classification.

It is to be observed that, the final document classification is internally obtained in HAN by selecting the class corresponding to the highest probability computed on *softmax* layer. As detailed in the next subsections, such predicted document class as well as the words and sentences weights (obtained from Equation 2) will feed the *Frequency-Based Ranker* algorithm in HANKE implementation.

B. Leveraging HAN

While HAN is a method for classification task, in this paper we explore it as a tool to identify relevant words that most contribute to the model prediction.

As detailed in Subsection IV-A, the manner how the attention layers are hierarchically arranged on HAN's structure naturally leads the network to evaluate the importance of the document contents first on words and then on sentences level. As consequence, such structure allows the clear explanation of the model prediction by simply evaluating the attention weights (expressed in Equation 2) of each element on input document.

Therefore, the Hierarchical Attention Network for Knowledge Extraction design takes advantage not only from the good performance presented by HAN model as classifier but also from its ability to explicitly provide human-readable explanations on text. Internally, HANKE explores such *explainability* property by accessing the attention weights of contents in the text in order to extract action-patterns based on the words relevance on predicting a specific event.

The *Training* step of HANKE, depicted in Fig. 3, consists of simply training a model in HAN architecture, in a such way that its attention layers will properly learn where to focus on text to correctly predict the labels of the *Domain Specific Training Dataset*. On the other hand, the *Knowledge Extraction* step focuses on acquiring action-patterns by exploring the *Exploring Dataset*. For performing this job, the previously trained HAN model provides not only the predicted classes but also the input text with its attention weights structured in such a way that allows the *Frequency-Based Ranker* (expressed in Algorithm 1) to put together relevant terms in the text and generate representations in action-patterns format.

Note that, the quality of these action-patterns extracted from HANKE is strongly connected to the performance of the HAN trained model on document classification task. Intuitively, the performance of this model as classifier will tell much about how accurate and precise the obtained patterns will be to indicate a specific label, or in our case, a root-code corresponding to a political interaction (see Table I).

C. Frequency-Based Ranker

As last step of the *Knowledge Extraction* process, we introduce the *Frequency-Based Ranker* algorithm, which was designed for automatically composing patterns based on the relevance of the terms and filtering out low-frequent representations. The algorithm receives as input the text documents with their attention weights and their predicted classes provided by the by the HAN component, as shown in Fig. 3 and described in Subsection IV-B.

The pseudo-code expressed in Algorithm 1 summarizes the steps performed in the Frequency-Based Ranker. As main input, it receives a set of documents, each of which containing additional data corresponding to the attention weights assigned to their contents. Besides, thresholds and output file path are also input as arguments. The algorithm starts going through each sentence of each document, disregarding those sentences that are not relevant enough based on the threshold $SThresh$ given as input. For each of those relevant sentences (line 4), the algorithm iterates over the words in decreasing order of relevance and keeps appending them into a list of relevant words $relW$ (line 9) until reaching an amount of captured relevance defined by the threshold $WThresh$. Note that, in line 5, it sorts (in descending order) the words belonging to sentence $sent$ by their relevance given by attribute $word.\alpha$ and uses sum_α variable to accumulate word's relevance in line 10. Following, the collected relevant words are sorted based on their position on the sentence and stored in $relW\_Sorted$ (line 13), which represents a candidate action-pattern. Next, in line 14, $relW\_Sorted$ with the predicted document class $D.prediction$ form a tuple, which later will serve as key for input in $patterns$ HashMap, in line 15. Finally, in line 16, $ranker$ function simply dumps to $outFile$ only the patterns whose frequency in $patterns$ HashMap is larger than $minFreq$ by each (predicted) class.

