New Approaches to Event Detection and Extraction from News Articles

Sneha Mehta

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Narendran Ramakrishnan
Ellen Riloff
Chang-Tien Lu
Huzefa Rangwala
B. Aditya Prakash

May 12, 2021
Arlington, Virginia

Keywords: Event Extraction, Deep Learning, Natural Language Processing

Copyright 2021, Sneha Mehta

New Approaches to Event Detection and Extraction from News Articles

Sneha Mehta

(ABSTRACT)

Event extraction refers to extracting specific knowledge of incidents from natural language text and consolidating it into a structured form. Some important applications of event extraction include search, retrieval, question answering and event forecasting. However, before events can be extracted it is imperative to detect events i.e. identify which documents from a large collection contain events of interest and from those extracting the sentences that contain the event related information. This task is challenging because it is easier to obtain labels at the document level than finegrained annotations at the sentence level. Current approaches for this task are suboptimal because they directly aggregate sentence probabilities estimated by a classifier to obtain document probabilities resulting in error propagation. To alleviate this problem we propose a method to compute document embeddings from sentence embeddings by leveraging attention and training a document classifier on those embeddings to mitigate the error propagation problem. However, we find that existing attention mechanisms are inept for this task, because either they are suboptimal or they use a large number of parameters. To address this problem we propose a lean attention mechanism which is effective for event detection. Current approaches for event extraction rely on finegrained labels in specific domains. Extending extraction to new domains is challenging because of difficulty of collecting finegrained data. Machine reading comprehension(MRC) based approaches, that enable zero-shot extraction struggle with syntactically complex sentences and long-range dependencies. To mitigate this problem, we propose a syntactic sentence simplification approach that is guided by MRC model to improve its performance on event extraction.

New Approaches to Event Detection and Extraction from News Articles

Sneha Mehta

(GENERAL AUDIENCE ABSTRACT)

Event extraction is the task of extracting events of societal importance from natural language texts. The task has a wide range of applications from search, retrieval, question answering to forecasting population level events like civil unrest, disease occurrences with reasonable accuracy. Before events can be extracted it is imperative to identify the documents that are likely to contain the events of interest and extract the sentences that mention those events. This is termed as event detection. Current approaches for event detection are suboptimal. They assume that events are neatly partitioned into sentences and obtain document level event probabilities directly from predicted sentence level probabilities. In this dissertation, under the same assumption by leveraging representation learning we mitigate some of the shortcomings of the previous event detection methods. Current approaches to event extraction are only limited to restricted domains and require finegrained labeled corpora for their training. One way to extend event extraction to new domains in by enabling zero-shot extraction. Machine reading comprehension(MRC) based approach provides a promising way forward for zero-shot extraction. However, this approach suffers from the long-range dependency problem and faces difficulty in handling syntactically complex sentences with multiple clauses. To mitigate this problem we propose a syntactic sentence simplification algorithm that is guided by the MRC system to improves its performance.

Dedications

To my wonderful Mom, Dad, and Brother

Acknowledgements

First and foremost, I would like to thank my advisor Dr. Naren Ramakrishnan for not only giving me an opportunity to work in his lab but also for his continued support, patience and encouragement throughout my stay at Virginia Tech. Thank you. This dissertation would be much less than what it is if not for the thoughtful and insightful comments from my committee. Thank you, Dr. Ellen Riloff, Dr. C.T. Lu, Dr. Huzefa Rangwala and Dr. B. Aditya Prakash. I am immensely greatful to Dr. Kurt Luther, Director of Crowd Intelligence Lab at Virginia Tech for his unwavering support and patience in the early years when I was still figuring out my research interests. I would also like to thank Ritwik Kumar, Vinith Misra, Boris Chen, Bahareh Azarnoush and everyone I collaborated with for making the summers of 2018 and 2019 at Netflix very exciting, productive and a great learning experience. I would like to thank my labmates including Sathappan Muthiah, Nikhil Muralidhar, Mohammad Raihanul Islam and Subhodip Biswas for collaborations and insightful discussions. I would also like to extend my sincere thanks to Juanita Victoria, Joyce Newberry, Jessica Mullins and Roxanne Paul for their help with all administrative tasks and for their continued encouragement. Finally, I would like to thank my friends and fellow graduate students Taha Hassan, Nurendra Chakraborty and Sukrit Venkatagiri for making my stay at Virginia Tech fun.

Contents

List of Figures x

List of Tables xiii

1 Introduction 1

1.1 Motivation 1

1.2 Event Extraction Overview 2

1.2.1 Event Coding 5

1.2.2 Related Tasks 5

1.3 Definition of Event 6

1.4 Definition of Event Detection & Extraction 7

1.5 Limitations of Existing Systems 9

1.5.1 Limitations of Existing Event Detection Systems 9

1.5.2 Limitations of Existing Event Extraction Systems 11

1.6 Goals and Contributions 14

1.6.1 Compact Multi-Head Self-Attention for Text Classification 14

1.6.2 Event Detection using Hierarchical Multi-Head Attention 15

1.6.3 Zero-Shot Event Extraction using Sentence Simplification 16

2 Attention for Text Classification **17**

2.1 Introduction . . . 17

2.2 Background . . . 19

2.3 Methods . . . 20

2.3.1 Sequence Encoder . . . 21

2.3.2 Single-Head Attention . . . 23

2.3.3 LAMA . . . 24

2.4 Experiments . . . 28

2.4.1 Sentiment Analysis . . . 29

2.4.2 News Classification . . . 30

2.4.3 Comparative Methods . . . 31

2.5 Results . . . 32

2.5.1 Contextual Attention Weights . . . 34

2.5.2 Why Multiple Heads? . . . 36

2.5.3 Runtime . . . 37

2.5.4 Parameters vs Accuracy . . . 39

2.5.5 Attentional Unit Efficiency . . . 39

2.6 Related Work . . . 42

2.7 Discussion . . . 43

3 Event Detection **45**

3.1 Introduction . . . 46

3.2 Hierachical Model for Event Detection . . . 49

3.2.1 Sequence Encoder . . . 49

3.2.2 Intra Sentence Attention . . . 50

3.2.3 Intra Document Attention . . . 52

3.3 Experiments . . . 55

3.3.1 Datasets . . . 55

3.3.2 Comparative Methods . . . 57

3.4 Results . . . 59

3.4.1 Event Detection . . . 59

3.4.2 Event Probabilities . . . 60

3.4.3 Event Extent and Trigger Extraction . . . 63

3.4.4 Event Attribute Extraction . . . 63

3.4.5 Transfer Learning for Attribute Extraction . . . 65

3.5 Related Work . . . 66

3.6 Discussion . . . 67

4 Event Extraction **69**

4.1 Introduction . . . 69

4.2 Methodology 72
4.2.1 Unsupervised Context simplification 73
4.2.2 Scoring function 74
4.2.3 Reading Comprehension Model 78
4.3 Datasets 78
4.3.1 Question-Answer pair Generation 79
4.4 Evaluation 80
4.5 Results 80
4.5.1 Long Range Dependencies 81
4.5.2 Qualitative Analysis 83
4.5.3 In-Domain Training 85
4.6 Related Work 87

5 Conclusion **89**

5.1 Future work 90

Bibliography **92**

List of Figures

1.1 The process of event detection and extraction. 8

2.1 LAMA as unimodal feature fusion. Each dimension in the output z represents a score between context c and an input term u_t 20

2.2 Figure describes a schematic of the model architecture and its major components including the Sentence Encoder, proposed multi-head attention mechanism LAMA, Structured Sentence Embedding and finally the MLP classifier. The attention computation is demonstrated for a single word. 24

2.3 Attention weight (x-$axis$) distribution of the positive words 'amazing', 'happy' & 'recommend' and negative words 'poor', 'terrible', & 'worst'. Positive words tend to get higher weights in reviews with higher ratings (3-5) whereas negative words get higher weights for lower ratings (1-2) 36

2.4 Figure shows the effect of using multiple attention heads. Validation accuracy of LAMA is plotted for different values of m for the Yelp-L dataset(left) and the IMDB dataset(right). x-$axis$ indicates the number of epochs, y-$axis$ indicates the accuracy. Accuracy peaks at $m = 15$ for both Yelp-L and IMDB. 37

2.5 Figure shows the average run time per epoch in seconds (averaged over 10 epochs) for LAMA Encoder (LE) and Transformer Encoder (TE) models as a function of input sequence length (50 to 250). It can be seen that TE (green) is more computationally expensive compared to LAMA (blue). LE is LAMA attention mechanism applied directly to word embeddings without computing GRU hidden states. 38

2.6 Figure shows the accuracy (y-axis) of the models LAMA, SAN, TE and BERT models on YELP-P dataset when viewed against the model parameters (x-axis). LAMA outperforms SAN and TE while also being more parameter efficient. BERT outperforms LAMA by 1.8% but by more than an order of magnitude increase in parameters. 39

2.7 Comparison of number of trainable parameters in the attention layer of SAN, LAMA and TE. Number of parameters increase linearly for SAN and LAMA, where as number of parameters in LAMA are less than SAN by a constant. Number of parameters in TE exceed LAMA by an order of magnitude. . . . 40

3.1 Supervised learning and multiple instance learning(MIL). In MIL a bag is positive if alteast once instance in the bag is positive and a bag is negative if all instances are negative. 45

3.2 An illustration of the model architecture. The LAMA component is used at the word level to get the matrix sentence embedding. 55

3.3 Comparison of event probabilities assigned by LAMA, MICNN and HAN on the CU English, Spanish and MANSA datasets. We can clearly see that mean probability is greater for LAMA in all the datasets for the event class. This depicts that LAMA is usually more confident than other methods in classifying the event articles. We also observe MANSA dataset contains some outliers (shown in red dots at the bottom). 61

3.4 Event Extraction accuracy for event type detection and population class detection from key sentences extracted by different models. LAMA outperforms all the methods. 62

4.1 An example of an event of the type 'Bring lawsuit against' from the ICEWS dataset (left). A generated QA record for that event (right). 70

4.2 Proposed approach. 72

4.3 The length distributions of sentences before and after simplifications. 83

List of Tables

1.1 Event terms and their definitions as used throughout this dissertation. . . . 8

2.1 Important notations. Unless specified bold upper-case letters indicate matrices, bold lower-case letters indicate vectors, unbolded lower-case letters indicate scalars in this dissertation. 24

2.2 Dataset statistics. # words indicates the average number of tokens per document in the corresponding datasets. 28

2.3 Performance with and without usage of the proposed LAMA mechanism. LAMA leads to improved performance over BiGRU alone. 32

2.4 Table reports the accuracy of the proposed models (LAMA, LAMA+ctx) against various baselines on sentiment analysis and news classification tasks. $+D_p$ refers to LAMA + Ctx with position-wise regularization whereas $+D_e$ refers to LAMA + Ctx with regularization over embeddings. 32

2.5 Top attended words for Yelp dataset from reviews with ratings 1 and 5 (indicated in parantheses) and Reuters(r8) Dataset. 35

3.1 First row indicates population classes participating in a protest in the CU dataset. Second row indicates the causes of protest. 56

3.2 Dataset statistics. Total number of news articles, average number of sentences per article and average number of words per article in the datasets. 57

3.3 Various baselines and their key characteristics. 57

3.4 Results Event Detection. LAMA refers to the proposed model. BSA refers to the Bilinear Single-aspect Attention mechanism presented in eq. 3.4, MI-GCNN refers to our MIL model, where the CNN encoder is replaced by the RNN encoder followed by simple dot product attention to extract key words. HAN, HSA & MI-CNN are other baselines. 58

3.5 Event Visualization. Green color highlights refer to Event Type keywords, Pink Color highlights refer to Population Keywords and Yellow Color highlights refer to Protest Related Keywords. These are top attended key words by an attention-based classifier picked from the top sentences selected by our model indicating that our model picks the most informative keywords. . . . 62

3.6 Top triggers extracted from sentences extracted by LAMA for each population type. 66

3.7 Top triggers extracted from event spans extracted by LAMA for each protest type. 66

3.8 Results of transferring different layers of the HAN model trained on the Population Type prediction task to Protest Type prediction task. For protest type prediction different layers as list in the first column were initialized from the corresponding weights in the population type model followed by finetuning the model. 67

4.1 Table lists the ICEWS event types used and their corresponding predicates that were identified for generating question templates. 73

4.2 Results of zero-shot event extraction on the ICEWS dataset. ν_{lm} coefficient $a = 1.5$ and ν_{entity} coefficient $b = 1$ for all settings in which simplification is performed. Δ $+ve$ indicates the % of records for which F1 improves after simplification, Δ $-ve$ indicates the % of records for which F1 becomes worse after simplification and Δ same indicates the % of records for which F1 remains unchanged. 82

4.3 Qualitative examples of zero-shot performance of MRC model before and after simplifying the context using the proposed algorithm. Underlined words are ground truth answers, emphasized words are predicates(triggers) and strikethrough indicates that words were removed by the algorithm. 86

4.4 Table shows the performance of a BERT-base-uncased model finetuned on in-domain dataset. It can be seen that even after finetuning, RUSS approach improves model performance (BERT-MRC-Simple). 87

Chapter 1

Introduction

1.1 Motivation

We're living in an information age; where there's a constant influx of information from social networks, online news, blogs, search engines etc. This has led to a surge in research related to utilizing this vast assortment of data to support decision making processes by encoding and forecasting upcoming events. Moreover, in recent times with the advancement of large scale data processing technologies both in terms of compute and algorithms it has become possible to leverage the large amount of data in novel ways. However, to make the data useful for downstream applications, it is important to parse unstructured text and convert it into a structured form. This can be facilitated by information extraction(IE) technology which enables automatic identification and classification of instances of user-specified types of entities, relations, and events from unstructured text. In this dissertation, our focus is on extraction of events which has a wide range of applications from question answering [93], knowledge base construction [89] and named entity recognition [77] to informing critical decisions in domains ranging from national security to cyber security [91]. We leverage the recent advancements in the field of machine learning and propose linguistically motivated solutions for detecting and extracting events. We first begin by reviewing in brief, the history

of event detection and extraction, followed by term definitions that will be used throughout this dissertation. We then identify the failure modes and shortcomings of the current state-of-the-art event extraction systems. Next, we motivate and define three problems to address some of these shortcomings. The rest of the dissertation addresses these problems.

1.2 Event Extraction Overview

Modern computational event extraction(EE) has its roots in efforts stemming from the information extraction(IE) task which is defined as the automatic identification and classification of instances of user-specified types of entities, relations, and events from text. The output is a structured database. Specifications are either examples or verbal descriptions of the information to be extracted. Since the information is restricted to specific individuals or specific events this is termed as closed domain IE. This is distinguished from open domain IE [3, 63] where normally the focus is on extracting two entities and a relationship between them. E.g. given a sentence 'Teachers protest against government' open domain IE systems aim to extract a tuple such as; (Teachers (entity), government (entity), protest against (relationship).

Early evaluation efforts for information extraction were led by US government via MUC(Message Understanding Conference) [85] which began in 1988. MUCs involved filling a number of slots on predefined templates for a few topics such as Navy exercise message traffic(MUC 1,2), Terrorism in North America(MUC 3,4) joint ventures(MUC 5), microelectronics(MUC 5), executive succession(MUC 6), rocket launches(MUC 7). Systems were judged on how accurately they filled these templates and F-measure was used for system evaluation. However, with growing number of slots and growing complexity of systems in order to fill these monolithic templates ACE [24] was started in 2001 whose focus was extracting a set of

elementary events and their arguments. The goal of ACE was to develop technology to automatically infer from human language data the entities being mentioned, the relations among these entities, and the events in which these entities participate. There are seven types of entities, six types of relations, and eight types of events (33 event subtypes) in the the ACE-2005 dataset which includes six different document categories: newswire, broadcast news, broadcast conversation, weblog, usenet, and conversational telephone speech. Each document in the corpus belongs to one of them. Relations are binary; events may have any number of arguments. With minor exceptions, arguments must be entities or temporal expressions. ACE made a sizeable investment in corpus annotation to support supervised training of extraction systems. New corpora were released annually. The largest, for ACE 2005, was 300,000 words of English and comparable amounts of Chinese and Arabic. Event extraction involved the interaction of the trigger (the principal word defining the event) and multiple arguments. It is consequently a structured prediction task. The simplest solution is to decide first on the type of event, if any, and then to analyze the arguments [1]. Systems taking such an approach are termed as pipeline systems. However, pipeline systems lose considerable accuracy because for many common verbs their meaning depends on the arguments it takes. A better solution is to use joint inference: optimize for a combination of label choices if these choices interact.

ACE was a success in terms of producing annotated corpora and research results, but it treated documents separately, whereas many realistic tasks involved large numbers of interrelated documents. Information about an individual may need to be pieced together from several documents. To address these questions, NIST (the US National Institute of Standards and Technology) organized the annual "Text Analysis Conference" and its central task, "Knowledge Base Population" (KBP) [34]. KBP participants were tasked with building a graph from a large collection of unannotated documents. Each node in the graph represented

an individual, organization, GPE (Geo-Political Entity), location, or facility mentioned in the test collection. Associated with each type of node were a set of properties whose value could be a number, a date, a string, or another node in the network. In addition, participants had to link the entities to the arguments of events appearing in the test collection. Compared to ACE, the test corpora were about two orders of magnitude larger and since it was unannotated scoring was done via sampling i.e. NIST selected some names mentioned in the test corpus and checked whether (1) the system had created a node for this name and (2) the node had the desired property. Large volumes of unannotated data lack of annotated training encouraged experimentation with semi-supervised methods—learning from partially labeled data.

With advances in deep learning and ability to scale neural network training IE community embraced deep learning like a lot of other areas of NLP. Event extraction can involve multiple interactions which may benefit from joint inference. These interactions can be captured directly by a neural network through a set of "memory matrices" whose values are assigned as part of the network training and then used for event trigger and argument prediction [69]. These methods used static word embeddings [66] which weren't capable of capturing sense distinctions. With introductions of contextual word embeddings such as Peters et al. [72] event extraction performance further improved [59]. Traditional approaches to the task of ACE event extraction usually depend on manually annotated data, which is often laborious to create and limited in size. Therefore, in addition to the difficulty of event extraction itself, insufficient training data hinders the learning process as well. With the success of large pretrained language models [17, 48, 57] on a variety of downstream NLP tasks most modern approaches to event extraction have adopted them in their model architectures [14, 61, 88].

Besides broadcast and newswire other domains in which event extraction has be studied include – Biology, Finance, Cyber security, Politics etc. In the Biology domain, GENIA

event extraction is a main task in the BioNLP Shared Task which focuses on events relevant to protein biology, and it defines 9 event types in BioNLP 2009 [40, 41] and 13 event types in BioNLP 2013 [41]. In the finance domain, event extraction has mostly concerned with extracting descriptions of events pertaining to a specific company, and analyzed how such events correlate with measures of that company's stock (price, volatility etc.) [22, 23, 94].

1.2.1 Event Coding

A closely related task to event extraction is event coding. Event coding refers to automated production of high-volume, near-real-time political event data [50, 79]. Event coding is more focused, usually discussed in the context of political science, social science, conflict and international crises domains and is real-time. Event coding makes use of the event extraction technology to populate event knowledge bases by extraction and mapping to a predefined taxonomy such as CAMEO (Conflict and Mediation Event Observations) [1]. State-of-the-art event coders use statistical as well as linguistic and lexicographical approaches for event extraction and include encoders such as TABARI [2] (Textual Analysis by Augmented Replacement Instructions)[74], BBN's SERIF (Statistical Entity and Relation Information Finder) [8] and more recently Petrarch [3] which is a parser-based coder, successor to TABARI which was pattern-based.