Algorithm 1: Frequency-Based Ranker

```
input   : Documents $D$, sentence threshold $SThresh$, word threshold $WThresh$,
          output file $outFile$, minimum frequency $minFreq$
output  : Action-pattern candidates.
1  $patterns \leftarrow HashMap(\emptyset)$
2  foreach $doc$ in $D$ do
3      foreach $sent$ in $doc$ do
4          if $sent.\alpha \geq SThresh$ then
5              $W\_Sorted \leftarrow sort(sent, word.\alpha, desc)$
6              $sum\_\alpha \leftarrow 0$
7              $relW \leftarrow List(\emptyset)$
8              foreach $word$ in $W\_Sorted$ do
9                  $relW.append(word)$
10                 $sum\_\alpha \leftarrow sum\_\alpha + word.\alpha$
11                 if $sum\_\alpha \geq WThresh$ then
12                     break
13             $relW\_Sorted \leftarrow sort(relW, word.pos)$
14             $tuple \leftarrow (D.prediction, relW\_Sorted)$
15             $patterns.increment(tuple)$
16 $ranker(patterns, minFreq, outFile)$
```

It is to be observed that both thresholds $SThresh$ and $WThresh$ should be in the range $[0, 1]$. Furthermore, by the Equation 2, the sum of all the weights (or relevance) of the words belonging to the same sentence, expressed by $word.\alpha$, totals 1. Hierarchically, the same holds for the weights of the sentences $sent.\alpha$ belonging to the same document.

D. *Human and Specialists Evaluation*

Although the proposed framework is able to recommend meaningful action-patterns with an acceptable quality level, it still does not provide enough confidence for adding the output candidates directly to a dictionary without a human expert examination. Besides eventual erroneous recommendations, human inspection is also necessary to semantically evaluate how useful are the outputs, to perform slight changes, and to eventually review the sentences containing such patterns.

V. Computational Experiments and Analysis

A. *Setup*

All the experiments presented in this paper was performed in a computer with NVIDIA Quadro RTX 8000 GPU. The architecture for the HAN component was entirely based on $PyTorch$ implementation, replicating its original configuration (expressed in [28]): we set the word embedding dimension to be 200 (training an unsupervised word2vec model [29]) and the GRU dimension to be 50 both for word and sentence levels. For training process on HANKE, we set a mini-batch of size 32 and stochastic gradient descent with momentum 0.9.

We use Universal PETRARCH [4] as our mechanism of automated coding and UDpipe [31] as universal dependency parser for obtaining both **CE** and **NCPA** already introduced in Subsection III-A. For the following experiments, we use 75% of the data for training and 25% for validation, randomly split based on uniform distribution.

B. *Training Step: HAN performance on Classification Task*

In Subsection IV, we defend hierarchical attention networks as main component on HANKE based on both reasons: its good performance on classification and its ability to provide human-readable explanations on text for its predictions. We still claim that the quality of the action-patterns obtained from HANKE is proportional to the performance of the HAN component on document classification task.

Therefore, we dedicate this subsection to evaluate HAN as classifier, by analyzing it against other baseline methods for document classification. Note that, some of them (like BERT, RoBERTa and ULMFiT) are nowadays considered state-of-the-art language model tools for text analysis tasks, including document classification.

For the purpose of classification task using BERT [32] and RoBERTa [33], we fine tuned their pre-trained base models with an additional single linear layer on top of their base architecture. Similarly, for experiments with ULMFiT [34], we fine-tuned a pre-trained language model and built a RNN learner on the top of that for applying it as classifier.

The numbers presented in Table II summarize the performance of HAN on predicting the category of crime reported by documents in validation portion of Organized Crime data (see Subsection III-B) and compare it to the baseline methods. The results are consistent with the previous analysis expressed by Yang et al. [28], reinforcing the empirical conclusion that HAN outperforms classical machine learning (SVM and Random Forest) and previous deep learning methods like CNN [35] and vanilla LSTM on such task.

TABLE II: Performance for Classification Task in Organized Crime Data

Techinque	Accuracy	Precision	Recall
SVM+TFIDF	0.77908	0.82372	0.77908
RF+TFIDF	0.75686	0.75686	0.75686
CNN	0.76470	0.77746	0.76471
LSTM	0.73445	0.74352	0.73445
HAN	0.85751	0.86323	0.85752
BERT	0.86535	0.86624	0.86536
RoBERTa	0.87712	0.87983	0.87712
ULMFit	0.85882	0.86363	0.85882

On the other hand, BERT and RoBERTa based models presented a slightly better performance. Note, however, that

BERT (and RoBERTa) architecture consists of multiple layers of Transformers [36] stacked to extract high quality language features from text data. Such complex architecture precludes the explicit access to the most relevant contents in the text that explain the predicted classes. Another limitation of BERT based models is the lack of ability to handle long text sequences. By default, BERT supports up to 512 tokens, which often is not enough for covering news articles or texts in political science domain.