1.2.2 Related Tasks

One of the most exciting applications of event coding is event forecasting. EMBERS [76] is an event forecasting system that ingests data from various sources such as social media

[1] http://eventdata.parusanalytics.com/data.dir/cameo.html
[2] http://eventdata.parusanalytics.com/software.dir/tabari.info.html
[3] http://eventdata.parusanalytics.com/tabari.dir/petrarch.html

(Twitter's public API, Facebook's event pages), traditional media(RSS news and blog feeds) and other sources such as Healthmap's alerts and reports,Talkwalker alerts, NASA satellite meteorological data, Google Flu Trends etc. and predicts civil unrest in the future. Civil unrest as defined by Ramakrishnan et al. [76] is a large concept intended to capture the myriad ways in which people express their protest against things that affect their lives and for which they assume that the government (local, regional or national) has a responsibility (e.g., cost of urban transportation, poor infrastructure, etc.). One of the early steps in EMBERS is event extraction. The extraction system in EMBERS is a multi-step process and deviates from the traditional extraction followed by mapping to a predefined ontology approach. The extraction model in EMBERS aims at detecting civil unrest events from traditional media and from social media. The model filters the input streams by matching to a custom multi-lingual lexicon of expressions such as preparaci´on huelga, llam´o a acudir a dicha movilizaci´on or plan to strike which are likely to indicate a planned unrest event. The event type and population are forecast using a multinomial naive Bayes classifier and the location information is determined using the enrichment geocoders.

1.3 Definition of Event

In this dissertation, our central goal is to investigate novel data driven methods using state-of-the-art advances in machine learning to improve the event extraction technology as used in *event coding* which includes event detection and event extraction in the domains of civil unrest, insurgency, politics and crises. With this goal is mind, we define an event of interest as any significant societal event that has happened in the recent past and is reported in a national news paper of repute. Societal events of interest include military action, non-state

actor, civil unrest and political events including the ones defined in the CAMEO [4] ontology. An event record generally contains information regarding — who(Actor/Target), why, when, where and other event attributes (population group, protest type etc.).

We call any source of text from which an event is extracted as an '**event document**' or just document. In a typical document, not all sentences contain an event. Some sentences might just provide supporting information about the main topic of the topic. There exist, but a small set of key sentences that provide detailed information for a specific event. We define such key sentences as '**event extent**'. Each article typically also contains a small set of key words or phrases that invoke the event and that typically are contained within an event extent. We call these words as '**event trigger**' or just trigger for short. Event extents can form the basis of automated event encoding as we can extract the final event based on the identified salient sentences. An '**event argument**' is defined as a participant involved in an event that can be found in an event extent. In this dissertation specifically, we're interested in event arguments such as 'Actor', 'Target' of a political event. An '**event attribute**' is additional metadata about an event that can be inferred from certain words or phrases in the extents. For example, 'population type' and 'protest type' are attributes of a civil unrest event. Table 1.1 lists the key terms and their definitions.

1.4 Definition of Event Detection & Extraction

Before an event can be extracted it is important to detect which documents from a large collection of documents obtained contain an event of interest. Event analysis can be categorized as a hierarchical task where the coarser level task is event detection – identification of documents containing a specific event, and identifying event extents and the fine-grained task

[4] http://data.gdeltproject.org/documentation/CAMEO.Manual.1.1b3.pdf

Table 1.1: Event terms and their definitions as used throughout this dissertation.

Term	Definition
Event	A political occuring of importance pertaining to domains such as civil unrest, insurgency or crises.
Event Document	A text document whose origins can be traced to online news blogs, news articles or social media, that contains one or more events of interest.
Event Extent	A sentence from a document that contains an event of interest.
Event Trigger	A main word or phrase in text, typically a verbal or nominal one, which most clearly expresses an event and occurs in an event extent.
Event Argument	A participant or attribute in text, typically a noun or a noun phrase, which is involved in an event.
Event Attribute	Additional metadata about an event.
Event Mention	A clause in text that describes an event, and includes both an event trigger and arguments.

of event extraction – identifying event arguments. This process of classifying the documents as either containing an event of interest or not is termed as *event detection*. Additionally, extracting event extents and event triggers from those documents also falls under the umbrella of event detection. Extracting event arguments from event extents is termed as *event extraction* in this dissertation.

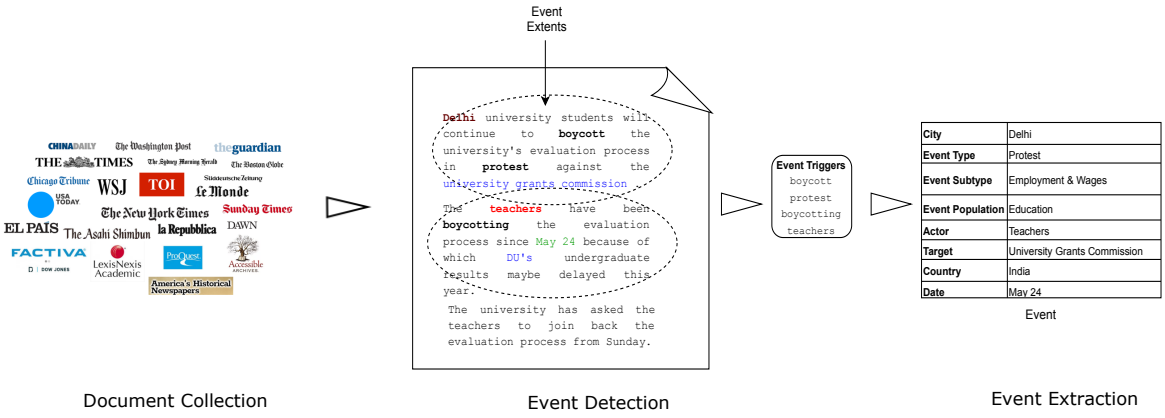


Figure 1.1: The process of event detection and extraction.

1.5 Limitations of Existing Systems

We motivate the contributions of this dissertation by shedding light on some limitations of existing systems for event detection and extraction.

1.5.1 Limitations of Existing Event Detection Systems

Event detection can be cast as a document classification problem when a corpora of labeled documents can be collected. Event extent extraction can be cast as a sentence classification problem if sentence-level labels are available. Most available event corpora contain fine-grained labels for a restricted domain [24]. Such fine-grained labeled training data is hard to obtain for a variety of domains and different types of events. On the other hand, labels at the document level are easier to obtain. Wang et al. [92] and Kotzias et al. [45] leverage this insight and have proposed instance-level Multiple Instance Learning (MIL) solution to partly alleviate the problems and relax the requirement of collecting sentence level labels.

MIL is a machine learning paradigm that concerns with classifying *bags* of *instances*. In the context of event detection and extraction a bag is a document which is a collection of instances, where an instance is a sentence. A bag can also refer to a sentence in which case it is a collection of words. Deep learning based MIL approaches can be categorized into two classes - instance-level and embedding-level [33]. In instance-level MIL a sentence-level classifier returns scores for each sentence and then individual scores are aggregated by MIL pooling to obtain the bag-level score. Instance-level MIL uses hard aggregation of instance-level probabilities using sentence embeddings to obtain a bag-level probability and hence doesn't take into account the complete context of the document resulting in erroneous predictions because event occurrences are not neatly partitioned into individual sentences. Hence, there is a need to explore other MIL approaches for the task. In embedding level MIL, sentences

are mapped to a low-dimensional embedding using a function f. MIL pooling is used to obtain a bag representation that is independent of the number of instances in the bag. The bag representation is further processed by a bag-level classifier to provide the bag probability. The MIL pooling operator can be defined using an attention mechanism [90]. Hence, event detection can be modeled as a hierarchical classification task using attention mechanism. We find that existing attention mechanisms [54, 96] are not the best choice for the event detection task. Simple dot-product attention mechanisms [96] are suboptimal whereas existing multi-head self-attention mechanisms [54, 87] which can learn richer representations contain too many parameters to be trained effectively in a hierarchical framework for event detection.

Another motivation for using attention-based methods for event detection is that systems trained on available labeled datasets such as ACE naturally cannot be scaled to real-time forecasting and coding systems such as EMBERS [76] and EMBERS AutoGSR [78]. Particularly, while developing hybrid coding systems such as AutoGSR one of the major considerations is to reduce the level of effort that analysts must place in steering such a human-in-the-loop system toward satisfactory performance. One way to do this is to provide explanations in terms of important subtexts useful in annotating a document. Such explanations can be used in various ways in an interactive system including human judgements to increase or decrease the importance of contextual patterns and using feedback to improve the annotations. For this purpose, we propose to utilize weakly-supervised feature importance methods such as attention [54, 60] as a tool towards providing these explanations since obtaining fine-grained supervision is an expensive endeavour.

Limitations of Existing Attention Mechanisms for Text Classification

In recent times, the most effective systems for NLP tasks such as machine translation [87], question answering [80] and text classification have attention as an essential component.

Moreover, with the success of self-supervised large pretrained models [17, 57] attention has established its place in state-of-the-art NLP systems. Although these models work well on a variety of tasks there are two major limitations: 1) they are computationally expensive to train and attention computation is a major bottleneck and 2) they usually have a large number of parameters that greatly increases the model size and memory requirements. For instance, the multilingual BERT-base-cased model has 110M parameters, the small GPT-2 model has 117M parameters [73] and the RoBERTa model was trained on 160GB of data [57]. It is natural to see how task specific training (ELMo) or fine-tuning (BERT, GPT-2) can be limiting when the training data and computational resources are scarce. Further, running inference on and storing such models can also be difficult in low resource scenarios such as IoT devices or low-latency use cases. Hence, supervised learning for task-specific architectures which are trained from scratch, especially where domain specific training data is available are useful. They are light-weight and easy to deploy. With this motivation, we focus on learning compact attention-based supervised language representations with text classification as the downstream task.

1.5.2 Limitations of Existing Event Extraction Systems

Current State of Event Extraction

Prior event extraction work [13, 68] has focused on extracting entities, detecting trigger terms (or keywords), and matching up event slots on such predefined templates. As discussed earlier(§ 1.2), approaches to event extraction can be categorized into two - (i) the pipeline approach that first performs trigger prediction and then identifies arguments in separate stages and (ii) the joint approach where event extraction is formulated as a classification problem, aiming to locate and categorize each event trigger/argument [13, 51, 69]

simultaneously via a sequence-labeling approach using either structured prediction models such as CRFs [47] or more recently BERT [17]. A lot of efforts have gone into the joint modeling approach because of error propagation problem with the pipeline approach. For example, Liu et al. [56] propose a framework to jointly extract multiple event triggers and arguments by introducing syntactic shortcut arcs to enhance information flow between multiple event occurrences with a sentence. Despite many advances, classification based methods are data-hungry, and require a great deal of training data to ensure good performance [12, 51, 56]. Moreover, such methods generally cannot deal with new event types never encountered during train time [32]. Generating ground truth data for new event types can be expensive and time consuming. A variety of event domains, types and definitions combined with the multiple sources of data, scarcity of fine-level labels and unstructured data makes this task challenging.

Difficulty of Extending Event Extraction to New Domains and Emergence of a New Paradigm

In recent years, with the success of large pretrained models for the tasks of reading comprehension [17, 57] reading-comprehension (MRC) based approaches for event extraction have begin to surface [26, 55]. This approach models event extraction as a MRC task where a question is generated for each argument to be extracted for a detected event and given the event extent an MRC system generates an answer. This paradigm for event extraction, besides alleviating the need for pipeline extraction systems that rely on other upstream systems to extract entities/triggers and hence sidestepping the error propagation problem also gives rise to the very promising possibility of zero-shot event extraction.

Zero-Shot Event Extraction

In zero-shot event extraction, at test time, the input to the MRC system are questions about arguments previously unseen during train time [26]. However, in most research on zero-shot event extraction [26, 32] both the seen and unseen roles are from the same domain and the MRC-systems still need domain specific data to train. Another challenge with the MRC-based approach is the task of generation of questions for each event argument. Recent studies have proposed approaches for question generation [26, 55] although investigations are still in a very nascent stage. A more practical zero-shot scenario is by leveraging MRC-systems trained on large publicly available corpora such as SQUAD [75] which is albeit in a different domain. This is called cross-domain data augmentation. We premise that zero-shot event extraction using cross-domain data augmentation presents a promising direction towards scaling event extraction to new domains.

The Long-Range Dependency Problem

A recurring problem observed by event extraction studies is capturing the long-range dependency, specifically, the connection between an event trigger and a distant event argument. Efforts have been made to incorporate syntactic dependencies into the models in an effort to mitigate this problem [56, 61, 82]. The drawback of long range dependencies has been shown to strung-along MRC-based approaches as well, specifically their vulnerability to syntactically complex sentences, containing multiple clauses and each clause containing event arguments [26, 55]. This raises the question of how can we use syntactic information to guide the MRC models to correctly answer the generated questions.

1.6 Goals and Contributions

To address the above challenges, in this dissertation, we focus on three problems in obtaining event-related information from news articles. (1) In the first problem we perform an analysis of existing weakly-supervised feature importance methods such as attention for general text classification tasks and find that existing (esp. multi-head attention mechanisms) which can be used as a useful tool for event detection are parameter inefficient and can hinder the application of such methods at scale and in low-resource scenarios. We propose a new and efficient multi-head attention mechanism for general text classification tasks; (2) In the second problem, we aim to answer the question - are existing MIL-based methods effective for event detection and extraction? We use the proposed attention mechanism in a hierarchical framework for event detection that we present which can also be used to extract event extents and triggers (3) In the last problem we study ways to enhance zero-shot event extraction by using syntactic sentence simplification.

1.6.1 Compact Multi-Head Self-Attention for Text Classification

First, we focus on improving methods for text classification. We will later use these methods for event detection task. In particular, we develop a novel multi-head self-attention mechanism, LAMA, for learning text representations. We show that – (1) the proposed mechanism is more effective than existing single-head mechanisms, (2) the proposed approach is more parameter efficient than existing multi-head approaches. We first establish the effectiveness of multi-head attention mechanisms over single-head attention mechanisms for certain text classification tasks. We then address the computational inefficiency of existing multi-head attention mechanisms by proposing a novel multi-head attention mechanism that is computationally cheaper than existing approaches. We then verify the effectiveness of the proposed

approach on several text classification benchmarks.

1.6.2 Event Detection using Hierarchical Multi-Head Attention

Event detection as defined earlier is the task of identifying documents likely to contain an event from a collection of documents. It also entails identifying event extents and event triggers. If labels for documents and finegrained labels for extents and triggers are available, these can be formulated as classification tasks. However, finegrained labels for event extents and triggers are not available and only event labels at the document level are assumed. Existing event detection systems [92] use instance-level MIL, that first obtain sentence(instance) embeddings via a convolutional neural network. Sentence-level probabilities are predicted by sentence-level classifiers. Document probability of containing an event is estimated by aggregating sentence-level probabilities. However, since labels are available only at the document level sentence-level classifiers might be insufficiently trained resulting in the error propagation problem.

To counteract the problem, we formulate event detection as an embedding-level MIL task. We consider a sentence as a bag-of-words and compute a sentence representation by leveraging the proposed attention mechanism LAMA as the MIL pooling operator. We show that using single-head mechanisms [96] can be suboptimal and using existing multi-head attention mechanisms [53] can result in poor performance since these mechanisms contain a large number of parameters which can be insufficiently trained. Finally, different from previous methods, instead of estimating document level probabilities from sentence level probabilities we compute a document representation via weighted aggregation of sentence embeddings. The weights are given by another attention mechanism. This helps compute rich document representations that not only contain information from event extents but also possibly from

surrounding sentences which is useful because the assumption that events will be neatly partitioned into individual sentences might not always hold true.

1.6.3 Zero-Shot Event Extraction using Sentence Simplification

Recent work has proposed to model event extraction as a reading comprehension (MRC) task and shown promising results [26, 55]. One of the advantages of using this formulation is the ability to extract previously unseen events (zero-shot extraction). However, most previous approaches addressing zero-shot event extraction [26, 32] consider an in-domain setting, where training data is assumed for a given domain and a few event types in the same domain are unseen during test time. A more practical setting is in which one leverages MRC systems based on large pretrained language models [18, 57] finetuned on large-scale publicly available MRC corpora [75]. However MRC systems stumble when context is complex for example when containing relative clauses and appositives or when there are long range dependencies between an argument and a trigger a.k.a the long range dependency problem. How can we guide the MRC systems when sentences are syntactically complex? We show how syntactic properties of language can be leveraged to simplify context guided by the MRC systems. Simplification candidates are generated by pruning constituency parse trees to discard vestigial subtrees and unimportant parts of the sentence such as relative clauses and appositives. A comprehensive score function is designed that takes into account MRC-system's feedback, a fluency score by a syntactic language model, and other components to guide candidate selection. In each iteration a higher scoring candidate is selected until it is no longer possible to generate a better candidate.

Chapter 2

Attention for Text Classification

In this chapter, we perform a comprehensive analysis of existing attention mechanisms on general text classification tasks and analyse the effectiveness and efficiency of different type of attention mechanisms such as single-head and multi-head. We propose a new way of computing multi-head attention which we show to be more parameter efficient (by an order of magnitude) and equally or more effective than existing state-of-the-art multi-head self-attention mechanisms.

2.1 Introduction

Learning effective language representation is important for a variety of text analysis tasks including sentiment analysis, news classification, natural language inference and question answering. Learning language representations has made substantial progress in recent years with the introduction of new techniques for language modeling combined with deep models like BERT [18], GPT-2 and GPT-3 [9, 73]. These methods have enabled transfer of learned representations via self-supervised pre-training to downstream tasks. Although these models work well on a variety of tasks there are two major limitations: 1) they are computationally expensive to train and 2) they usually have a large number of parameters that greatly increases the model size and memory requirements. For instance, the multilingual BERT-base cased model has 110M parameters, the small GPT-2 model has 117M parameters [73]

and the RoBERTa model was trained on 160GB of data [57]. It is natural to see how task specific training (ELMo [72]) or fine-tuning (BERT, GPT-2) can be limiting when the training data and computational resources are scarce. Further, running inference on and storing such models can also be difficult in low resource scenarios such as IoT devices or low-latency use cases. Hence, supervised learning for task-specific architectures which are trained from scratch, especially where domain specific training data is available are useful. They are light-weight and easy to deploy. Supervised learning using neural networks commonly entails learning intermediate sentence representations usually using an attention mechanism followed by a task specific layer. For text classification tasks; this is usually a fully connected layer followed by an *N-way* softmax layer where N is the number of classes. In this chapter, we focus on learning compact attention-based supervised language representations with text classification as the downstream task.

Computation at attention layers of modern neural networks can get prohibitive. Especially, multi-head self-attention mechanisms [29, 54, 87] (multiple attention distributions over a given sentence) that form an integral part of many state-of-the-art architectures for NLP tasks [18, 87] can be expensive to compute. We argue that the attention layer giving rise to multiple attentions in these methods is over-parameterized. In this work, we propose a novel **l**ow-rank f**a**ctorization based **m**ulti-head **a**ttention mechanism (LAMA), which is computationally cheaper than prior approaches and sometimes exceeds the performance of state-of-the-art baselines.

Contrary to previous approaches [29, 54] that are based on the additive attention mechanism [5], LAMA is based on the multiplicative attention [60] which replaces the additive attention by the dot product attention for faster computation. We further introduce a bi-linear projection while computing the dot product to capture similarities between a global context vector and each word in the sentence. The function of the bilinear projection is to

capture nuanced context dependent word-importance as corroborated by previous works [12]. Next, we use a low-rank formulation of the bilinear projection matrix based on hadarmard product [42, 97] to compress the attention layer and speed up the computation of multiple attention distributions for each word. We leverage this approach to decompose a single bilinear matrix to produce multiple attentions between the global context vector and each word as opposed to having a different learned vector [54] or matrix [98]. We evaluate our model by performing experiments on multiple datasets spanning two different tasks namely Sentiment Analysis and Text Classification. We find that our model restores or in some cases outperforms state-of-the-art approaches with fewer parameters for efficient computation.

We organize the rest of the discussion as follows. In the next section we describe the proposed model (§2.3). Next we describe the experiments and the datasets (§2.4), followed by results and discussion (§2.5), before concluding (§2.7).