In terms of run-time complexity, using BERT or other Transformers-based models [36] for fine-tuning on any specific NLP task may eventually become a prohibitive exercise on CPU architectures due to their structural complexity (e.g. BERT base model contains $110M$ parameters). For the experiment expressed on Table II, training a BERT based model over 20 epochs took $1,962$ seconds while RoBERTa took $1,981$ seconds. For the same values of epochs and mini-batch sizes, HAN took only 97 seconds. Thus showing a considerable advantage on run-time.

The discussion about the results presented in this subsection supports our inspiration on HAN method for implementing HANKE. As conclusion, HAN outperforms many traditional machine learning and deep learning methods, and show results comparable to those obtained when fine-tuning state-of-the-art language modeling tools for document classification. Simultaneously, its *explainability* property (discussed on Subsection IV-B) reinforces our confidence in the choice of HAN as essential component on HANKE design.

C. Validation on Political Coded Events Dataset

For the purpose of validating *Knowledge Extraction* step of HANKE, we make use of **CE** dataset previously described in Subsection III-A. The following experiments concentrate on analyzing how efficient HANKE is for finding existing CAMEO action-patterns matched by the automated coding mechanism. Therefore, we execute some sensitivity exercises to evaluate HANKE outputs on different thresholds based on two main measures: (i) action patterns *capturing rate* and (ii) *false positive term extraction*. The *capturing rate* basically expresses the ratio between the number of existing CAMEO patterns that were captured and the total number of existing patterns matched in the input data. An existing CAMEO pattern is considered captured when the set of terms suggested by HANKE as a candidate pattern contains all the terms that form such existing CAMEO pattern. The *false positive term extraction* indicates, on average, how many terms in the candidate patterns suggested by HANKE are not part of the existing patterns.

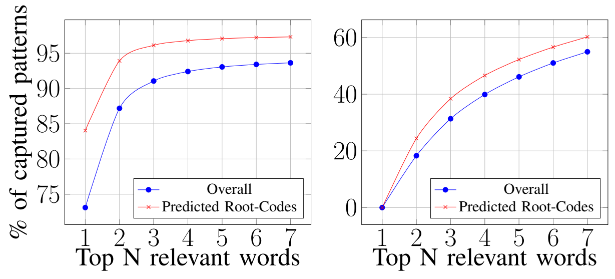
Note that, before performing the *Knowledge Extraction* experiments, we trained HANKE on Coded Events **CE** dataset. Thus, the attention layers in the internal HAN component properly learned how to identify the crucial set words that appearing together on a sentence will lead to a specific type of political interaction. For such training process, we run 20 epochs on HAN and obtained accuracy of 80.2% on validation for predicting the 20 labels listed on Table I.

Table III expresses the results obtained from HANKE when considering the Top N most significant words, according to their attention weights, as candidate patterns suggested by the framework. We evaluate capturing rate and false positive term extraction for *Simple* (CAMEO patterns containing only one term) and *Composed Patterns* (containing more than one term). The numbers show that by selecting only the most informative word for each sentence, we would capture 73.11% of the *Simple* patterns (but obviously 0% of the *Compound* ones). For *Compound* patterns, by setting $N = 6$, we would be able to find 51.01% of all these CAMEO action-patterns. On the other hand, we also would need to discard a large amount of extra words, which may result in more human effort on examining the patterns. As a record, the average size of existing *Compound* patterns is 2.10 terms, which explains the large numbers on *FP terms* for this type of patterns.

TABLE III: TOP N Most Informative Words on Existing CAMEO Patterns.