2.2 Background

Multimodal tasks such as image captioning [52], visual reasoning [35] and visual question answering [97] (VQA) are challenging because they require simultaneous processing and modeling of interactions between high dimensional multimodal inputs. The approaches used to represent the images and text in a fine-grained manner and to fuse these multimodal features play a key role in performance. Bilinear pooling based models [86] have been shown to outperform traditional linear models (e.g., concatenation or element-wise addition) for VQA. However, these approaches are computationally intensive with a huge number of model parameters that may seriously limit the applicability of bilinear pooling. Approaches such as MCB [27], MLB [39] and MFB [98] have been proposed to make this operation more computationally efficient.

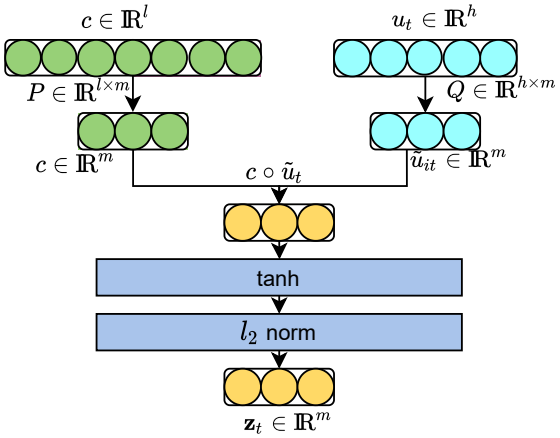


Figure 2.1: LAMA as unimodal feature fusion. Each dimension in the output z represents a score between context c and an input term u_t

Bilinear pooling has been less explored in modeling text to text interactions. One instance is when Chen et al. [12] use a bilinear term to capture the similarity between each paragraph term and the question for open-domain question answering.

The proposed approach, LAMA shown in Fig. 2.1, can be viewed as unimodal pooling i.e. fusing feature vectors from two text sources. Each dimension in the fused output can be viewed as a separate score between a global context vector c (input 1) and a term in the text u_t (input 2).

2.3 Methods

A document (review or a news article) is first tokenized and converted to a word embedding via a lookup into a pretrained embedding matrix. The embedding of each token is encoded

via a bi-GRU sentence encoder to get a contextual annotation of each word in that sentence. The LAMA attention mechanism then obtains multiple attention distributions over those words by computing an alignment score of their hidden representation with a word-level context vector. Sum of the word representations weighted by the scores from multiple attention distributions then forms a matrix sentence embedding. The matrix embedding is then flattened and passed onto downstream layers (either a classifier or another encoder depending on the task). Since we model all tokens in the text together without using any hierarchical structure, without loss of generality the terms sentence and document are used interchangeably in the rest of the chapter. Upper-case bold letters indicate matrices, lower-case bold letters indicate vectors and lower-case letters indicate scalars. Please use table 2.1 to cross-reference the important notations in this chapter.

2.3.1 Sequence Encoder

We use the GRU [5] RNN as the sequence encoder. GRU uses a gating mechanism to track the state of the sequences. There are two types of gates: the reset gate $\mathbf{r}_t$ and the update gate $\mathbf{z}_t$. The update gate decides how much past information is kept and how much new information is added. At time t, the GRU computes its new state as:

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}} \tag{2.1}$$

and the update gate $\mathbf{z}_t$ is updated as:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z * \mathbf{x}_t + \mathbf{U}_z * \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{2.2}$$

The RNN candidate state $\tilde{\mathbf{h}_t}$ is computed as:

$$\tilde{\mathbf{h}_t} = tanh(\mathbf{W}_h\mathbf{x}_t + \mathbf{r}_t \odot (\mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.3}$$

Here $\mathbf{r}_t$ is the reset gate which controls how much the past state contributes to the candidate state. If $\mathbf{r}_t$ is zero, then it forgets the previous state. The reset gate is updated as follows:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r) \tag{2.4}$$

Consider a document D_i containing T words. $D_i = \{\mathbf{w}_1, ..., \mathbf{w}_t, ..., \mathbf{w}_T\}$. Let each word be denoted by $\mathbf{w}_t$, $t \in [0, T]$ where every word is converted to a real valued word vector $\mathbf{x}_t$ using the pre-trained embedding matrix $\mathbf{W}_e = R^{d\times|V|}$, $\mathbf{x}_t = \mathbf{W}_e\mathbf{w}_t$, $t \in [1, T]$ where d is the embedding dimension and V is the vocabulary. The embedding matrix $\mathbf{W}_e$ is fine-tuned during training. Note that we have dropped the subscript i as all the derivations are for the i^{th} document and it is assumed implicit in the following sections.

We encode the document using a bi-directional GRU (bi-GRU) that summarizes information in both directions along the text to get a contextual annotation of a word. In a bi-GRU the hidden state at time step t is represented as a concatenation of hidden states in the forward and backward direction. The forward GRU denoted by $\overrightarrow{GRU}$ processes the sentence from w_1 to w_T whereas the backward GRU denoted by $\overleftarrow{GRU}$ processes it from w_T to w_1.

$$\mathbf{x}_t = \mathbf{W}_e\mathbf{w}_t \tag{2.5}$$

$$\overrightarrow{\mathbf{h}_t} = \overrightarrow{GRU}(\mathbf{x}_t, \mathbf{h}_{(t-1)}, \boldsymbol{\theta}) \tag{2.6a}$$

$$\overleftarrow{\mathbf{h}_t} = \overleftarrow{GRU}(\mathbf{x}_t, \mathbf{h}_{(t+1)}, \boldsymbol{\theta}) \tag{2.6b}$$

Here the word annotation $\mathbf{h}_t$ is obtained by concatenating the forward hidden state $\overrightarrow{\mathbf{h}_t}$ and the backward hidden state $\overleftarrow{\mathbf{h}_t}$.

2.3.2 Single-Head Attention

To alleviate the burden of remembering long term dependencies from GRUs we use the global attention mechanism [60] in which the sentence representation is computed by attending to all words in the sentence. Let $\mathbf{h}_t$ be the annotation corresponding to the word $\mathbf{x}_t$. First we transform $\mathbf{h}_t$ using a one layer Multi-Layer Perceptron (MLP) to obtain its hidden representation $\mathbf{u}_t$. We assume Gaussian priors with 0 mean and 0.1 standard deviation on $\mathbf{W}_w$ and $\mathbf{b}_w$.

$$\mathbf{u}_t = tanh(\mathbf{W}_w \mathbf{h}_t + \mathbf{b}_w) \tag{2.7}$$

Next, to compute the importance of the word in the current context we calculate its relevance to a global context vector $\mathbf{c}$.

$$f_t = \mathbf{c}^T \mathbf{W}_i \mathbf{u}_t \tag{2.8}$$

Here, $\mathbf{W}_i \in \mathbb{R}^{2h \times 2h}$, is a bilinear projection matrix which is randomly initialized and jointly learned with other parameters during training. h is the dimension of the GRU hidden state and $\mathbf{u}_t$ & $\mathbf{c}$ are both of dimension $2h \times 1$ since we're using a bi-GRU. The mean of the word embeddings provides a good initial approximation of the global context of the sentence. We initialize $\mathbf{c} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$ which is then updated during training. We use a bilinear model because they are more effective in learning pairwise interactions. The attention weight for the word $\mathbf{x}_t$ is then computed using a *softmax* function where summation is taken over all the words in the document.

$$\alpha_t = \frac{exp(f_t)}{\sum_{t'} exp(f_{t'})} \tag{2.9}$$

Table 2.1: Important notations. Unless specified bold upper-case letters indicate matrices, bold lower-case letters indicate vectors, unbolded lower-case letters indicate scalars in this dissertation.

Notation	Meaning
N	Corpus size
T	# of words tokens in a sample
m	# of aspects
f_t	alignment score
α_t	attention weight
$\mathbf{u}_t$	word hidden representation
$\mathbf{c}$	global context vector
h	GRU hidden state dimension

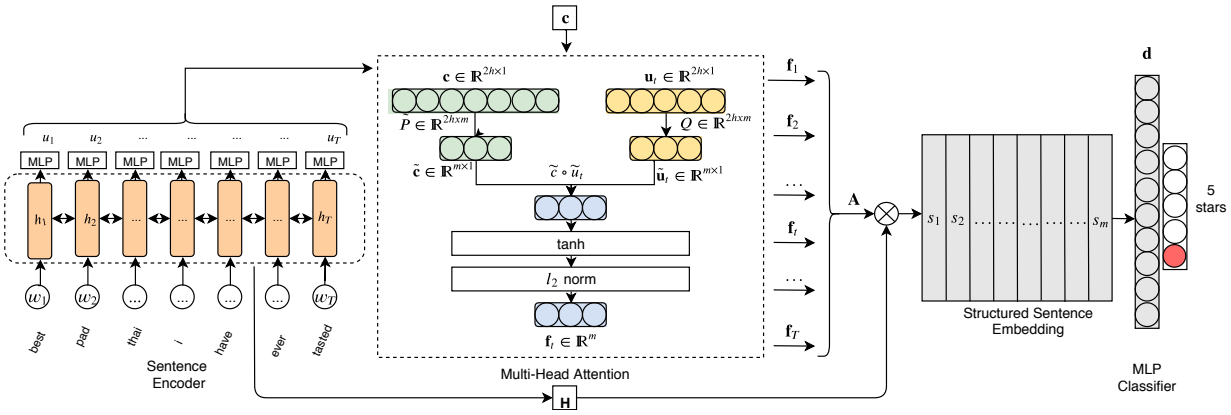


Figure 2.2: Figure describes a schematic of the model architecture and its major components including the Sentence Encoder, proposed multi-head attention mechanism LAMA, Structured Sentence Embedding and finally the MLP classifier. The attention computation is demonstrated for a single word.

2.3.3 LAMA

The attention distribution above usually focuses on a specific component of the document, like a special set of trigger words. So it is expected to reflect a component of the semantics in a document. This type of attention is useful for smaller pieces of texts such as tweets or short reviews. For larger reviews there can be multiple semantic components that describe that review. For this we introduce a novel way of computing multiple heads of attention that capture different aspects.

Suppose m heads are to be extracted from a sentence, we need m alignment scores between each word hidden representation $\mathbf{u}_t$ and the context vector $\mathbf{c}$. To obtain an m dimensional output $\mathbf{f}_t$, we need to learn m weight matrices given by $\mathbf{W} = [\mathbf{W}_1, ..., \mathbf{W}_m] \in \mathbf{R}^{m \times 2h \times 2h}$ as demonstrated in previous works. Although this strategy might be effective in capturing pairwise interactions for each head it also introduces a huge number of parameters that may lead to overfitting and also incur a high computational cost especially for a large m or a large h. To address this, the rank of matrix $\mathbf{W}$ can be reduced by using low-rank bilinear method to have less number of parameters [42, 98]. Consider one head; the bilinear projection matrix $\mathbf{W}_i$ in Eq. 2.8 is factorized into two low rank matrices $\mathbf{P}$ & $\mathbf{Q}$.

$$f_t = \mathbf{c}^T \mathbf{P} \mathbf{Q}^T \mathbf{u}_t = \sum_{d=1}^{k} \mathbf{c}^T p_d q_d^T \mathbf{u}_t = \mathbb{1}^T (\mathbf{P}^T \mathbf{c} \circ \mathbf{Q}^T \mathbf{u}_t) \tag{2.10}$$

where $\mathbf{P} = [p_1, ..., p_k] \in \mathbb{R}^{2h \times k}$ and $\mathbf{Q} = [q_1, ..., q_k] \in \mathbb{R}^{2h \times k}$ are two low-rank matrics, $\circ$ is the Hadamard product or the element-wise multiplication of two vectors, $\mathbb{1} \in \mathbb{R}^k$ is an all-one vector and k is the latent dimensionality of the factorized matrices.

To obtain m scores, by Eq.2.10, the weights to be learned are two three-order tensors $\mathbb{P} = [\mathbf{P}_1, ..., \mathbf{P}_m] \in \mathbb{R}^{2h \times k \times m}$ and $\mathbb{Q} = [\mathbf{Q}_1, ..., \mathbf{Q}_m] \in \mathbb{R}^{2h \times k \times m}$ accordingly. Without loss of generality $\mathbb{P}$ and $\mathbb{Q}$ can be reformulated as 2-D matrices $\tilde{\mathbb{P}} \in \mathbb{R}^{2h \times km}$ and $\tilde{\mathbb{Q}} \in \mathbb{R}^{2h \times km}$ respectively with simple reshape operations. Setting $k = 1$, which corresponds to rank-1 factorization. Eq.2.10 can be written as:

$$\mathbf{f}_t = \tilde{\mathbb{P}}^T \mathbf{c} \circ \tilde{\mathbb{Q}}^T \mathbf{u}_t \tag{2.11}$$

This brings the two feature vectors $\mathbf{u}_t \in \mathbf{R}^{2h}$, the word hidden representation and $\mathbf{c} \in \mathbf{R}^{2h}$, the global context vector in a common subspace and are given by $\tilde{\mathbf{u}}_t$ and $\tilde{\mathbf{c}}$ respectively.

$\mathbf{f}_t \in \mathbb{R}^m$ now is a multi-head alignment vector for the word $\mathbf{x}_t$. For computing attention for one head, this is equivalent to replacing the projection matrix W_i in Eq 2.8 by the outer product of vectors $\tilde{\mathbb{P}}_i$ and $\tilde{\mathbb{Q}}_i$ - rows of the matrices $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{Q}}$ respectively and rewriting it as the Hadamard product. As a result each row of matrices $\tilde{\mathbb{P}}_i$ and $\tilde{\mathbb{Q}}_i$ represent the vectors for computing the score for a different head.

The multi-head attention vector $\boldsymbol{\alpha}_t \in \mathbb{R}^m$ is obtained by computing a softmax function along the sentence length:

$$\boldsymbol{\alpha}_t = \frac{exp(\mathbf{f}_t)}{\sum_{t'} exp(\mathbf{f}_{t'})} \tag{2.12}$$

Before computing softmax, similar to [42, 98] to further increase the model capacity we apply the *tanh* nonlinearlity to $\mathbf{f}_t$. Since element-wise multiplication is introduced the values of neurons may vary a lot so we apply an l_2 normalization layer across the m dimension. Although l_2 is not strictly necessary since both $\mathbf{c}$ and $\mathbf{u}_t$ are in the same modality empirically we do see improvement after applying l_2. Each component k of $\boldsymbol{\alpha}_t$ is the contribution of the word $\mathbf{x}_t$ to the k^{th} head.

Let $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ...\mathbf{h}_T)$ be a matrix of all word annotations in the sentence; $\mathbf{H} \in \mathbb{R}^{T \times 2h}$. The attention matrix for the sentence can be computed as:

$$\mathbf{A} = softmax(l2(tanh(\tilde{\mathbb{P}}^T \mathbf{C}_g \circ \tilde{\mathbb{Q}}^T \mathbf{H}^T))) \tag{2.13}$$

where, $\mathbf{C}_g \in \mathbb{R}^{2h \times T}$ is $\mathbf{c}$ repeated T times, once for each word, $l_2(\mathbf{x}) = \frac{\mathbf{x}}{||\mathbf{x}||}$ and *softmax* is applied row-wise. $\mathbf{A} \in \mathbb{R}^{m \times T}$ is the attention matrix between the sentence and the global context with each row representing attention for one head.

Given $\mathbf{A} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ...\boldsymbol{\alpha}_T]$, the multi-head attention matrix for the sentence; $\mathbf{A} \in \mathbb{R}^{m \times T}$.

The sentence representation for an aspect j given by $\boldsymbol{\alpha}_j = \{\alpha_{j1}, \alpha_{j2}, ..\alpha_{jT}\}$ can be computed by taking a weighted sum of all word annotations.

$$\mathbf{s}_j = \sum_{k=1}^{T} \mathbf{h}_k * \alpha_{jk} \tag{2.14}$$

Similarly, sentence representation can be computed for all heads and is given in a compact form by:

$$\mathbf{S} = \mathbf{AH} \tag{2.15}$$

Here $\mathbf{S} \in \mathbb{R}^{m \times 2h}$ is a matrix sentence embedding and contains as many rows as the number of heads. Each row contains an attention distribution for a new aspect. It is flattened by concatenating all rows to obtain the document representation $\mathbf{d}$. From the document representation, the class probabilities are obtained as follows.

$$\hat{y} = softmax(\mathbf{W}_c\mathbf{d} + \mathbf{b}_c) \tag{2.16}$$

Loss is computed using cross entropy.

$$l = -\sum_{c=1}^{C} y_c log(\hat{y_c}) \tag{2.17}$$

where C is the number of classes and $\hat{y_c}$ is the probability of the class c. The final training loss is given by:

$$L = \sum_{d} l \tag{2.18}$$

The summation is taken over all the documents in a mini-batch. We use the mini-batch stochastic gradient descent algorithm [38] with momentum and weight decay for optimizing the loss function and the backpropogation algorithm is used to compute the gradients. Fig. 2.2 illustrates a schematic of the model architecture. A single document and its flow through

Table 2.2: Dataset statistics. # words indicates the average number of tokens per document in the corresponding datasets.

Dataset	# Classes	# Train	# Test	# words
YELP	5	499,976	4,000	118
YELP-L	5	175,844	1,378	226
YELP-P	2	560,000	38,000	137
IMDB	2	25,000	25,000	221
Reuters	8	4,484	2,189	102
News	4	151,328	32,428	352

various model components is shown. The middle block illustrates the proposed attention mechanism for one word $\mathbf{w}_t$ of the document.

Hyperparameters

We use a word embedding size of 100. The embedding matrix $\mathbf{W}_e$ is pretrained on the corpus using word2vec. All words appearing less than 5 times are discarded. The GRU hidden state is set to $h = 50$, MLP hidden state to 512 and apply a 0.4 dropout to the hidden layer. We use a batch size of 32 for training and an initial learning rate of 0.05. For early stopping we use $patience = 5$.

2.4 Experiments

We evaluate the performance of the proposed model on two tasks with six different datasets. Table 2.2 gives an overview of the datasets and their statistics.

2.4.1 Sentiment Analysis

For the sentiment analysis task we chose two datasets - the YELP Ratings dataset and the IMDB Movie Sentiment Dataset.

Yelp The Yelp dataset [1] consists of 2.7M Yelp reviews and user ratings from 1 to 5. Given a review as input the goal is to predict the number of stars the user who wrote that review assigned to the corresponding business store. We treat the task as 5-way text classification where each class indicates the user rating. We randomly selected 500K review-star pairs as training set, 4,000 for the dev set, and 4,000 for test set. Reviews were tokenized using the Spacy tokenizer [2]. 100-dimensional word embeddings were trained from scratch on the train dataset using the gensim [3] software package.

Yelp-Long Multi-head attention capturing multiple heads is more useful for classifying ratings that are more subjective i.e. longer reviews where people express their experiences in detail. We create a subset of the YELP dataset containing all longer reviews i.e. reviews containing longer than 118 tokens which we found to be the mean length of the reviews in the dataset. The training set consists of 175,844 reviews, the dev set consists of 1,416 reviews and the test set consists of 1,378 reviews. The goal is to predict the ratings from the above subset of the Yelp dataset. We refer to this dataset as Yelp-L (Yelp-Long) since it consists of all longer reviews. We hypothesize that having multi-head attention would benefit in this setting where more intricate foraging of information from different parts of the text is required to make a prediction. The model hyperparameters and training settings remain the same as the above.

[1] https://www.yelp.com/dataset challenge
[2] https://spacy.io/
[3] https://radimrehurek.com/gensim/

Yelp-Polarity The Yelp reviews polarity dataset [99] is constructed by considering stars 1 and 2 negative, and 3 and 4 positive from the Yelp dataset. For each polarity 280,000 training samples and 19,000 testing samples are taken randomly. In total there are 560,000 training samples and 38,000 testing samples. This datset is referred as Yelp-P in the paper.