TOP	Simple		Compound	
N	Captured (%)	FP terms (Avg #)	Captured (%)	FP terms (Avg #)
N=1	73.11%	0.27	0.00%	0.00
N=2	87.19%	1.10	18.29%	0.96
N=3	91.06%	2.05	31.34%	1.75
N=4	92.41%	3.03	39.89%	2.61
N=5	93.06%	4.02	46.12%	3.51
N=6	93.42%	5.01	51.01%	4.44
N=7	93.64%	6.00	54.96%	5.37

If we focus on analyzing only those records for which the trained HAN was able to correctly classify the root-code labels, we observed HANKE could obtain a higher capturing rate of existing patterns. This empirically supports our argument expressed on Subsection IV-B, showing evidences that the performance of the HAN classifier is strongly connected to the quality of the patterns extracted from HANKE. The Graphs depicted on Fig. 4 illustrate the gap on capturing rate for the whole dataset versus the same measure considering only the records whose root code labels were correctly predicted.



(a) Simple (b) Compound

Fig. 4: Capturing rate Overall versus Correctly Predicted labels (Top N relevance threshold).

Instead of creating candidate patterns based on the Top N most relevant words, we could consider a threshold based on the *aggregated attention weights* K. In other words, we collect the terms in a given sentence in decreasing order of relevance until the summation of their corresponding attention weights reaches a threshold K. To illustrate the idea, we take the real record in **CE** dataset:

```
- Sentence: That attack led US President
Donald Trump to launch the first US air
strikes on a Syrian air base.
- Root-Code: FIGHT
- CAMEO Pattern: LAUNCH * AIR STRIKES
```

For this entry, HANKE was able to correctly identify the political interaction as FIGHT and internally calculated the importance of the each word. The five most relevant words with their corresponding weights in decreasing order are:

```
'strikes': 0.23808932
'air': 0.17757249
'launch': 0.16088013
'US': 0.08372052
'led': 0.06590183
```

For this case, the summation of the weights corresponding to 'strikes' and 'air' will sum up to 0.41566181 and by adding the weights of 'launch' word we would obtain 0.57654194. Therefore, by setting the threshold $K = 0.4$ would not be enough for HANKE to correctly capture this pattern. On the other hand, assuming $K = 0.5$ would make HANKE to generate exactly the existing CAMEO pattern.

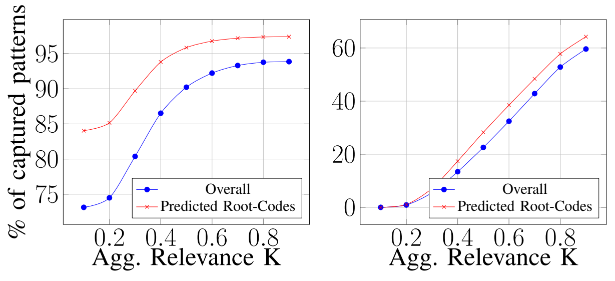
To test this design, we experiment a range of values for threshold K varying from 0.1 to 0.9. Table IV summarizes the results for such experiment considering the same measures previously evaluated plus the *average size of the extracted patterns* in terms of number of words.

TABLE IV: Aggregated Attention Weight K as Threshold on Existing CAMEO Patterns.

K	Simple Capt. (%)	Simple Size (Avg#)	Simple FP (Avg#)	Compound Capt. (%)	Compound Size (Avg#)	Compound FP (Avg#)
0.1	73.13%	1.00	0.27	0.00%	1.01	0.34
0.2	74.49%	1.08	0.34	0.90%	1.14	0.43
0.3	80.38%	1.37	0.55	5.52%	1.52	0.68
0.4	86.52%	1.78	0.90	13.43%	2.04	1.03
0.5	90.22%	2.37	1.43	22.57%	2.70	1.54
0.6	92.23%	3.24	2.27	32.43%	3.63	2.31
0.7	93.31%	4.57	3.58	42.82%	4.95	3.48
0.8	93.77%	6.51	5.51	52.81%	6.72	5.11
0.9	93.86%	8.52	7.52	59.61%	8.53	6.83

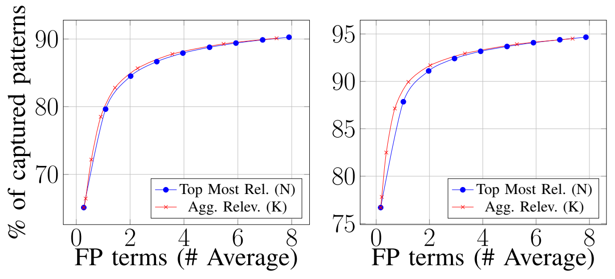
When evaluating HANKE with the threshold based on aggregated relevance (or Aggregated Attention Weight) focusing on the correctly classified records only, we observe the same effect as we previously did: the capturing rate of existing patterns increases when comparing to the figures obtained from the overall dataset (reported on Table IV). Similarly as we have done when evaluating the Top N based threshold, the graphs depicted in Fig. 5 shows the gap in terms of capturing rate for the overall dataset versus the correctly classified records only.