Movie Reviews The large Movie Review dataset [62] contains movie reviews along with their associated binary sentiment polarity labels. It contains 50,000 highly polar reviews ($score \leq 4$ out of 10 for negative reviews and $score \geq 7$ out of 10 for positive reviews) split evenly into 25K train and 25K test sets. The overall distribution of labels is balanced (25K positive and 25K negative). In the entire collection, no more than 30 reviews are allowed for any given movie because reviews for the same movie tend to have correlated ratings. Further, the train and test sets contain a disjoint set of movies, so no significant performance is obtained by memorizing movie-unique terms and their associated with observed labels. From the training set we randomly set aside 1000 reviews for validation. We refer to this dataset as IMDB in the rest of the Chapter.

2.4.2 News Classification

For this task we use two datasets News Aggregator and Reuters.

News Aggregator This dataset [19] contains headlines, URLs, and categories for news stories collected by a web aggregator between March 10th, 2014 and August 10th, 2014. News categorsies included in this dataset include business; science and technology; entertainment; and health. Different news articles that refer to the same news item (e.g., several articles about recently released employment statistics) are also categorized together. Given a news article the task is to classify it into one of the four categories. Training dataset consists of

151,328 articles and test dataset consists of 32,428. The average token length is 352.

Reuters This dataset [4] is taken from Reuters-21578 Text Categorization Collection. This is a collection of documents that appeared on Reuters newswire in 1987. The documents were assembled and indexed with categories. We evaluate on the Reuters8 dataset consisting of news articles about 8 topics including acq, crude, earn, grain, interest, money-fx, ship,trade.

2.4.3 Comparative Methods

Table 2.3 shows the result of ablation experiments with and without LAMA. We use a bi-GRU [15] model with maxpooling referred as BiGRU as a baseline i.e. first we encode the given text with a bi-GRU and then take max over all hidden states of the GRU over each hidden state dimension giving us a fixed sized embedding. We use a 2-layer MLP classifier for prediction.

Next, we compare LAMA against strong established baselines. The first baseline method uses a simple average of word embeddings (AVG) [83] to get a fixed dimensional embedding followed by an MLP classifier layer. The recently proposed BERT model [18] is used as another baseline. We use a pretrained BERT implementation [5] and finetune it with a task-specific classifier. This finetuning is performed for 10 epochs using the ADAM optimizer [44] with a learning rate of 5e-6.

Next, we use convolutional neural network with *max-over-time* pooling [43] as another baseline. We refer to this as CNN. We use three kernels of size 3,4 & 5 with 100 filters each.

Among attention-based multi-head models we use the Self Attention Network proposed in [54]. We refer to this baseline as SAN. Following the original paper we have used 30

[4] https://www.cs.umb.edu/ smimarog/textmining/datasets/
[5] https://gluon-nlp.mxnet.io

Table 2.3: Performance with and without usage of the proposed LAMA mechanism. LAMA leads to improved performance over BiGRU alone.

Methods	News	Reuters	Yelp	IMDB	Yelp-L	YELP-P
BiGRU	0.905	0.867	0.663	0.876	0.608	0.943
LAMA	0.922	0.965	0.697	0.895	0.653	0.947
LAMA + Ctx	**0.923**	**0.973**	**0.716**	**0.90**	**0.665**	**0.952**

Table 2.4: Table reports the accuracy of the proposed models (LAMA, LAMA+ctx) against various baselines on sentiment analysis and news classification tasks. $+D_p$ refers to LAMA + Ctx with position-wise regularization whereas $+D_e$ refers to LAMA + Ctx with regularization over embeddings.

Methods	News	Reuters	Yelp	IMDB	Yelp-L	Yelp-P
SAN (Lin et al. [54])	0.876	0.942	0.68	0.831	0.638	0.945
CNN (Kim [43])	0.914	0.96	0.693	0.874	**0.672**	0.953
TE (Vaswani et al. [87])	0.899	0.901	0.655	0.817	0.569	0.925
BERT (Devlin et al. [18])	0.92	0.97	0.715	0.894	**0.672**	**0.97**
AVG (Arora et al. [4])	0.91	0.795	0.653	0.874	0.652	0.928
LAMA	0.922	0.965	0.697	0.895	0.653	0.947
LAMA + Ctx	**0.923**	**0.973**	**0.716**	**0.90**	0.665	0.952

attention heads and MLP hidden size of 512 for this baseline. For the next baseline, we use the scaled dot product multi-head attention proposed in [87] over sequence of pretrained word embeddings. We use one encoder layer and $m = 16$ attention heads, $d_{model} = 1024$ such that $d_{model}/m = 64$ as used in the original paper. We name this Transformer Encoder(TE) For our attention-based models we performed a grid search to identify the optimal number of attention heads to get the best performance.

2.5 Results

Table 2.3 reports the performance with and without using LAMA and 2.4 reports the accuracy of the baseline methods as compared to LAMA. Numbers highlighted in bold represent

best performing models. Tables report the accuracy of the best model on the test set after performing a 3-fold cross-validation.

We observe that a BiGRU with LAMA and global context initialization (LAMA+ctx) outperforms a BiGRU-only baseline by 2.0%(News), 12.2%(Reuters) 7.9%(Yelp) and 9.4%(Yelp-L) and 2.7%(IMDB), which confirms the value-add of self-attention for text classification tasks. The improvement is especially large on larger datasets (9.4% on Yelp-L and 7.9% on Yelp datset).

Table 2.4 compares the proposed models against a vareity of baselines from CNN to attention-based(SAN, TE) to language models(BERT). From the table it can be noted that LAMA+ctx outperforms SAN [54] on all tasks from 3.3% (Reuters)to 8.2% (IMDB). Proposed models outperform the TE baseline on all tasks. It should be noted that this performance improvement also comes with fewer parameters (§2.5.5). When compared to large pretrained language models such as BERT we find that LAMA outperforms BERT on News, Reuters, Yelp and IMDB datasets. On YELP-L and YELP-P BERT outperforms LAMA. However, it should be noted that besides being trained on large-scale text corpora and having a large memory footprint compared to LAMA, BERT models also take a longer time for training. For instance, for Yelp it took 8.5 hours as opposed to 25 mins for LAMA on one Nvidia Tesla P100 GPU. For the non-contextual baseline of average of word embeddings there's an improvement of 1.4% (News), 22.4%(Reuters) 9.8% (Yelp), 11.4% (Yelp-L) and 3.0%(IMDB) which confirms that contextual information captured by BiGRU or CNN models are indeed important for these tasks. Compared to CNN models an improvement of 1% (News), 1.4%(Reuters) and 3.3%(Yelp) and 3.0%(IMDB) can be observed. On the Yelp-L dataset our model and CNN perform similarly.

2.5.1 Contextual Attention Weights

To verify that the proposed model captures context dependent word importance, we plot the distribution of the attention weights of the positive words 'amazing', 'happy' and 'recommended' and negative words 'poor', 'terrible' and 'worst' from the test split of the Yelp data set as shown in Figure 2.3. We plot the distribution when conditioned on the ratings of the review. It can be seen from Figure 2.3 that the weight of positive words concentrates on the low end in the reviews with rating 1 (blue line). As the rating increases, the weight distribution shifts to the right. This indicates that positive words play a more important role for reviews with higher ratings. The trend is opposite for the negative words where words with negative connotation have lower attention weights for reviews with rating 5 (purple curve). However, there are a few exceptions. For example, it is intuitive that 'amazing' gets a high weight for reviews with high ratings but it also gets a high weight for reviews with rating 2 (orange curve) . This is because, inspecting the Yelp dataset we find that 'amazing' occurs quite frequently with the word 'not' in the reviews with rating 2; 'above average but not amazing', 'was ok but not amazing'. Our model captures this phrase-level context and assigns similar weights to 'not' and 'amazing'. 'not' being a negative word gets a high weight for lower ratings and hence so does 'amazing'. Similarly, other exceptions such as 'terrible' for rating 4 can be explained due to the fact that customers might dislike one aspect of a business such as their service but like another aspect such as food.

To further illustrate context-dependent word importance Table 2.5 lists top attended keywords for Yelp and Reuters datasets. It can be seen that certain words that often occur in pairs such as 'would recommend', 'very professional', 'very delicious' appear in the list for Yelp and 'exchange rate', 'trade deficit' for Reuters. This is because the model assigns close attention weights to the words in these phrases because they occur frequently together. Note also that superlatives such '100 stars' appear in the list which are strong indicators of the

Table 2.5: Top attended words for Yelp dataset from reviews with ratings 1 and 5 (indicated in parantheses) and Reuters(r8) Dataset.

Yelp(1)	Yelp(5)	r8(ship)	r8(money)
inconsiderate	recommend	kuwait	currencies
rudest	trust	gulf	monetary
goodnight	referred	south	miyazawa
worst	downside	tanker	stoltenberg
ever	professional	cargo	accord
boycott	100	warships	louvre
loud	happy	pentagon	fed
livid	attentive	says	cooperation
some	stars	begin	rate
hassle	please	iranian	poehl
friendly	delicious	shipping	exchange
ugh	safe	from	stability
dealership	worth	demand	currency
brutal	very	salvage	german
rather	would	trade	reagan
1	blown	production	pact
pizza	sensitive	japan	policy
slow	removed	gulf	support
torture	impressed	india	trade
absolute	looks	combat	deficit

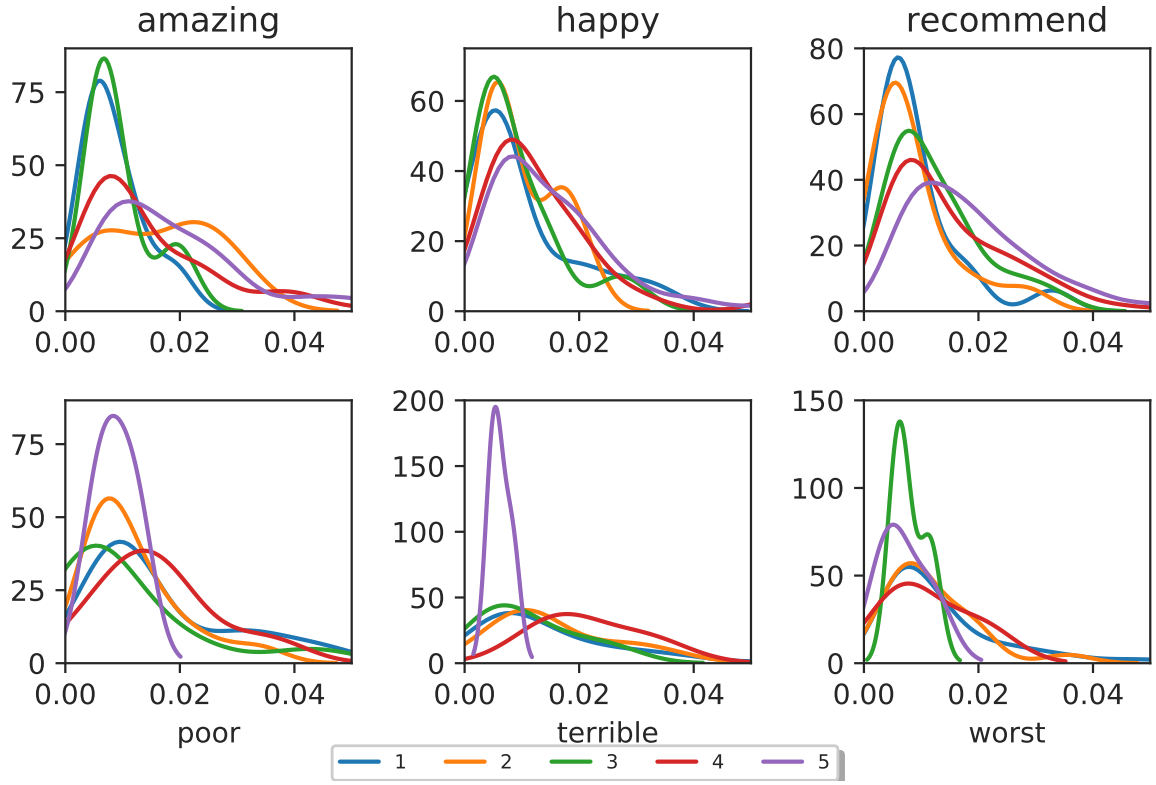


Figure 2.3: Attention weight (*x-axis*) distribution of the positive words 'amazing', 'happy' & 'recommend' and negative words 'poor', 'terrible', & 'worst'. Positive words tend to get higher weights in reviews with higher ratings (3-5) whereas negative words get higher weights for lower ratings (1-2)

sentiment of a review.

2.5.2 Why Multiple Heads?

Having a structured embedding with multiple rows, provides for more contextual representations. To verify this we evaluate the model performance as we vary the number of attention heads m from 1 to 25. Specifically, we plot the validation accuracy vs. epochs for different values of m, for the Yelp-L and IMDB datasets. We vary m from 1 to 20 to get 5 models with $m = 1$, $m = 5$, $m = 10$, $m = 15$ and $m = 20$. The plots are shown in Figure 2.4. From

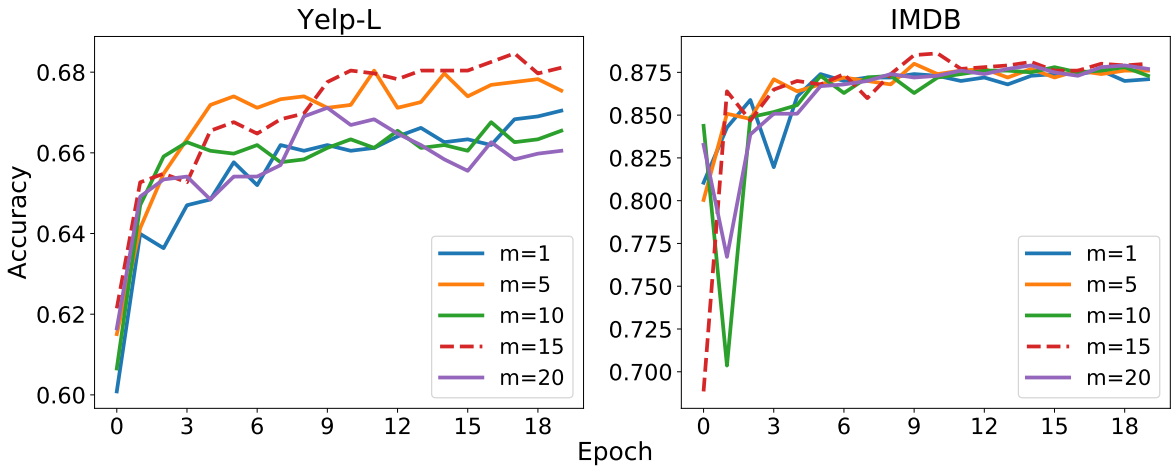


Figure 2.4: Figure shows the effect of using multiple attention heads. Validation accuracy of LAMA is plotted for different values of m for the Yelp-L dataset(left) and the IMDB dataset(right). *x-axis* indicates the number of epochs, *y-axis* indicates the accuracy. Accuracy peaks at $m = 15$ for both Yelp-L and IMDB.

the figure we can see that for the Yelp-L dataset model performance peaks for $m = 15$ and then starts falling for $m = 20$. We can clearly see a significant difference between $m = 1$ and $m = 20$, showing that having a multi-aspect attention mechanism helps. For the IMDB dataset model with $m = 15$ performs the best whereas model with $m = 1$ performs the worst although performances for $m = 5, 15, 20$ are similar to each other for this task.

2.5.3 Runtime

It should be noted that since LAMA is applied on top of RNN the computational bottleneck is the sequential computation on RNNs since attention cannot be computed unless all RNN hidden states are available. Hence, the computational time increases linearly with input length and quadratically with hidden state dimension ($O(n * h^2)$). Once RNN hidden states are available the complexity of the attention layer LAMA is $O(n * m * h)$. In comparison, the complexity of the self-attention mechanism in Transformer is $O(n^2 * h)$, where h is the hidden state size. It increases quadratically with increasing input length and linearly with hidden

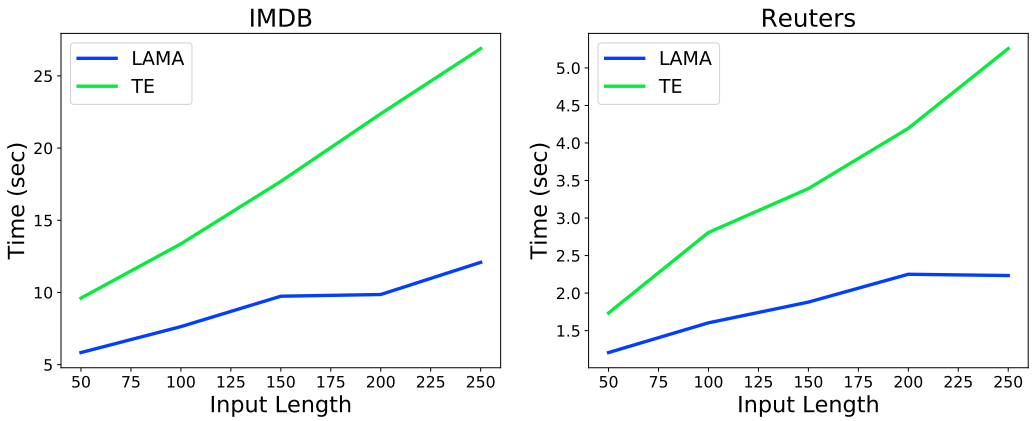


Figure 2.5: Figure shows the average run time per epoch in seconds (averaged over 10 epochs) for LAMA Encoder (LE) and Transformer Encoder (TE) models as a function of input sequence length (50 to 250). It can be seen that TE (green) is more computationally expensive compared to LAMA (blue). LE is LAMA attention mechanism applied directly to word embeddings without computing GRU hidden states.

state dimension. In this section, we plot the runtimes of LAMA and a 1-layer TE. For a fair comparison, we compare the runtimes of LAMA free from RNN. We propose, LAMA Encoder – a model that doesn't use RNN to model sequential dependencies and directly operates on the word embeddings of the inputs. That is, the hidden state matrix $\mathbf{H}$ in Eq. 2.13 is populated by the pretrained word embeddings. Figure 2.5 shows the average runtime per epoch (averaged over 10 epochs) of LAMA Encoder (LE) and Transformer Encoder (TE) when sequence lengths are increased from 50 to 250 for IMDB and Reuters datasets. It can be seen from the figure that TE is computationally more expensive per epoch than LE. LE is also a competitive model with test accuracies of 83.0% on IMDB and 93.9% on Reuters as compared to 81.7% (IMDB) and 90.1% (REUTERS) of TE.

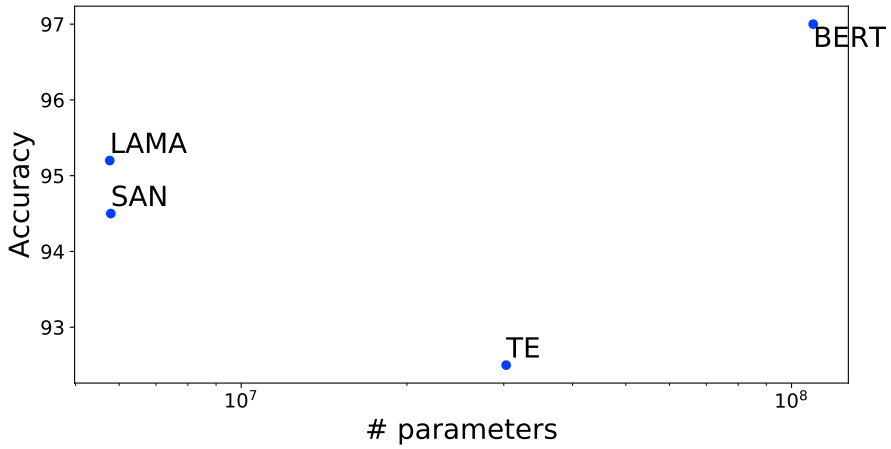


Figure 2.6: Figure shows the accuracy (y-axis) of the models LAMA, SAN, TE and BERT models on YELP-P dataset when viewed against the model parameters (x-axis). LAMA outperforms SAN and TE while also being more parameter efficient. BERT outperforms LAMA by 1.8% but by more than an order of magnitude increase in parameters.

2.5.4 Parameters vs Accuracy

On the YELP-P dataset our model is outperformed by BERT by 1.8 %. However it it worth noting the cost of this performance improvement. Fig. 2.6 shows the accuracy of different models and their corresponding parameters on the YELP-P dataset. It can be seen that LAMA+ctx outperforms SAN & TE with fewer parameters (difference being linear). BERT slightly outperforms LAMA+ctx however with more than an order of magnitude increase in parameters.

2.5.5 Attentional Unit Efficiency

Now lets compare the number of trainable parameters in the attention layer of three multi-head attention mechanisms – proposed model (LAMA), Self-Attentive Network (SAN) and TE (Figure 2.7).

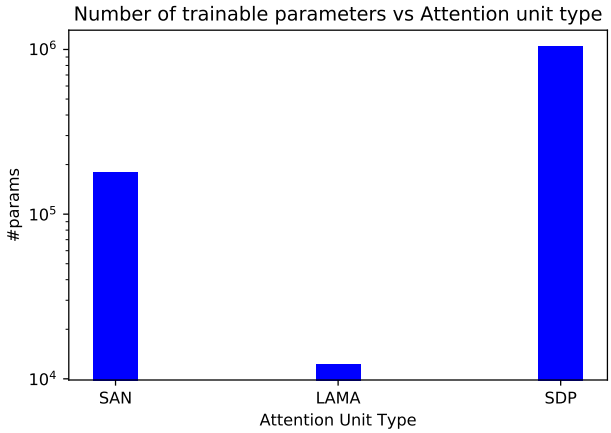


Figure 2.7: Comparison of number of trainable parameters in the attention layer of SAN, LAMA and TE. Number of parameters increase linearly for SAN and LAMA, where as number of parameters in LAMA are less than SAN by a constant. Number of parameters in TE exceed LAMA by an order of magnitude.

Other variables such as input sentence length and dimension of the hidden state representations held constant, the computational complexity depends on the attention layer. Here, we show how the proposed model LAMA compares to the self attentive network (SAN) [54] and TE with respect to the number of parameters needed to compute an attention matrix for a sentence. In SAN the attention matrix $\mathbf{A}$ can be computed as:

$$\mathbf{A} = softmax(\mathbf{W}_{s_2} tanh(\mathbf{W}_{s_1} \mathbf{H}^T)) \tag{2.19}$$

where, $\mathbf{H}$ is a matrix of word annotations with the shape T-by-$2h$, $\mathbf{W}_{s_1}$ is d_a-by-$2h$ and $\mathbf{W}_{s_2}$ is m-by-$2h$ where m is number of heads. So the total number of parameters needed to be learned are $2hd_a + 2hm$ ($\approx 2hd_a$ for $h_a \gg m$). For the proposed model, to compute the attention matrix given by Eq. 2.13 the parameters to be learned are matrices $\tilde{\mathbb{P}}$, $\tilde{\mathbb{Q}}$ and the context vector $\mathbf{c}$. So the total number of parameters required are $2hm + 2hm$. Comparing the above terms the reduction factor is $\frac{d_a+m}{2m}$. For $m << d_a$, reduction factor is $\frac{d_a}{2m}$. Even though, both are $O(m)$ the parameter savings come from the constant factor $\frac{d_a}{2m}$. For the

Transformer model [87], the multi-head attention in one layer is given by;

$$Multihead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Concat(head_1, ..., head_m)\mathbf{W}^O \tag{2.20}$$

$$\text{where head}_i = Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \tag{2.21}$$

where the projections are parameter matrices $\mathbf{W}_i^Q$ & $\mathbf{W}_i^K \in R^{d_{model} \times d_k}$, $\mathbf{W}_i^V \in R^{d_{model} \times d_v}$ and $\mathbf{W}^O \in R^{md_v \times d_{model}}$. In the self-attention case, all of $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the input representations from the previous layer. The number of parameters required to compute the multi-head attention in Eq. 2.20 are $O(m^2)$. It should be noted that even though the attention computation in Transformer is an order of magnitude higher, it does not contain any GRU layers. So the overall complexity depends on the choice of the input layer such as convolutional or recurrent layers. However,our experiments indicate that LAMA-Encoder (where LAMA is directly applied over a sequence of word embeddings is equally effective on some tasks).

Figure 2.7 illustrates the attention computation in LAMA, SAN respectively. In LAMA the number of trainable parameters are proportional to $2 * m * h$, where in SAN the number of trainable parameters are proportional to $2 * h * d_a$. For comparison, Figure 2.7 plots the number of parameters for some standard values of the the quantities $h = 512, m = 12, d_a = 350$. As can be seen the number of parameters in TE are significantly higher than number of parameters in SAN which is higher than number of parameters in LAMA. Such high number of parameters perhaps explain the difficulty that we observed in training TE for text classification tasks.

2.6 Related Work

The practice of applying attention has a history particularly in computer vision problems [16, 49, 95]. Attention mechanism can be of two types. "Hard" attention model learns to put attention to only specific parts of the sequences or images [67] whereas, 'soft" attention learns the attention weights over the entire sequence. Gulcehre et al. [28] enhance the attention mechanism by incorporating a planning mechanism. Based on the past actions and current observation the model makes a plan of future alignments. Xu et al. [95] use attention to automatically generate description of images. They have explored both "soft" and "hard" attention mechanisms.

Spearheaded by their success in neural machine translation [5, 6, 28, 60, 87] attention mechanisms are used in problems involving sequence to sequence models such as question answering [20, 31, 80] text summarization [12, 71] and visual question answering [97, 98]. In sequence modeling, attention mechanisms allow the decoder to learn which part of the sequence it should "attend" to based on the input sequence and the output it has generated so far [5]. A special case of attention known as self-attention [54] or intra attention [96] is used for text classification tasks such as sentiment analysis and natural language inference. A by-product of self-attention mechanism is visualization of words being attended leading to the explanability of the models.

Models have been proposed that compute multiple attention distributions over a single sequence of words. In multi-view networks proposed by Guo et al. [29] they use a different set of parameters for each view which leads to a large increase in the number of parameters with increasing number of views. Lin et al. [54] alleviate this problem by producing a matrix embedding from a single set of parameters. Both these methods use a special case of additive attention proposed by Bahdanau et al. [6] in the context of neural machine translation. Lu-

ong et al. [60] simplify additive attention operation by introducing multiplicative attention which is faster to compute. In multiplicative attention, score between two feature vectors is learned using a bilinear projection matrix. Dot product attention [60] is a special case of multiplicative attention where the score between two features vectors is computed by a simple dot product between them. Yang et al. [96] use dot product attention to compute the similarity of word hidden representation to a word-level context vector which is learned with the rest of the model. In this work we compute the score between the context vector and the word hidden representation using a bilinear projection matrix and then we use an approach inspired by multi-modal low rank bilinear pooling proposed by Kim et al. [42] to factorize the matrix into two low rank matrices to compute multiple attention distributions over words. Contrary to Guo et al. [29] we use matrix factorization to alleviate the problem of increasing parameters with increasing views and our approach uses fewer parameters than Lin et al. to compute multiple attentions and performs superior to their approach. We call this as multi-aspect attention as it attends to different aspects or parts of a sentence for constructing a sentence embedding.

2.7 Discussion

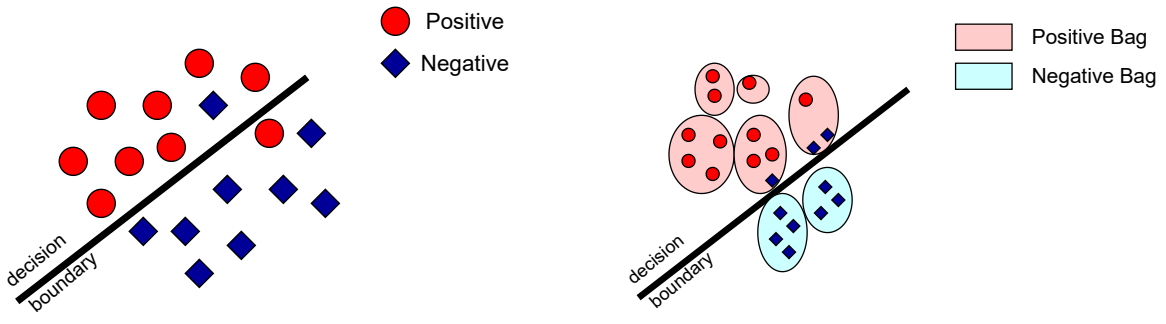
In this chapter, we presented a novel compact multi-head self-attention mechanism and illustrated its effectiveness on text classification benchmarks. The proposed method computes multiple attention distributions over words which leads to contextual sentence representations. The results showed that this mechanism performed better than several other approaches including non-contextual unsupervised baselines such as average of word embeddings, contextual baselines such as LSTM-based methods and CNNs and also other attention mechanisms with fewer parameters. We further demonstrated the computational superiority

of our approach in comparison to previously proposed multi-head mechanisms in computing attentions and hence is more amenable in low-resource scenarios. In the next chapter, we will see how LAMA can be used to learn rich representations for event detection.

Chapter 3

Event Detection

In this chapter, we model the tasks of event detection (event document classification, event extent and event trigger extraction) from news articles in a unified framework without explicit labels at the sentence level by leveraging the implicit hierarchy in the text. We model this task in a hierarchical MIL framework. We show that using the LAMA attention mechanism proposed in Chapter 2 as the word-level MIL aggregator function leads to more effective event detection performance.



(a) Supervised Learning

(b) Multiple Instance Learning

Figure 3.1: Supervised learning and multiple instance learning(MIL). In MIL a bag is positive if alteast once instance in the bag is positive and a bag is negative if all instances are negative.

3.1 Introduction

In event detection, we're interested in identifying documents containing event mentions as well as the event extents and event triggers. We can use the framework of multi-instance learning(MIL) for this task. MIL is a bag-level supervised learning task. The training data is a set of labeled bags, where a bag refers to a document, and each document contains several instances where an instance refers to a sentence. Formally, we have a dataset $D = (X_i, y_i)_{i=1}^{n}$, where $X_i = \{s_j\}_{j=1}^{n_i}, y_i \in \{1, 0\}$ is the label of X_i indicating whether an event is mentioned in the document or not, N is the number of training documents, n_i is the number of sentences in X_i. Each $S_j \in I$ is an instance, where I denotes the instance space. Fig. 3.1 illustrates the difference between standard supervised learning and MIL.

Deep Learning based MIL approach consists of three steps: (i) a transformation of instances to a low-dimensional embedding, (ii) a permutation-invariant (symmetric) aggregation function, and (iii) a final transformation to the bag probability.

The model can be trained by optimizing the log-likelihood function where the bag label is distributed according to the Bernoulli distribution with the parameter $\theta(X) \in [0, 1]$, i.e., the probability of $Y = 1$ given the bag of instances X.

For a given aggregation function there are two main approaches to MIL: i) The instance-level approach: an instance-level classifier returns scores for each instance. Then individual scores are aggregated by MIL pooling to obtain $\theta(X)$. (ii) The embedding-level approach: instances are mapped to a low-dimensional embedding. MIL pooling is used to obtain a bag representation that is independent of the number of instances in the bag. The bag representation is further processed by a bag-level classifier to provide $\theta(X)$.

Prior approaches to event detection employ the instance level approach [92]. However, since the individual labels are unknown, there is a threat that the instance-level classifier might be

trained insufficiently and it introduces additional error to the final prediction. To alleviate this issue we use embedding-level approach. Moreover, prior approaches use non-differential and non-trainable MIL pooling operators such as max or mean operators to obtain the bag level probability. A flexible and adaptive MIL pooling could potentially achieve better results by adjusting to a task and data. Ilse et al. [33] show that attention mechanisms can be used as differential MIL pooling operators. Attention-based MIL pooling allows to assign different weights to instances within a bag and hence the final representation of the bag could be highly informative for the bag-level classifier. Moreover, this allows us to mitigate the drawbacks of the strong assumption that event extents are self-contained i.e. all the event related information is neatly partitioned into individual sentences. If some event related information is contained within peripheral sentences attention weights will be trained in accordance for those instances.

Moreover, in the case of a positive label, $X_i = 1$, high attention weights should be assigned to positive instances i.e. $S_j = 1$ (key instances). The attention mechanism allows to easily interpret the provided decision in terms of instance-level labels. In fact, the attention network does not provide scores as the instance-based classifier does but it can be considered as a proxy to that.

Event detection can be modeled as a hierarchical MIL task where the coarser level task is identification of documents containing an event and identifying key sentences(event extents) and the fine-grained task of identifying trigger words from the detected key sentences. For the fine-grained task, each sentence is a considered as a bag of words, and sentence embeddings are obtained by using an attention mechanism over word embeddings.

A deviation from prior MIL based approaches especially for vision tasks [33] is that we don't assume the bag embedding to be permutation-invariant. We use an RNN to compute

instance embeddings which makes them dependent on the position in the bag [1].

To summarize, we use a recurrent neural network (RNN) based hierarchical model with attention mechanism at both levels in the hierarchy that constructs sentence and document representations using RNN based encoders. The sentence representation is constructed by attending to all words in the sentence, whereas the document representation is constructed by attending to all sentences. The model inherently weighs words and sentences and hence can construct a representation that captures the entire context of a document. To capture richer sentence representations we use the LAMA multi-head attention mechanism proposed in chapter 2 that allows for multiple attention distributions over words. Simple dot-product attention mechanisms [96] might be suboptimal whereas existing multi-head self-attention mechanisms [54, 87] contain too many parameters to be trained effectively in a hierarchical setting. Hence, LAMA mechanism which learns richer representations than single-head approach and is more parameter efficient than existing multi-head approaches can be used.

Figure 1.1 gives an overview of the propose approach. Proposed model consists of five main components: 1) A sentence encoder that uses a bi-directional GRU [7] to construct word representations, 2) the previously proposed LAMA mechanism that helps construct sentence representations and extract trigger words, 3) a document encoder that constructs contextual sentence representations using a bi-directional GRU, 4) a sentence level attention mechanism that helps construct document representations and extract key sentences and finally 5) a multi-layer perceptron (MLP) classifier that predicts document labels.

A given news article is first segmented into sentences and each sentence is encoded via the bi-GRU sentence encoder to get a contextual annotation of each word in that sentence. The LAMA attention mechanism then obtains multiple attention distributions over those words

[1] We stick to the terms 'bag' and 'instance' to be consistent with the literature, even though it is not a 'bag' in the strictest sense of its usage as in 'unordered collection of items'.

by computing an alignment score of their hidden representation with a word-level context vector. Sum of the word representations weighted by the scores from multiple attention distributions then forms a matrix sentence embedding. The matrix embedding is obtained for each sentence, flattened and then passed to a GRU document encoder. Contextual annotation of each sentence is obtained by concatenating the forward and backward hidden states of a GRU. A single pass attention distribution is obtained over sentence hidden representations to finally obtain a document representation using a weighted sum of the sentence annotations. Finally a classifier is used to predict whether the resulting document embedding represents an event or not. Please refer to fig. 3.2 for a schematic of our model.

In the following sections we describe each model component in detail. Matrices are denoted by bold capital letters, vectors by bold small letters and unbolded small letters are scalars.

3.2 Hierachical Model for Event Detection

3.2.1 Sequence Encoder

We use the GRU [7] RNN as the sequence encoder. For detailed description of GRU encoder please refer to §2.3.1. Consider a news article containing n sentences and each sentence contains T words. A sentence consists of word tokens w_{it}, $t \in [0, T]$ where every word is converted to a real valued word vector $\mathbf{x}_{it}$ using the pre-trained embedding matrix $\mathbf{W}_e = R^{d \times |V|}$, $\mathbf{x}_{it} = \mathbf{W}_e \mathbf{w}_{it}$, $t \in [1, T]$ where d is the embedding dimension and V is the vocabulary. The embedding matrix $\mathbf{W}_e$ is fine-tuned during training. We encode the sentence using a bi-directional GRU (bi-GRU) that summarizes information in both directions along the sentence to get a contextual annotation of a word. In a bi-GRU the hidden state at time step t is represented as a concatenation of hidden states in the forward and backward direction.

The forward GRU denoted by $\overrightarrow{GRU}$ processes the sentence from w_{i1} to w_{iT} whereas the backward GRU denoted by $\overleftarrow{GRU}$ processes it from w_{iT} to w_{i1}.

$$\mathbf{x}_{it} = \mathbf{W}_e \mathbf{w}_{it} \tag{3.1}$$

$$\overrightarrow{\mathbf{h}_{it}} = \overrightarrow{GRU}(\mathbf{x}_{it}, \mathbf{h}_{i(t-1)}, \boldsymbol{\theta}) \tag{3.2a}$$

$$\overleftarrow{\mathbf{h}_{it}} = \overleftarrow{GRU}(\mathbf{x}_{it}, \mathbf{h}_{i(t+1)}, \boldsymbol{\theta}) \tag{3.2b}$$

Here the word annotation $\mathbf{h}_{it}$ is obtained by concatenating the forward hidden state $\overrightarrow{\mathbf{h}_{it}}$ and the backward hidden state $\overleftarrow{\mathbf{h}_{it}}$.

3.2.2 Intra Sentence Attention

Event triggers should be given in a sentence should be assigned higher weight than other words while computing a sentence representation. Since an event related trigger can occur anywhere in a sentence we choose the global attention mechanism [60] in which the sentence representation is computed by attending to all words in the sentence. Let $\mathbf{h}_{it}$ be the annotation corresponding to the word $\mathbf{x}_{it}$. First we transform $\mathbf{h}_{it}$ using a one layer Multi-Layer Perceptron (MLP) to obtain its hidden representation $\mathbf{u}_{it}$.

$$\mathbf{u}_{it} = tanh(\mathbf{W}_w \mathbf{h}_{it} + \mathbf{b}_w) \tag{3.3}$$

We measure the importance of word by computing an alignment score of $\mathbf{u}_{it}$ to a word level context vector $\mathbf{u}_w$ using a bilinear model:

$$f_{it} = \mathbf{u}_w^T \mathbf{W}_i \mathbf{u}_{it} \tag{3.4}$$

Here, $\mathbf{W}_i$ is a bilinear projection matrix, $\mathbf{u}_w$ is randomly initialized and jointly learned with other parameters during training. Similar to Yang et al. [96], $\mathbf{u}_w$ can be seen as a high dimensional representation of the fixed query 'What is the informative word'. $\mathbf{u}_w \in \mathbb{R}^l$, $\mathbf{u}_{it} \in \mathbb{R}^{2h}$ and $\mathbf{W}_i \in \mathbb{R}^{l \times 2h}$.

The attention weight for the word $\mathbf{x}_{it}$ can be computed through a softmax function.

$$\alpha_{it} = \frac{exp(f_{it})}{\sum_{t'} exp(f_{it'})} \tag{3.5}$$

The attention distribution above usually focuses on a specific component of the sentence, like a special set of trigger words or phrases. So it is expected to reflect a component of the semantics in a sentence. To capture richer sentence representations we use a multi-head attention mechanism. However, since this mechanism is deeper in the hierarchy and the word attention weights will only be trained via document-level supervision it is important for this module to be lean. For this, we use the low-rank multi-head attention mechanism, LAMA proposed in the previous chapter. Concretely, the multi-head attention matrix for the sentence i can be obtained using Eq. 2.13.

$$\mathbf{A}_i = softmax(l2(tanh(\tilde{\mathbb{P}}^T \mathbf{C}_g \circ \tilde{\mathbb{Q}}^T \mathbf{H}_i^T))) \tag{3.6}$$

Here, $\mathbf{H}_i = (\mathbf{h}_{i1}, \mathbf{h}_{i2}, ...\mathbf{h}_{iT})$ is a matrix of all word annotations in a sentence; $\mathbf{H}_i \in \mathbb{R}^{T \times 2h}$. $\mathbb{P}$ and $\mathbb{Q}$ are parameters of the model and $\mathbf{C}_g$ is context matrix, the rows of which is the word-level context vector $\mathbf{u}_w$. Let $\mathbf{A}_i = [\boldsymbol{\alpha}_{i1}; \boldsymbol{\alpha}_{i2}; ...\boldsymbol{\alpha}_{iT}]$; $\mathbf{A}_i \in \mathbb{R}^{m \times T}$ and each $\boldsymbol{\alpha}_{ij}$ is a column, be the multi-head attention matrix for the sentence. The sentence representation corresponding to one head j given by $\boldsymbol{\alpha}_{ij} = [\alpha_{j1}, \alpha_{j2}, ..\alpha_{jT}]$ corresponding to a row in $\mathbf{A}_i$,

can be computed by taking a weighted sum of all word annotations.

$$\mathbf{s}_{ij} = \sum_{k=1}^{T} \mathbf{h}_{ik} * \alpha_{jk} \tag{3.7}$$

Similarly, sentence representation can be computed for all heads and is given in a compact form by:

$$\mathbf{S}_i = \mathbf{A}_i \mathbf{H}_i \tag{3.8}$$

Here $\mathbf{S}_i \in \mathbb{R}^{m \times 2h}$ is a matrix sentence embedding and contains as many rows as the number of attention heads. Each row contains a sentence representation corresponding to a different attention head. This matrix can be flattened by concatenating all rows for further processing.