The graphs presented on Fig. 6 show that on the overall view (combining both Simple and Compound patterns), both ways of setting the thresholds (Top N and Aggregated Relevance K) are similarly efficient for capturing the patterns. Given the slightly better efficiency observed for Aggregated Relevance fashion, we constructed the Algorithm 6 to incorporate this design, by accepting *WThresh* as input argument. Note that, in Fig. 6 (b), the Aggregated Relevance based threshold provides less False Positive terms (on average) for the same capture rate on Correctly Predicted records.



(a) Simple (b) Compound

Fig. 5: Capturing rate Overall versus Correctly Predicted labels (Aggregated Relevance threshold).



(a) Overall (b) Correctly Predicted Only

Fig. 6: Top Relevance versus Aggregated Relevance based thresholds for capturing CAMEO Existing patterns.

To close the discussion about the experiments presented on this subsection, we highlight the good potential of HANKE on capturing the existing CAMEO patterns on **CE** soft labeled data (more than 90% of Simple existing patterns and around 40% of Compound Existing Patterns by setting K between 0.6 and 0.7). Furthermore, HANKE shows even better capturing rate on those records for which HAN component could correctly predict the labels. This is an interesting property to be considered when exploring a dataset for which the labels are known, like we do in Subsection V-E.

D. Case Study 1: CAMEO Extension on the same domain

The first case study consists on applying HANKE for extending CAMEO in its existing domain. In other words, the goal of this case study is to extract new action-patterns that lead to one of the 20 types of political interactions shown in Table I, covered in CAMEO domain. For this task, we used the **NCPA** dataset, already detailed in Subsection III-A and HANKE with HAN component trained on **CE** dataset (same model used for experiments in Subsection V-C).

By running HANKE's *Knowledge Extraction* step with a Frequency-Based Ranker Algorithm 1 set at $WTresh$ = 0.75 and $minFreq$ = 10, we obtained $1,587$ action-patterns overall: 701 simple and 886 compound.

We submitted a random sample of 100 instances out of the $1,587$ patterns (with the sentences they appear) to the validation of a political science experts committee. In the case of disagreements in the final evaluations, we considered the majority of votes for each instance. As a result, 82 out of the 100 patterns were considered meaningful action-patterns for extending CAMEO. From these 82, HANKE provided the correct root-codes for 59 of them, with 17 not requiring any change (add or remove terms).

As an example of a well formed extracted action-pattern and a sentence where it appears, we have:

```
- Candidate Pattern: impose * sanctions
- Root-Code: 16 - REDUCE RELATIONS
- Sentence: Canada had already imposed
sanctions on senior Venezuelan officials in a
move which angered the Venezuelan government.
```

On the other hand, below we present an example of an action-pattern which is part of the event but it does not completely describe this event. Therefore, such pattern would certainly require refinements during human evaluation:

```
- Candidate Pattern: approve
- Root-Code: 08 - YELD
- Sentence: Earlier this month, the Israeli
ministers were set to approve a bill absorbing
major Israeli settlements currently in the
occupied West Bank into Jerusalem by enlarging
the city limits.
```

E. Case Study 2: CAMEO Extension Towards Organized Crime domain

The main goal for this case study is to use HANKE for extending CAMEO towards a new domain, instead of only enriching the knowledge bases of the existing ontology. For this task, we will use the Organized Crime dataset detailed in Subsection III-B. We assume that the crime types given as labels in the dataset constitute the new concepts that we want to include as new interactions for extending CAMEO.