3.2.3 Intra Document Attention

News articles consist several sentences with a few of them being event extents and the rest describing supporting facts. The event extents should be assigned higher weights while computing document representation. Instead of hard selecting top K sentences and aggregating their probabilities as in prior instance-level MIL approaches such as [92] we use the global attention mechanism over the sentence annotations to get the document representation. Specifically, given a document containing sentence embeddings $\{\mathbf{s}_1, ..., \mathbf{s}_i, ..., \mathbf{s}_n\}$ where each $\mathbf{s}_i$ is a flattened representation of the matrix sentence representation $\mathbf{S}_i$ as given by eq. 3.8 we get the document vector as follows.

$$\overrightarrow{\mathbf{h}_i} = \overrightarrow{GRU}(\mathbf{s}_i, \mathbf{h}_{i-1}, \boldsymbol{\theta}) \tag{3.9a}$$

$$\overleftarrow{\mathbf{h}_i} = \overleftarrow{GRU}(\mathbf{s}_i, \mathbf{h}_{i+1}, \boldsymbol{\theta}) \tag{3.9b}$$

$$\mathbf{h}_i = \{\overrightarrow{\mathbf{h}_i}, \overleftarrow{\mathbf{h}_i}\} \tag{3.9c}$$

The sentence annotation $\mathbf{h}_i$ is obtained by concatenating the forward and backward hidden representations of the bi-GRU. Document representation is obtained by attention over sentences.

$$\mathbf{u}_i = tanh(\mathbf{W}_s\mathbf{h}_i + \mathbf{b}_s), \tag{3.10a}$$

$$\alpha_i = \frac{exp(\mathbf{u}_i^T\mathbf{u}_s)}{\sum_t' exp(\mathbf{u}_{t'}^T\mathbf{u}_s)} \tag{3.10b}$$

$$\mathbf{d} = \sum_i \alpha_i\mathbf{h}_i \tag{3.10c}$$

Here $\mathbf{u}_s$, the sentence level context vector, is randomly initialized and learned along with other model parameters while training and $\mathbf{d}$ is the document representation that summarizes all the information in the article. Given the document representation, we use a two hidden layer MLP with dropout to get the class scores.

$$\hat{y} = \mathbf{W}_c\mathbf{d} + \mathbf{b}_c. \tag{3.11}$$

Loss for the document is computed using the standard cross entropy.

$$l = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y})) \tag{3.12}$$

The sentence embedding given by eq. 3.8 can suffer from redundancy issues if the attention mechanism always provides similar weights for all the m aspects. To attend to a small set of trigger words in each aspect and to encourage diversity in different aspects we use penalization as described by Lin et al. [54] that is added to the loss:

$$P = \frac{\sum_i \left\|\mathbf{A}_i\mathbf{A}_i^T - \mathbf{I}\right\|_F^2}{n} \tag{3.13}$$

Where $\|.\|_F$ stands for the Frobenius norm of a matrix and the summation is taken over all sentences in the document. The final training loss is given by:

$$L = \sum_d l + \lambda P \tag{3.14}$$

The summation is taken over all the documents in the batch and λ is a hyperparameter. We use the mini-batch stochastic gradient descent algorithm [38] with momentum and weight decay for optimizing the loss function and the backpropogation algorithm is used to compute the gradients.

Hyperparameters

We use a word embedding size of 100. The embedding matrix $\mathbf{W}_e$ is pretrained on the corpus using the gensim [2] implementation of the widely used distributed representations model word2vec [66]. All words appearing less than 5 times are discarded. The GRU hidden state is set to $h = 50$. In LAMA the dimension of $\mathbf{u}_{it}$ is given by the dimension of the GRU hidden state, but the dimension of $\mathbf{u}_w$ can be tuned. Empirically we find that setting the dimension of $\mathbf{u}_w$ to 32 gives us the best results. We set the classifier MLP hidden state to 512 and apply a 0.4 dropout to the hidden layer. We use a batch size of 64 for training and an initial learning rate of 0.05. For early stopping we use $patience = 5$.

[2] https://radimrehurek.com/gensim/

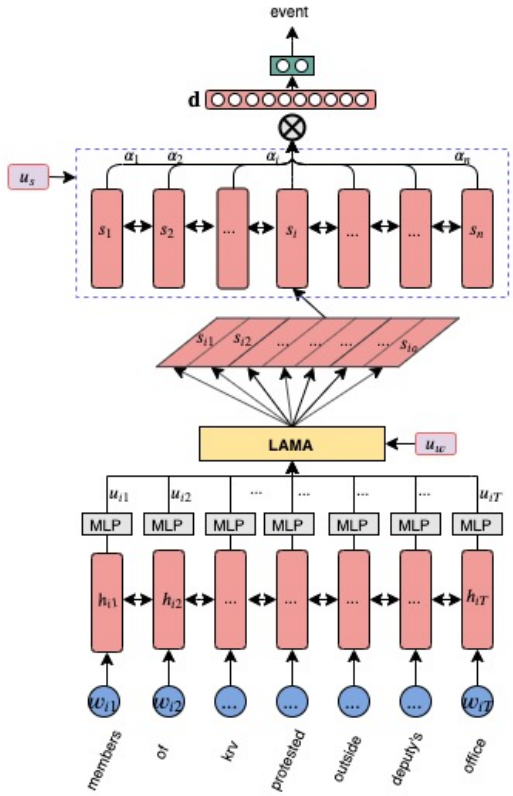


Figure 3.2: An illustration of the model architecture. The LAMA component is used at the word level to get the matrix sentence embedding.

3.3 Experiments

3.3.1 Datasets

To evaluate our approach we use three event datasets - (i) the Civil Unrest Gold Standard Report labeled manually by analysts from MITRE corporation for the experimental evaluations [76]. It contains encodings of civil unrest events from 10 Latin American countries in Argentina, Brazil, Chile, Colombia, Ecuador, El Salvador, Mexico, Paraguay, Uruguay, and Venezuela. The encodings are obtained from major national newspapers as identified by 4imn.com. It contains a total of 24,110 news articles out of which 18% mention a protest event while the rest are non-protest articles. All the articles are in Spanish and we refer

to this dataset as CU (Spanish) (ii) The AutoGSR dataset – This dataset comes from the Embers AutoGSR system [78] which is a web based system that generates validated civil unrest events extracted from news articles. It contains a total of 32,019 news articles out of which 18% describe protest events (protest articles) and the rest do not describe any protest events (non-protest articles). We refer to this dataset as CU (English). For each protest article, the CU English & Spanish datasets contain the population type and protest event type. The population type indicates which class of population are involved in the protest and the event type indicates the main reason behind the protest. These are listed in table 3.1.

Table 3.1: First row indicates population classes participating in a protest in the CU dataset. Second row indicates the causes of protest.

Event Population	General Population, Business,Legal, Labor, Agricultural, Education, Medical, Media
Event Type	Government Policies, Employment and Wages, Energy and Resources, Economic Policies, Housing

Finally, we evaluate on iii) the Military Action and Non-State Actor (MANSA) GSR dataset which is in English and Arabic. This contains event encodings from gulf countries namely, Bahrain, Egpyt, Iraq, Jordan, Lebanon, Qatar, Saudi Arabia and Syria. The event types include 'Military Actions' (MA) which are actions by military, police, or security organization and 'Non-State Actor'(NSA) which are actions initiated by non-governmental groups or individuals to further political, social, religious or ideological objectives. These events are encoded from news articles collected from the web and print media. Event collection techniques include Google Advanced Search (limited to the newspaper website), Nexis queries, and IHS Janes. Google Advanced Search is used to collect events in online media. Nexis and IHS Janes are used to collect events in print media. About 34% articles describe an event and rest are non-event articles. We refer to this dataset as MANSA dataset. MA &

Table 3.2: Dataset statistics. Total number of news articles, average number of sentences per article and average number of words per article in the datasets.

Datset	**# articles**	**# sents**	**# words**
MANSA	1,05,858	6.3	250.3
CU Spanish	24,110	11.4	337.1
CU English	32,019	10.1	358.2

Table 3.3: Various baselines and their key characteristics.

Feature Method	**hierarchical**	**key words**	**key sents**	**bilinear attention**	**multi-aspect attention**
MICNN	✓	×	✓	×	×
MIGCNN	✓	✓	✓	×	×
HAN	✓	✓	✓	×	×
BSA	✓	✓	✓	✓	×
HSA	✓	✓	✓	×	✓
LAMA	✓	✓	✓	✓	✓

NSA events are further divided into subtypes. In this work we combine NSA & MA events together for detection. Please refer to table 3.2 for the overview of our datasets.

‘

3.3.2 Comparative Methods

Table 3.3 shows the different approaches that were evaluated in this study along with their key characteristics. CNNs within a multiple instance learning (MIL) framework have been used by Wang et al. [92]. We consider the MI-CNN model proposed by Wang et al. and its variants as our baselines. The MI-CNN approach first constructs a sentence vector by applying convolution in the temporal dimension followed by $k - maxpooling$. Then a document vector is formed from sentence vectors in a similar way. Instance representation for each sentence is constructed by concatenating the sentence representation and the document representation. Finally probabilities at instance level are estimated and aggregated to

Table 3.4: Results Event Detection. LAMA refers to the proposed model. BSA refers to the Bilinear Single-aspect Attention mechanism presented in eq. 3.4, MI-GCNN refers to our MIL model, where the CNN encoder is replaced by the RNN encoder followed by simple dot product attention to extract key words. HAN, HSA & MI-CNN are other baselines.

Dataset	**Method**	**Precision** (std.)	**Recall** (std.)	**F1** (std.)
MANSA	MI-CNN **(Wang et al.)**	0.731 (0.012)	0.785 (0.004)	0.686 (0.021)
	MI-GCNN (Ours)	**0.793 (0.003)**	0.622 (0.013)	0.697 (0.007)
	HSA **(Lin et al.)**	0.733 (0.001)	0.823 (0.009)	0.775 (0.003)
	HAN **(Yang et al.)**	0.740 (0.007)	0.831(0.007)	0.783 (0.004)
	BSA (Ours)	0.737 (0.007)	**0.834 (0.003)**	0.782 (0.003)
	LAMA (Ours)	0.747(0.003)	0.831 (0.003)	**0.787(0.002)**
CU (Spanish)	MI-CNN **(Wang et al.)**	0.742 (0.036)	**0.813 (0.041)**	0.775 (0.006)
	MI-GCNN (Ours)	**0.834 (0.011)**	0.721 (0.009)	0.773 (0.005)
	HSA **(Lin et al.)**	0.763 (0.009)	0.745 (0.016)	0.754 (0.011)
	HAN **(Yang et al.)**	0.811 (0.011)	0.775 (0.011)	0.793 (0.005)
	BSA (Ours)	0.812 (0.015)	0.779 (0.011)	0.795 (0.007)
	LAMA (Ours)	0.816 (0.017)	0.784 (0.010)	**0.800 (0.008)**
CU (English)	MI-CNN **(Wang et al.)**	**0.824 (0.01)**	0.644(0.009)	0.723 (0.009)
	MI-GCNN (Ours)	0.815 (0.006)	0.68 (0.017)	0.742 (0.011)
	HSA **(Lin et al.)**	0.746 (0.008)	0.710(0.017)	0.727 (0.010)
	HAN **(Yang et al.)**	0.779 (0.012)	0.746 (0.024)	0.762 (0.015)
	BSA (Ours)	0.786 (0.008)	**0.757 (0.007)**	**0.771 (0.006)**
	LAMA (Ours)	0.785 (0.006)	0.745 (0.007)	0.764 (0.005)

compute the document level probability. This model extracts key sentences but does not extract trigger words from the document nor attends to those words while constructing a sentence representation. In our first baseline we replace the convolutional sentence encoder by a GRU sentence encoder followed by a simple attention mechanism that attends to words while constructing a sentence representation. We refer to this model as MI-GCNN and it extracts both trigger words and key sentences.

Since, the proposed model is a multi-aspect attention mechanism, we compare it with another multi-aspect attention mechanism proposed by Lin et al. [54]. We refer to as Hierarchical Self Attention (HSA) where we replace the word-level attention mechanism by the Self-Attentive mechanism proposed by them.

We also evaluated the Hierarchical Attention Network (HAN) model proposed by Yang et. al. [96], which attends to both trigger words and keys sentences while constructing sentence and document representations respectively but it only consists of a single aspect attention mechanism at both levels.

Finally, we replace the word level attention in the HAN model with the bilinear attention mechanism given by eq. 3.4. Having bilinear attention not only enables the model to learn better pairwise similarity between the word level context vector $\mathbf{u}_w$ and the the word hidden representations $\mathbf{u}_{it}$ but it also gives us the flexibility to vary the dimension of $\mathbf{u}_w$. We refer to this model as Bilinear Single Aspect Attention model (BSA).

We compare the performance of these models with the proposed model(LAMA). In all the datasets, the event class is a rare class and hence we report the precision, recall and F1 score of that class for the test set. For each dataset we trained our model using 5-fold cross-validation with an 80/20 train/test split and employed early stopping. Models took less than an hour on a single Tesla P100 GPU to train. In our experiments all models were implemented using the Pytorch[3] open source framework.

3.4 Results

3.4.1 Event Detection

Table 3.4 reports the performance of our models compared to the baseline approaches. On average, the proposed model outperforms the instance-level MIL-based MICNN and MIGCNN models by 14.2 % & 12.9% on MANSA, 32.2 % & 3.5 % on CU Spanish and 6.6% & 3.9% on CU English datasets respectively. The proposed model also outperforms HAN across

[3]https://pytorch.org/

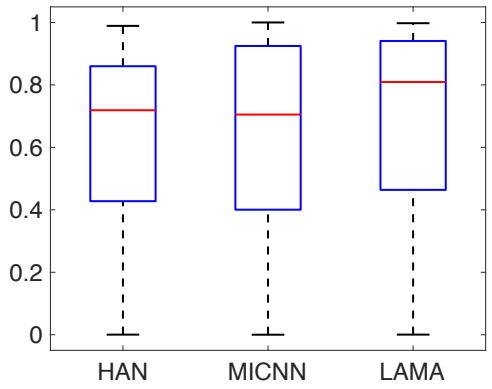
the three datasets. Moreover, unlike HAN our model attends to different aspects while constructing a sequence representation, which results in an increased model size due to added parameters. One key aspect of our model is the ability to tune the dimensionality of the word level context vector $\mathbf{u}_w$. We find that models with $l \leq 2h$ where l is the dimension of $\mathbf{u}_w$ and $2h$ is the dimension of the word hidden representation tend to outperform models with $l > 2h$. The LAMA model is forced to learn a more compact representation of the word-level context vector and thus, retains only the most relevant information.

LAMA also outperforms HSA on CU English, CU Spanish & MANSA datasets by 5.1 %, 6.1 % and 1.5% respectively. Moreover our attention mechanism uses fewer parameters than the Self-Attentive model proposed by Lin et al. [54].

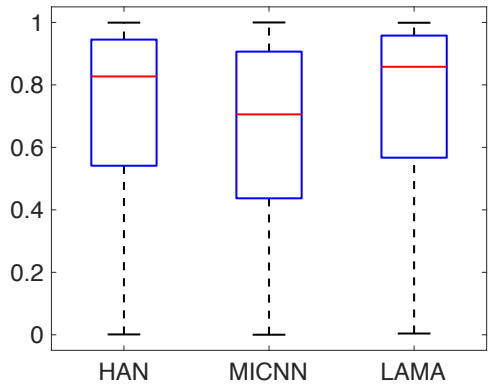
We also observe that the simple bilinear attention models outperform the dot product based attention models in HAN. For the CU English dataset, the BSA approach outperforms HAN & MIL-based baselines by 1.2% and 6.4% & 3.9% respectively. For the CU English and CU Spanish datasets we set $l = 2h$, whereas for MANSA dataset we set $l = 64$. These values were empirically found to perform best on the corresponding datasets.

3.4.2 Event Probabilities

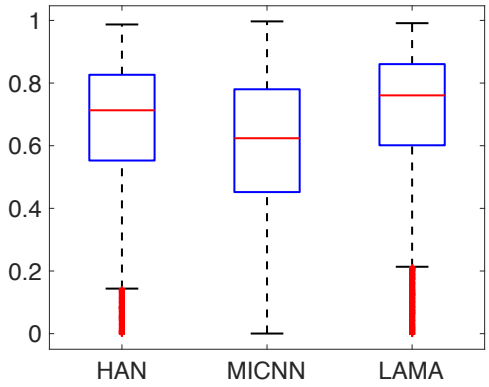
For event encoding it is important that models have a high precision otherwise it may lead to false-positives and incorrect representations. This is also an issue with traditional detection approaches that detect events based on occurrence of certain set of trigger words from a pre-curated list, without taking into account the sentence context or relationship between different entities. Hence, it is important for the models to assign a high confidence to positive articles and a low confidence to negative articles. We present a distribution of event probabilities assigned to the positive articles by the LAMA, HAN and MICNN models for the



(a) English



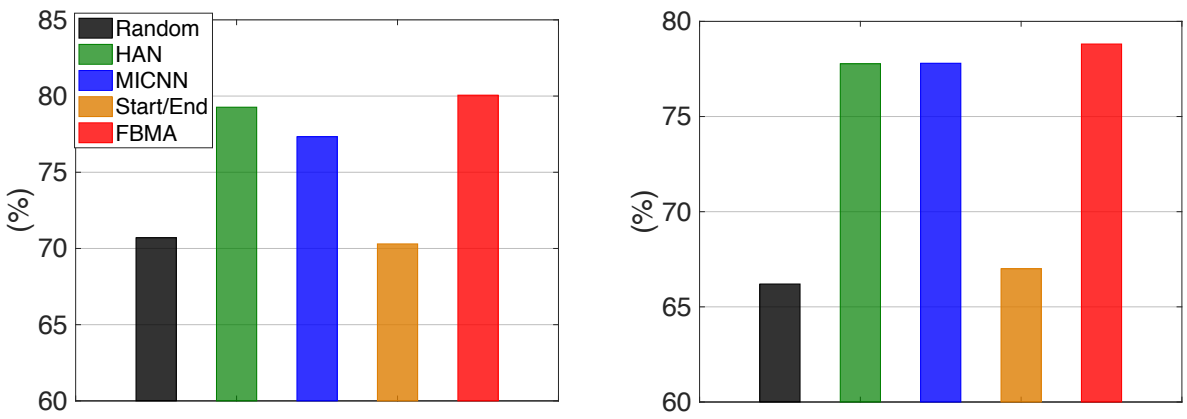
(b) Spanish



(c) MANSA

Figure 3.3: Comparison of event probabilities assigned by LAMA, MICNN and HAN on the CU English, Spanish and MANSA datasets. We can clearly see that mean probability is greater for LAMA in all the datasets for the event class. This depicts that LAMA is usually more confident than other methods in classifying the event articles. We also observe MANSA dataset contains some outliers (shown in red dots at the bottom).

test sets for all the three event datasets in Figure 3.3. We observe that generally the average probability assigned by LAMA is higher than MICNN and HAN confirming our hypothesis that having multi-aspect attention for each sentence increases the model confidence for a positive article.



(a) Population (b) Event

Figure 3.4: Event Extraction accuracy for event type detection and population class detection from key sentences extracted by different models. LAMA outperforms all the methods.

Table 3.5: Event Visualization. Green color highlights refer to Event Type keywords, Pink Color highlights refer to Population Keywords and Yellow Color highlights refer to Protest Related Keywords. These are top attended key words by an attention-based classifier picked from the top sentences selected by our model indicating that our model picks the most informative keywords.

Sentence	**Population Class**	**Event Type**
congress and left workers at a rally in kolkata to protest against post poll violence in the state	Labor	Government Policies
however traffic was affected in certain pockets within the city limit owing to road picketing members of trade unions staged demonstration picketed roads and attempted rail roko to show their protest against the central government	Labor	Employment & Wages
traffic movement in and around majestic area was disrupted for several hours following a rally organized by karnataka rajya raitha sangha demanding land for the poor and landless farmers the rally comprising over 5,000 members was flagged off by freedom fighter doreswam	Agriculture	Energy & Resources
hundreds of teachers of delhi university took to the streets on friday again to protest a university grants commissions ugc notification that could lead to around 5,000 temporary teachers losing their jobs	Education	Employment & Wages
Rawalpindi: The residents staged a protest demonstration against prolonged power outages and blocked Adiala road yesterday. The commuters and pedestrians faced hardships and vehicular movement remained suspended for many hours as violent protestors placed trolleys and trucks in the middle of the road. According to details a large number of residents...	General Population	Energy & Resources
dalit organizations have called for a bandh on wednesday to protest the brutal thrashing of the community youths in una	Ethnic	Government Policies

3.4.3 Event Extent and Trigger Extraction

One of the advantages of our approach is the ability to identify key sentences and trigger words. Sentences and trigger words with high attention weights can be used to enhance the recommendation models of semi-automated event encoding systems such as AutoGSR [78] and also increase the efficiency of human encoders. In this section we demonstrate how we use the event extents or the key sentences identified in the event detection stage to extract more relevant information about the event such as 'who' and 'why' of the event. Specifically, we demonstrate the use of our model to identify the population class and the event type from the summaries of the articles for the Civil Unrest (English) dataset. Each trigger word corresponding to the event type and population class might have multiple references throughout the article but might not be contributing to the event. Our hypothesis is that the key sentences extracted by our model would be more informative of event related information and hence would do a better job in identifying the relevant mentions.

3.4.4 Event Attribute Extraction

In the previous section we saw how we can extract event extents and triggers. In this section, we look into how we can extract event attributes such as population type and event type from the extracted event extents.

For a subset of articles in the CU English dataset, we have labels for the population type that participated in the protest, and the type of event. The population classes corresponding to one of the 11 classes are listed in Table 3.1, including 'Education', 'Labor', 'Legal' etc. The event type refers to the cause of the event and belong to one of the 5 event types like 'Government Policies', 'Economic Policies' etc. listed in Table 3.1.

Each news article is first summarized by selecting K_i sentences based on the article length,

where K_i is dynamically computed as $K_i = max(1, \lceil x_i \rceil * \eta)$ where η is a predefined value in $(0, 1]$. We set $\eta = 0.3$ which selects top 2 or 3 sentences from each article. Our goal is to predict the population class and event type labels from the summary of the article. This is a multi-class classification problem and we pick the HAN classifier for this task.

We train all models using 3-fold cross-validation with early stopping and use the same set of hyperparameters as for detection. Performance is evaluated based on the summaries chosen by our model against other baselines. We compare this approach to the following baselines: (i) *Random*, where we randomly select K_i sentences from an article as our summary. (ii) *Start/End*; News articles tend to organize key information in the beginning and end of the article so another baseline uses $\frac{K_i}{2}$ sentences from the top and $\frac{K_i}{2}$ sentences from the bottom of the article. (iii) *MI-CNN* We use the top K sentences ranked by the MI-CNN model and the HAN models as other baselines. Figure 3.4 shows the accuracy of the classifier trained on LAMA summaries vs. baselines. Classifier trained on LAMA summaries outperforms other baselines indicating that our model picks the most informative key sentences containing event related information.

For qualitative visualization we highlight the words assigned the highest attention weight in each sentence, by the binary attention-based classifier, the protest classifier and the population class classifier in Table 3.5. The words highlighted in pink indicate the population specific words, the words highlighted in yellow show the protest trigger words and the words highlighted in green show the protest type specific words or the 'why' of the event. As can be seen in Table 3.5 words such as 'teachers' for education population, 'residents' for general population, 'workers' for labor population indicating the 'who' of the protest are assigned the highest attention weights. The event type classifier attends to the words indicative of the reason for protest such as 'poll violence' for Government Policies, 'trade' for Employment and Wages, 'power outages' for Energy and Resources. The event detection classifier

overwhelmingly attends to protest trigger words such as 'protest', 'rally', 'strikes', 'demonstrations'. Tables 3.6 and 3.7 give a complete list of top 5 triggers attended from event spans extracted by LAMA for population type and protest type of the event.

3.4.5 Transfer Learning for Attribute Extraction

Inspecting Tables 3.6 and 3.7 it can be seen that population and protest type identification, both these tasks share common features. For instance. Employment and Wages(protest type) and Education (population type) both have the common feature "teacher", Housing (protest type) and Media (population type) have the common feature "policemen" and Legal (population type) and Government Policies (protest type) share the feature "advocates". This alludes to the fact that there is correlation between a few of these categories between both these tasks. To help our models pick up on this correlation we experiment with transfer learning where we first train a model on one task only and then initialize layers of the model for second task by the trained weights of the first task. Finally we finetune all the layers of the model for the second task till convergence. For both tasks we use the HAN model trained on $top-K$ sentences extracted by the LAMA model. The results are indicated in Table 3.8. When sentence GRU layers are transferred from the population type prediction model to protest type prediction model we find that we get the most performance improvement(1.1%) over no transfer learning. When we perform the reverse transfer i.e. transfer sentence GRU layers from trained protest classifier to population type classifier we find that the performance decreases.

Table 3.6: Top triggers extracted from sentences extracted by LAMA for each population type.

Media	Medical	Legal	Agriculture	Labor	Religious	Education	Ethnic
Journalists Police Assistant policemen	Doctors Medical Nurses Patients Pharmacies	Advocates Lawyers Bar Madras court	Farmers Raitha Kisan	Workers Employees Trade Congress unions	Hindu Muslim Catholic Christians Bishops	Students Teachers DU Parents	Maratha Dalit Jat Tribal

Table 3.7: Top triggers extracted from event spans extracted by LAMA for each protest type.

Housing	Energy &Resources	Employment &Wages	Government Policies	Economic Policies
Compensation Violence Evacuation Policemen	Cauvery Land Supply Power Karnataka	pay teachers salaries Congress Union	advocates government BJP arrest	price bank liquor loan

3.5 Related Work

In event extraction supervised approaches usually rely on manually labeled training datasets and handcrafted ontologies. Li et al. [51] utilize the annotated arguments and specific keyword triggers in text to develop an extractor. Supervised approaches have also been studied using dependency parsing by analyzing the event-argument relations and discourse of event interactions [64]. These approaches are usually limited by the availability of the fine-grained labeled data and required elaborately designed features. Different from these approaches our method uses attention mechanism to implicitly weigh words and sentences and is able to extract the event extents and trigger words with labels only at the document level. This formulation is suitable because labels at document level are easier to obtain than per-sentence level or word level. This makes the task of event extraction also amenable to Multiple Instance Learning (MIL) [21] solutions. In MIL 'bags' are groups of 'instances' which are to

Table 3.8: Results of transferring different layers of the HAN model trained on the Population Type prediction task to Protest Type prediction task. For protest type prediction different layers as list in the first column were initialized from the corresponding weights in the population type model followed by finetuning the model.

Transferred Layer	Accuracy
No Transfer Learning	81.5 %
word GRU layers + sentence GRU layers	81.0 %
sent GRU layers + sentence attention layers	81.9 %
sent GRU layers + word attention layers	82.3 %
sent GRU layers	**82.4**%

be classified. In standard MIL formulation individual instance level labels are not available and labels are provided only at the group/bag level. Each bag is labeled positive if it contained at least one positive instance and negative otherwise. Kotzias et al. [45] focus on instance-level predictions from group level labels and allow for the application of general aggregation functions for sentiment classification. Wang et al. [92] use a similar idea and have the previous state-of-the-art results on one of the datasets we evaluate on. Contrary to these approaches our method is hierarchical and computes the feature representation for the next level in the hierarchy using a weighted average of feature representations in the current layer [84].

3.6 Discussion

In this chapter, we presented a framework for event detection and extraction. We used hierarchical attention based models for the task because of their ability to attend to words and sentences while constructing sentence and document representations respectively. With the hierarchical models we used our proposed LAMA mechanism which computes multiple attention distributions over words which leads to contextual sentence and document rep-

resentations. Our results showed that this mechanism performed better than several other approaches and especially single-head mechanisms that miss out on the context because there is only one attention distribution. We saw how LAMA extracted event-spans were the most informative for population and protest type prediction. Finally, we saw how transfer learning between related tasks can help improve subtask results.

Chapter 4

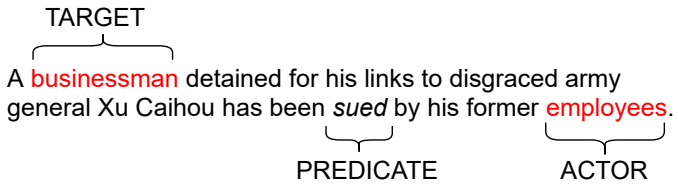
Event Extraction

In the last chapter, we saw how we can use hierarchical attention-based models for event detection(§ 3.2) and event extent and trigger extraction(§ 3.4.3). In this chapter, we look into event extraction. Specifically, we propose a solution to address the long-range dependency problem in event extraction systems introduced in Section 1.5. We first motivate the paradigm of MRC-based event extraction(§ 4.1), then we propose a context simplification algorithm(§ 4.2.1) to mitigate the long-range dependency problem and finally we present the results in Section 4.5.

4.1 Introduction

Traditional methods of event extraction rely on techniques such as syntactic parsing, inducing frame-roles etc. Recent successful approaches to event extraction are based on dense features extracted by neural models [13, 56, 69] as well as contextualized lexical representations from pretrained language models [88]. These approaches also however, rely on pipeline entity extraction systems for argument extraction which result in error propagation from downstream tasks.

On the related task of semantic role labeling, to alleviate the need for predefined frames or thematic role ontologies, researchers have proposed to formulate it as a question answering task [30]. For each predicate in the sentence, they propose to label it with a question/answer



Q-Actor	Who *sued* someone?
Q-Target	Who was *sued* by someone?

Figure 4.1: An example of an event of the type 'Bring lawsuit against' from the ICEWS dataset (left). A generated QA record for that event (right).

pair. For example, the verb "propose" in the previous sentence would be labeled with the questions "What is proposed?", and "Who proposed something?", each paired with the phrase from the sentence that gives the correct answer.

Moreover, great progress has been made recently in reading comprehension facilitated mainly by the success of large pretrained language models such as BERT [17] and its variants [48, 57]. These independent developments make an appealing case for modeling event extraction as a span-based extractive reading comprehension (MRC) task. Besides alleviating the problem of error propagation due to pipeline systems MRC systems can also be used as zero-shot event extractors. Figure 4.1 gives an example event from the ICEWS dataset(§ 4.3) and the corresponding QA-record generated for that event.

In the recent past, a few studies have attempted to model event extraction as a MRC task [26, 55]. The input to the MRC model is the event extent and a question corresponding to each argument and the task is to extract a span from the sentence as an answer to that question. These approaches mainly focus on generating questions corresponding to different argument roles that adapt to the context. The aim of these studies is to generate questions that take background information from context into account and create an overlap between the tokens in the context and the question. However, these systems may still result in non-natural, syntactically incorrect questions.

Recent studies have also observed that MRC-based event extraction systems struggle from

the long-range dependency problem where there is a large distance between a trigger word and its argument in terms of number of words. For instance, Liu et al. [55] observe that one typical error from their MRC-based extraction system is related to long-range dependency between an argument and a trigger, accounting for 23.4% errors on the ACE-2005 event dataset [24] (here "long-range" denotes that the distance between a trigger and an argument is greater than or equal to 10). Du and Cardie [26] observe that one of the failure modes of their extraction system is sentences with complex sentence structures containing multiple clauses, each with trigger and arguments (such as when triggers are partial or elided).

Moreover, scaling event extraction to new domains is challenging due to difficulty in collecting good quality labeled training data. MRC-based event extraction offers a promising way forward by enabling zero-shot event extraction.

In this chapter, we first propose a method for generating questions grounded on the event type for extracting event argument roles. Next, to mitigate the long-range dependency problem in MRC-based EE systems and to make sentences syntactically less complex we propose an unsupervised sentence simplification approach.

Our premise is that MRC models will benefit from simple sentences and reduced syntactic complexity and perform better or on par on simple sentences as compared to original sentences. To reduce the complexity and to perform simplification we propose an unsupervised approach that is guided by a scoring function that incorporates syntactic fluency, simplicity and the performance of the MRC model. Moreover, we seek to investigate if the simplification model can be guided by the said MRC model in a reward-based framework. Our analysis sheds light on the brittleness of ubiquitous models such as BERT. Specifically, we find that language prior can still help the models answer questions without having the complete details in the context. We perform follow-up analyses and show that simplification can be controlled based on desired factors such as preference for high performance for a certain

argument role. Figure 4.2 gives an overview of the proposed approach.

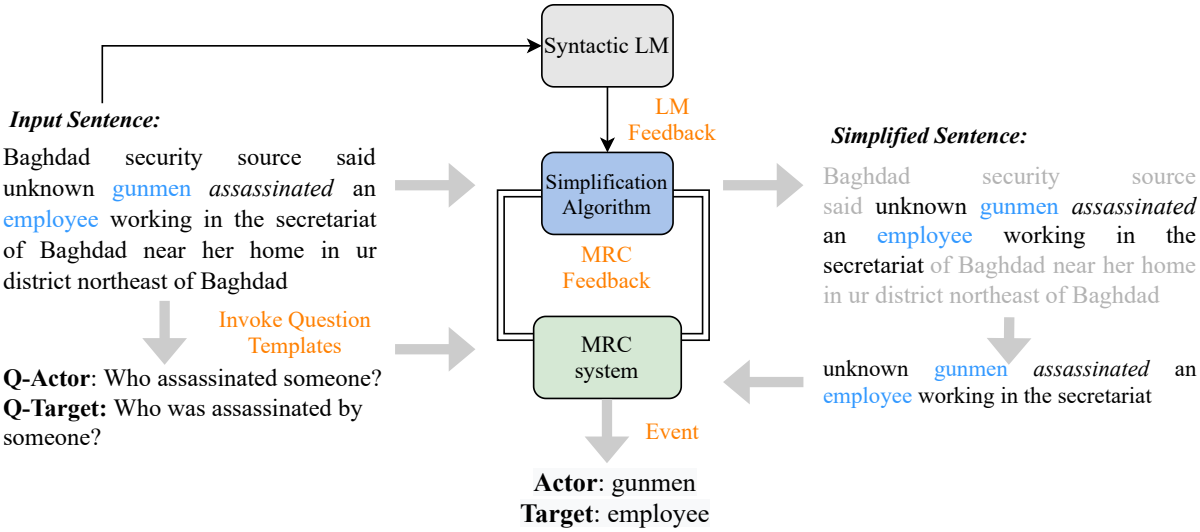


Figure 4.2: Proposed approach.

4.2 Methodology

Reading comprehension models can be brittle to subtle changes in context. They can be thrown-off by syntactic complexity, especially when the questions are not specific and do not include words overlapping with the context. Moreover, long range dependencies between the trigger/predicate and the argument can also throw the model off. For this purpose, we propose an M**R**C-guided **u**nsupervised **s**entence **s**implification algorithm (RUSS), that iteratively performs deletions and extractions from the context in search for a higher-scoring candidate. The score function incorporates components that ensure sentence fluency, information preservation and the confidence of the target MRC model.

The algorithm assumes that an event has been detected in the sentence and the task is to identify the arguments of the detected event. For instance, for the example shown in Fig. 4.1 the task is to identify the arguments 'Actor' and 'Target' of the event 'Bring lawsuit

Table 4.1: Table lists the ICEWS event types used and their corresponding predicates that were identified for generating question templates.

Event Type	Predicates
Abduct, hijack, or take hostage	kidnapped, abducting, abducted, captured
Accuse	blame, blaming, accused, alleged, accusing
Apologize	apologize, apology
Assassinate	carried out assassination of, assassinate
Bring lawsuit against	is suing someone, sued, has sued, filed a suit against
Demonstrate or rally	condemn, protest, demonstrate
Arrest, detain, or charge with legal action	arrested, sentenced, detained, nabbed, captured, arresting, capture, jailed, routinely arrested, prosecuted, convicted
Use conventional military force	killed, shelled, combating, shells, strikes, strike, kill

against'. The algorithm also assumes that QA pairs corresponding to each argument role to be extracted are available. The QA generation procedure for the dataset used for evaluation is outlined in section 4.3.1. For the RUSS algorithm we assume that we have a record that contains a question for each argument role for an event in a sentence. Input to RUSS is a sentence(event extent) and a question corresponding to each argument to be extracted. RUSS outputs a simplified sentence.

4.2.1 Unsupervised Context simplification

Given an input sentence s and a list of questions $\{q_1, ..., q_n\}$ corresponding to different arguments, our algorithm iteratively performs deletions and extractions in search for a higher-scoring candidate and outputs a simplification c. For generating candidates, the algorithm first obtains the constituency parse tree of the context using a span-based constituency parser [36]. It then sequentially performs two operations on the parse tree – deletion and extraction.

Deletion

In this operation, the algorithm sequentially drops subtrees from the parse tree corresponding to different phrases. Note that the subtrees with the NP (Noun-Phrase) label are omitted because it is expected that many entities will be noun phrases and deleting them from the sentence would result in significant information loss.

Extraction

This operation simply extracts a phrase, specifically corresponding to the the S and SBAR labels as the candidate sentence. This allows us to select different clauses in a sentence and remove remaining peripheral information.

These operations generate multiple candidates. Candidates with fewer than a threshold of t words are filtered out. We heuristically determine $t = 5$. From the remaining candidates, a highest-scoring candidate is chosen based on the score function described in the next section(§ 4.2.2). The algorithm terminates if the maximum score assigned to a candidate in the current iteration does not exceed the previous maximum score. The simplification algorithm RUSS is outlined as Algorithm 1 and the candidate generation algorithm is outlined as Algorithm 2.

4.2.2 Scoring function

We score a candidate simplification as a product of different scores corresponding to fluency, simplicity and its amenability to the downstream MRC model.