For the purpose of this exercise, we performed both the *Training* and *Knowledge Extraction* HANKE steps on the Organized Crime dataset with $STresh$ = 0.25 and $WTresh$ = 0.75. In order to improve the quality of the extracted patterns, we restricted our exploration to the records for which HANKE correctly classified the type of crime (given as label). We obtained 652 action-patterns with their corresponding concepts. A small sample of the output patterns and their corresponding sentences are listed below.

```
- Pattern: arrested * accused laundering money
- Label: MONEY LAUNDERING
- Sentence: Police have arrested a Colombian
woman accused of laundering money for the
Sinaloa cartel,...
```

Note that this example expresses a generic type of pattern. For instance, by replacing the term *laundering money* by *trafficking arms* (or some synonym related to this category of crime), we would obtain a new action-pattern representing the category of crime *Arms Trafficking*.

The same is observed for the following examples:

```
- Pattern: is involved * laundering money
- Label: MONEY LAUNDERING
- Sentence: Prosecutors believe the businessman
is involved in laundering money for the Teofilo
Forero column of the revolutionary...
```

```
- Pattern: control illegal mining operations
- Label: ILLEGAL MINING
- Sentence: Colombia's national police revealed
that the FARC and Bacrims control illegal
mining operations in nearly half of ...
```

Furthermore, the output generated additional non-trivial action-patterns with more meaningful structure. For example:

```
- Pattern: confiscate * cigarettes
- Label: CONTRABAND OR COUNTERFEIT
- Sentence: Brazilian police confiscated the
cigarettes in the state of Rio Grande do Sul,
in the southernmost part of the country.
```

```
- Pattern: prostitution * exploit * minors
- Label: HUMAN TRAFFICKING OR SMUGGLING
- Sentence: Authorities in Colombia have broken
up a prostitution ring exploiting indigenous
minors in the Amazonas department, ...
```

```
- Pattern: trafficker move drugs
- Label: MICROTRAFFICKING
- Sentence:  Violence and criminality are
on the rise on the island, which has long
been used as a transshipment point for
traffickers moving drugs from South ...
```

Members of the research team with political science background and expertise on criminal behavior consider that these results are promising. There may be need of implementing additional refinements to improve the output, but the general direction is encouraging.

VI. Final Comments and Future Works

In this paper we introduced an innovative framework called HANKE, which combines HAN and the novel Frequency-Based Ranker algorithm to automatically extract knowledge representations from unstructured sources, in order to extend CAMEO's knowledge bases. We initially showed the performance of HAN on classification task and empirically evidenced the correlation between its good performance on this task with the quality of the action-patterns extracted from HANKE.

Furthermore, we analyzed HANKE's potential on identifying existing CAMEO's action patterns on a soft labeled dataset. The satisfactory results on this experiment show evidences favoring HANKE's capability on extracting high quality representations for enriching CAMEO's knowledge bases.

Finally, we demonstrate real world applications of HANKE for extending CAMEO both on the same domain (conflict and mediation relations) and towards a new domains in political science field (Organized Crime). Promising action-patterns were obtained from both case studies and human experts examination of a random sample of the extracted representations supports HANKE as a good alternative for extending CAMEO.

An open discussion for future work is to investigate whether HANKE will perform well for other languages besides English and domains different than those explored in this paper. Furthermore, we intend to expand the case study on Organized Crime domain by collecting additional data from other channels covering different localities and crime categories.

Acknowledgements

The research reported herein was supported in part by NSF awards DMS-1737978, DGE-2039542, OAC-1828467, OAC-1931541, DGE-1906630; and an IBM faculty award (Research).

References

[1] S. Salam, P. Brandt, J. Holmes, and L. Khan, "Distributed framework for political event coding in real-time," *Proceedings of the 2018 conference on Electrical Engineering and Computer Science (EECS)*, 2018.

[2] D. J. Gerner, P. A. Schrodt, and O. Yilmaz, "Cameo conflict and mediation event observations event and actor codebook," Available at http://data.gdeltproject.org/documentation/CAMEO.Manual.1.1b3.pdf (2020/05/22), 2012, unpublished Manuscript.

[3] C. Norris, "Petrarch 2: Petrarcher," Available at https://ssrn.com/abstract=2733766 (2020/05/22), 2016, unpublished Manuscript.

[4] J. Lu, "Universal petrarch: Language-agnostic political event coding using universal dependencies," Available at https://github.com/openeventdata/UniversalPetrarch (2020/05/22).

[5] C. Breen, L. Khan, and A. Ponnusamy, "Image classification using neural networks and ontologies," *Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, pp. 98–102, 2002.

[6] L. Khan and L. Wang, "Automatic ontology derivation using clustering for image classification," *Proceedings of the 8th International Workshop on Multimedia Information Systems*, pp. 56–65, 2002.

[7] L. Khan, "Ontology-based information selection," Ph.D. dissertation, Los Angeles CA, 2000.

[8] F. Luo and L. Khan, "Ontology construction for information selection," *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, pp. 122–127, 2002.

[9] L. Khan, D. McLeod, and E. Hovy, "Retrieval effectiveness of an ontology-based model for information selection," *The VLDB Journal*, pp. 71–85, 2004.

[10] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011, 2009.

[11] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 541–550, 2011.

[12] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 2124–2133, 2016.

[13] S. Wu, K. Fan, and Q. Zhang, "Improving distantly supervised relation extraction with neural noise converter and conditional optimal selector," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7273–7280, 2019.

[14] M. Du and R. Yangarber, "Acquisition of domain-specific patterns for single document summarization and information extraction," *Proceedings of the 2nd International Conference on Artificial Intelligence and Pattern Recognition*, p. 30, 2015.

[15] R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen, "Automatic acquisition of domain knowledge for information extraction," *Proceedings of the 18th conference on Computational linguistics*, vol. 2, pp. 940–946, 2000.

[16] E. Riloff, R. Jones *et al.*, "Learning dictionaries for information extraction by multi-level bootstrapping," *Proceedings of the 16th National Conference on Artificial Intelligence*, pp. 474–479, 1999.

[17] R. Huang and E. Riloff, "Multi-faceted event recognition with bootstrapped dictionaries," *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 41–51, 2013.

[18] A. Hanna, "Mpeds: Automating the generation of protest event data," Available at https://osf.io/preprints/socarxiv/xuqmv (2020/05/22), 2017, unpublished Manuscript.

[19] J. Beieler, "Generating politically-relevant event data," *In Proceedings of the First Workshop on NLP and Computational Social Science*, pp. 37–42, 2016.

[20] B. Radford, "Multitask models for supervised protest detection in texts," Available at https://arxiv.org/abs/2005.02954 (2020/05/22), 2019, unpublished Manuscript.

[21] S. Krause, E. Alfonseca, K. Filippova, and D. Pighin, "Idest: Learning a distributed representation for event patterns," *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1140–1149, 2015.

[22] B. O'Connor, B. M. Stewart, and N. A. Smith, "Learning to extract international relations from political context," *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 1094–1104, 2013.

[23] M. Solaimani, S. Salam, L. Khan, P. T. Brandt, and V. D'Orazio, "Repair: Recommend political actors in real-time from news websites," *Proceedings of the International Conference on Big Data (Big Data)*, pp. 1333–1340, 2017.

[24] E. S. Parolin, S. Salam, L. Khan, P. Brandt, and J. Holmes, "Automated verbal-pattern extraction from political news articles using cameo event coding ontology," *International Conference on Intelligent Data and Security*, pp. 258–266, 2019.

[25] P. Makarov, "Automated acquisition of patterns for coding political event data: two case studies," *Proceedings of the 2nd Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social, Sciences, Humanities and Literature*, pp. 103–112, 2018.

[26] B. J. Radford, "Automated dictionary generation for political eventcoding," *Political Science Research and Methods*, pp. 1–15, 2019.

[27] "Insight crime: Investigation and analysis of organized crime," Available at https://www.insightcrime.org (2020/05/22).

[28] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1480–1489, 2016.

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

[30] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *Proceedings of the International Conference on Learning Representations*, 2015.

[31] M. Straka, J. Hajic, and J. Straková, "Udpipe: trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing," *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pp. 4290–4297, 2016.

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

[33] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019, unpublished Manuscript.

[34] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *Association for Computational Linguistics*, 2018.

[35] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1746–1751, 2014.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.