Algorithm 1 Context Simplification Algorithm – RUSS

Input: $sentence := s$, $questions = \{q_1, ..q_n\}$
Output: $simplification := c$
Function `RUSS(`s`)`:
  $maxIter \leftarrow M$
  for $iter \in maxIter$ **do**
    $candidates \leftarrow$ `generateCandidates`(c)
    $scores \leftarrow \emptyset$
    $maxScore \leftarrow 0$
    for $cand \in candidates$ **do**
      $scores \leftarrow scores \cup \nu^a_{lm} * \nu^b_{entity} * \nu^c_{pred} * \prod \nu_{rc^{r_i}_{role_i}}$
    end
    $currMax \leftarrow max(scores)$
    if $currMax > maxScore$ **then**
      $maxScore \leftarrow currMax$
      $c \leftarrow candidates[argmax(scores)]$
    end
end
return c

LM Score (ν_{lm})

This score is designed to measure the language fluency and structural simplicity of a candidate sentence. Instead of using LM-perplexity we use the syntactic log-odds ratio (SLOR) [10, 70] score to measure the fluency. SLOR was also shown to be effective in simplification to enhance text readability [37, 46]. Given a trained language model (LM) and a sentence s, SLOR is defined as

$$SLOR(s) = \frac{1}{|s|}(ln(P_{LM}(s)) - ln(P_U(s)) \tag{4.1}$$

where P_{LM} is the sentence probability given by the language model, $P_U(s) = \prod_{w \in s} P(w)$ is the product of the unigram probability of a word w in the sentence, and $|s|$ is the sentence length. SLOR essentially penalizes a plain LM's probability by unigram likelihood and the length. It ensures that the fluency score of a sentence is not penalized by the presence of rare words. A probabilistic language model (LM) is often used as an estimate of sentence fluency. In our work, instead of using a plain LM we use a syntax-aware LM, i.e., in addition to words, we use part-of-speech (POS) and dependency tags as inputs to the LM [101]. For

Algorithm 2 Candidate Generation Algorithm

Input: *sentence* := s
Output: *candidates*
Function `generateCandidates(`s`)`:
  $parseTree \leftarrow getParseTree(s)$
  $toRemove \leftarrow \emptyset$
  $extractions \leftarrow \emptyset$
  $candidates \leftarrow \emptyset$
  $phraseTags \leftarrow getValidPhraseTags()$
  for $pos \in parseTree.positions$ **do**
    if $parseTree[pos] \in phraseTags$ **then**
      $toRemove \leftarrow toRemove \cup parseTree[pos].leaves$
    end
    if $pos.label \in [S, SBAR]$ **then**
      $extractions \leftarrow extractions \cup parseTree[pos].leaves$
    end
  end
  for $phrase \in toRemove$ **do**
    $candidate \leftarrow s.replace(phrase, \emptyset)$
    if $candidate.length > t$ **then**
      $candidates \leftarrow candidates \cup candidate$
    end
  end
  for $phrase \in extractions$ **do**
    if $phrase.length > t$ **then**
      $candidates \leftarrow candidates \cup candidate$
    end
  end
  return *candidates*

a word w_i , the input to the syntax-aware LM is $[e(w_i); p(w_i); d(w_i)]$, where $e(w_i)$ is the word embedding, $p(w_i)$ is the POS tag embedding, and $d(w_i)$ is the dependency tag embedding. Note that our LM is trained on the original train corpus. Thus, the syntax-aware LM helps to identify candidates that are structurally ungrammatical.

Entity Score (ν_{entity})

Entities help identify the key information of a sentence and therefore are also useful in measuring meaning preservation. The desired argument roles are also entities. Thus, if any entity detected in the original sentence is omitted from a candidate the entity score for that candidate is 0, else it is set to 1.

Predicate Score (ν_{pred})

This score preserves the event predicates in a candidate simplification. It checks if a candidate contains any predicate of interest corresponding to the event detected (Table 4.1. If it does not then ν_{pred} is set to 0, else it is set to 1.

MRC Score (ν_{rc})

Transformer-based MRC models can be brittle to subtle changes in context. To make the context robust to the MRC model this score allows us to control the complexity of context with respect to the confidence of the MRC model. It is computed separately for each role. Each argument of an event is a span in the context. $\nu_{rc^{r_i}_{role_i}}$ is the score of the best span in the context for the argument role i, where the score of a candidate span is defined as $S \cdot T_x + E \cdot T_y$ where $S \in R^H$ is a start vector and $E \in R^H$ is an end vector as defined in Devlin et al. [17]. T_x and T_y are the final layer representations from the BERT model of the x^{th} and y^{th} tokens in the context. Note that for a valid span, $y > x$. This score is computed separately for each argument role (Actor and Target in Example 4.1). The importance of the i^{th} role can be controlled by the exponent r_i. The total contribution of each role is computed as the product of score corresponding to each role, given by $\prod \nu^{r_i}_{rc_{role_i}}$.

The final score of a candidate c is computed as follows:

$$\nu(c) = \nu_{lm}(c)^a * \nu^b_{entity}(c) * \nu^c_{pred}(c) * \prod \nu^{r_i}_{rc_{role_i}}(c) \tag{4.2}$$

Note that b, c can be either 1 or 0 since ν_{entity} and ν_{pred} are binary. In later sections, we evaluate how the simplification can be controlled by varying the constants r_i's.

4.2.3 Reading Comprehension Model

For the supervised experiments we finetune a BERT-based reading comprehension model for event extraction. Let $q = \{q_1, q_2, ...q_n\}$ be the query and $c = \{c_1, c_2, ...c_n\}$ be the context. First q and c are jointly encoded to learn input representations by constructing a sequence "$[CLS]q[SEP]c$" as input to BERT. Given the joint representation H_c^q of q and c, two probability vectors containing the start and end positions of the answer over every position in c are computed.

$$p_{start}^c = softmax(H_q^c W_{start}) \tag{4.3}$$

$$p_{end}^c = softmax(H_q^c W_{end}) \tag{4.4}$$

The score of a span i, j is given by $p_{start}^i + p_{end}^j$. The highest scoring span is selected as the answer with the constraint that the position of start index should be smaller than end index i.e. $i < j$.

We finetune all the layers of BERT with an initial learning rate of 3e-5 with the AdamW optimizer [58] on four Nvidia V100 GPUs with early stopping and $patience = 5$.

4.3 Datasets

We evaluate RUSS on the ICEWS event dataset[1] from years 2013 to 2015. Event data consists of coded interactions between socio-political actors (i.e., cooperative or hostile actions between individuals, groups, sectors and nation states). Events are automatically identified and extracted from news articles by BBN-SERIF(statistical Entity and Relation Information Finder) encoder to identify triplets of type actor-subject-target from article text. Note

[1] https://dataverse.harvard.edu/dataverse/icews

that SERIF is not publicly available. These triples consist of a source actor, an event type (according to the CAMEO [2] taxonomy of events), and a target actor. From the ICEWS dataset, we aim to extract the Actor and Target roles for event types shown in Table 4.1. We use the data from years 2013-2015 as train data and 2016 for test in section 4.5.3. See Appendix for distribution of event types.

4.3.1 Question-Answer pair Generation

For the ICEWS dataset, we create one question template per predicate per event type for each slot. For each event type we identified a list of most common predicates (triggers) for that event type since trigger labels are not available in the ICEWS dataset. For example, for 'Demonstrate or rally' event type the predicates identified are 'condemn', 'protest', 'demonstrate' and for 'Accuse' event type the predicates are 'blame', 'blaming', 'accused', 'alleged', 'accusing'. Table 4.1 enumerates all of the ICEWS event types used and their identified predicates. For each of the predicates identified for each event type we use one question template for each of the two argument roles Actor and Target. For the Actor role, the template used an active construction 'Who $predicate$ someone? and for the same event for the Target role the templated used a passive construction – 'Who was $predicate$ by someone?'. This results in 37,894 records with a sentence and two questions one each for the Actor and Target roles respectively. The list of entities for the ICEWS dataset comes from the ICEWS actors and agents dictionaries[3].

[2] https://parusanalytics.com/eventdata/data.dir/cameo.html
[3] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/28118

4.4 Evaluation

We evaluate the performance of an MRC system before and after context simplification in the cross-domain zero-shot setting. In this setting, we do not finetune a pretrained MRC model with the generated QA dataset. Rather, the aim is to assess the model performance in a zero-shot setting, without using any training data from the target domain whatsoever. We used the pretrained BERT model finetuned on the SQUAD 2.0 dataset [75] and use the predictor API provided here [4]. We further conduct follow-up analysis to study the controllability of simplification by performing ablation analysis and assessing model performance for different values of score component coefficients. For evaluation, we extracted the best span(s) and computed an exact match F1 score [81] matching the span against the ground truth answer.

4.5 Results

The results of zero-shot extraction on the ICEWS dataset are layed out in Table 4.2. In the baselines used, simplification is performed with score function exponents for ν_{lm} as $a = 1.5$ and ν_{entity} as $b = 1$ held constant while varying c for ν_{pred}, r_1 for ν_{actor} and r_2 for ν_{target}. With no simplification we get F1 scores of 0.412 and 0.354 for actor and target roles respectively. For the most basic setting for simplification with $c = 0$, $r_1 = 1$ and $r_2 = 1$ scores improve by 4.6% for actor prediction to 0.431 and by 10.4% to 0.391 for target prediction respectively which shows that simplifying context can still help a powerful model like BERT in a cross-domain zero-shot setting. For actor prediction, out of 37,894 records we find that for 10.99% records, F1 score improves after simplification, for 6.54% records F1 decreased after simplification and for the rest the score remained unchanged. For target prediction, for 17.4% records scores improve where as for 7.9% records the scores decreased and for the

[4] https://docs.allennlp.org/models/v2.4.0/models/rc/predictors/transformer_qa/

rest of the records, the scores remained unchanged. After introducing the predicate score ($c = 1$) we see that these improvements drop slightly. This is counter-intuitive, because one would expect model performance to improve when relevant predicates are present in the context. We attribute this behavior to the MRC model leveraging the language priors in the training data to predict the answers. For instance, the model could predict the subject of the predicate as an answer for 'Who' type of questions.

Next, we increase the coefficients of Actor and Target roles from 1 to 3. The reason why we choose an odd number for this exponent is because sometimes for bad candidates the MRC scores can be negative and since all the scores are combined in a multiplicative way, raising a negative score to an even power would reverse the desired effect. Observing the results in rows 5 & 6 of Table 4.2 we can see that percentage of sentences with similar scores before and after simplification have increased. We can also observe that percentage of sentences for which scores decrease after simplification have also decreased. We can conclude that by raising the coefficients of role specific scores we can make the simplification models more robust to inaccurate simplifications for those roles. We also observe, when $r_1 = 3$, we get the highest F1 for actor prediction, an improvement of 5.6% over no simplification and for $r_2 = 3$ we can an F1 on-par with the highest obtained in row 2. Our results clearly indicate the benefit of simplification over no simplification and also the gradual improvement in scores when the argument coefficients r_1, r_2 are varied from 0 to 3.

4.5.1 Long Range Dependencies

Fig: 4.3 shows the distribution of lengths of sentences before and after simplification.

Mean length of the original sentences is 32 words where as mean length of the sentences after simplification is 22 words (row 2 setting). This indicates that simplification doesn't

Table 4.2: Results of zero-shot event extraction on the ICEWS dataset. ν_{lm} coefficient $a = 1.5$ and ν_{entity} coefficient $b = 1$ for all settings in which simplification is performed. Δ $+ve$ indicates the % of records for which F1 improves after simplification, Δ $-ve$ indicates the % of records for which F1 becomes worse after simplification and Δ same indicates the % of records for which F1 remains unchanged.

		Actor				**Target**			
	Method	F1	Δ $+ve$	Δ $-ve$	Δ same	F1	Δ $+ve$	Δ $-ve$	Δ same
1	No simplification	0.412	-	-	-	0.354	-	-	-
2	$c = 0, r_1 = 1, r_2 = 1$	0.431	**10.99**%	6.54 %	82.45%	0.391	**17.35**%	7.9%	74.9%
3	$c = 1, r_1 = 0, r_2 = 0$	0.429	10.81%	6.57 %	82.61%	0.390	16.54%	7.53%	75.93%
4	$c = 1, r_1 = 1, r_2 = 1$	0.424	10.5%	6.3 %	83.1%	0.387	16.29%	7.64%	76.05 %
5	$c = 1, r_1 = 3, r_2 = 0$	**0.435**	9.72%	**5.67**%	**84.6**%	0.391	16.89%	7.97%	75.12%
6	$c = 1, r_1 = 0, r_2 = 3$	0.427	10.54%	6.95%	82.5%	**0.391**	16.12%	**7.29**%	**76.59**%

make sentences too short as is intuitive because cutting relevant information would harm the performance.

Let's see if simplification has addressed the long-range dependency problem. We look at statistics concerning the distance between the predicate and its arguments (Actor and Target) for the setting $c = 0, r_1 = 1, r_2 = 1$, that is, when the predicate score(ν_{pred}) is not taken into account. As Table 4.2 indicates for 11% of the records performance increases after simplification. We find that for those records the average distance between the predicate and its argument Actor is about 13 words and the average distance between the predicate and target in the simplified context is about 10 words. For the argument Target the average distance between the predicate and target is about 8 words for original and about 6 words for the simplified context.

We see that RUSS cuts about 3 words for Actor prediction and 2 words for Target prediction on an average. We conclude that a certain percentage of improvement comes from cutting down the distance between the predicates and arguments hence mitigating the long-range dependency problem.

Moreover, we observe that actor prediction performance is better than target prediction. This could be attributed to the fact that for sentences that are in active construction, the subject of the predicate is a candidate for the actor and in active constructions the distance between the predicate and its subject will be small. However, for a complex sentence a clause or a phrase can occur between the subject and the predicate. In this case, the distance between them increases and hence we expect the performance to go down. We quantitatively verify this as follows – the average length in the simplification between the actor and predicate for which the scores improve is about 7 words and for which the scores decreases is about 8 words. Hence, whenever the distance between the predicate and actor is large the performance tends to become worse.

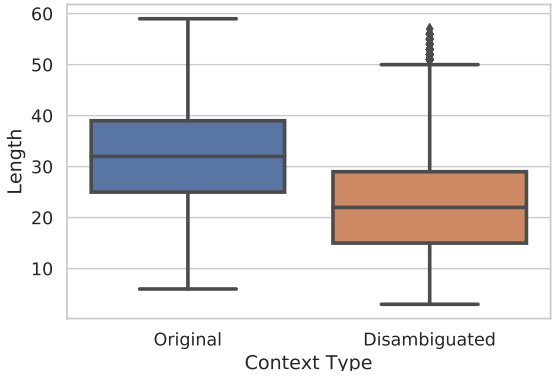


Figure 4.3: The length distributions of sentences before and after simplifications.

4.5.2 Qualitative Analysis

Let's look at some failure cases and some successful simplifications of the proposed method. Table 4.3 lists some cases in which simplification helps MRC system perform better. In the first example, the original context was "Islamabad prime minister Nawaz Sharif personally apologized to the opposition today for what he called unfortunate comments made by Chaudhry Nisar Ali khan against PPP's Aitzaz Ahsan", our model simplified the context

to "Islamabad prime minister Nawaz Sharif apologized to the opposition today for what he called unfortunate comments made against PPP's Aitzaz Ahsan". The question posed to the zero shot MRC model was "Who is being apoloized to by someone" and the ground truth answer is "the opposition". For the original context the model extracts "Nawaz Sharif" as the answer which is the wrong answer, whereas after removing the adverb "personally", it gets the correct answer. Note that this is decreasing the distance between the predicate *apologized* from its argument Nawaz Sharif. We also observe that after removing "Islamand" as well MRC model extracts the correct answer. This example highlights the brittleness of the ubiquitous Transformer based models and underlines the need for the simplification approach proposed. In the second example, MRC model extracts the closest noun "Xu Caihou" as answer which is incorrect. simplification deletes the PP "to disgraced army general Xu Caihou" aiding the MRC model in extracting the correct answer. Note that in this case it was especially important to delete the above phrase due to the inherent ambiguity of the construction. This case also highlights the limitations of the current MRC systems as the system was not able to successfully associate employees with businessman and predicted the NP closest to the predicate *sued*.

Error Analysis

From 6.54% records for which the score decreased after simplification (row 2 of Table 4.2) for 39.5% records from those the prediction from the original context is a substring of the prediction from the simplified context. This means that for some cases, both the original and the simplified context facilitate the correct answer, rather it is the case that the answer from the simplified context contains extra information for which it is penalized during F1 score computation. For example consider the context "baghdad security source said unknown gunmen assassinated an employee working in the secretariat of baghdad near her home in ur

district northeast of baghdad" which after running the simplification algorithm is shortened to "~~in baghdad security source said~~ unknown gunmen assassinated an employee working in the secretariat ~~of baghdad near her home in ur district northeast of baghdad~~". The strikethrough text represents the text deleted by the proposed algorithm. For the question; "Who was assassinated by someone?" when presented with the original context the MRC model extracts "an employee" whereas after removing the strikethrough text, MRC model extracts "an employee working in the secretariat". The ground truth answer for this is "employee". As can be seen both answers are correct but the simplified contex is penalized for extra words. Interestingly, such cases make up 48% of records for cases for which performance improves after simplification. This is intuitive, because since context becomes shorter and more precise after simplification and hence one expects MRC models to extract more precise answers. The fact that this happens in 39.5% cases in the reverse scenario is surprising.

4.5.3 In-Domain Training

In sections 4.5.1- 4.5.3 we saw how RUSS improved zero-shot event extraction performance in the cross-domain data augmentation setting. In this section, we consider the scenario when we have labeled in-domain training data available and we wish to investigate if simplification still helps improve performance when the MRC system has been finetuned on in-domain data. We use the BERT-base-cased model [17] as our base model and finetune it on the ICEWS train dataset. We finetune all layers as opposed to just the classification layer as we observe large improvement in the former case as compared to the latter. We use an initial learning rate of 3e-5 and use early stopping with *patience* = 5 to find the best model. For training we use the ICEWS dataset from years 2013-2015 and the year 2016 for testing. The QA generation procedure is described in section 4.3.1. There are total 75,788 (37,894×2) examples for training and 5,906 (2,953×2) for test. BERT-MRC in Table 4.4

Table 4.3: Qualitative examples of zero-shot performance of MRC model before and after simplifying the context using the proposed algorithm. Underlined words are ground truth answers, emphasized words are predicates(triggers) and strikethrough indicates that words were removed by the algorithm.

Question	Who is being apologized to by someone?
Sentence	Islamabad prime minister Nawaz Sharif personally *apologized* to the opposition today for what he called unfortunate comments made against PPP's Aitzaz Ahsan
Answer	Nawaz Sharif
Simplified	Islamabad prime minister Nawaz Sharif ~~personally~~ *apologized* to the opposition today for what he called unfortunate comments made against PPP's Aitzaz Ahsan
Answer	the opposition
Question	Who is being sued by someone?
Sentence	Scmp a businessman detained for his links to disgraced army general Xu Caihou has been *sued* by his former employees
Answer	Xu Caihou
Simplified	~~Scmp a~~ businessman detained for his links ~~to disgraced army general Xu Caihou~~ has been *sued* by his former employee
Answer	businessman
Question	Who is being accused of something?
Sentence	Thus after having attacked the two elected to his party ump Brice Hortefeux and Claude Goasguen it was accused of pressure and insults. Rachida Dati has *accused* Claude Goasguen to take to her because she had refused to sleep with him and this during an altercation proved by the Canard Enchan.
Answer	Rachida Dati
Simplified	~~Thus after having attacked the two elected to his party ump Brice Hortefeux and Claude Goasguen it was accused of pressure and insults.~~ Rachida Dati has *accused* Claude Goasguen ~~to take to her~~ because she had refused to sleep with him and this during an altercation proved by the Canard Enchan.
Answer	Claude Goasguen

indicates the performance of the model on the original test set. We use the RUSS algorithm to obtain simplifications of the test set. BERT-MRC-Simple indicates the performance of the model on this simplified test set. It can be observed that simplification brings about an improvement(1.4%) even on a model that's finetuned on in-domain data.

Table 4.4: Table shows the performance of a BERT-base-uncased model finetuned on in-domain dataset. It can be seen that even after finetuning, RUSS approach improves model performance (BERT-MRC-Simple).

Model	F1
BERT-MRC	0.776
BERT-MRC-Simple	**0.787**

4.6 Related Work

Event extraction(EE) has been an active area of research in the past decade. In EE, supervised approaches usually rely on manually labeled training datasets and handcrafted ontologies. Li et al. [51] utilize the annotated arguments and specific keyword triggers in text to develop an extractor. Supervised approaches have also been studied using dependency parsing by analyzing the event-argument relations and discourse of event interactions [64]. These approaches are usually limited by the availability of the fine-grained labeled data and required elaborately designed features. Recent work formulates event argument extraction as an MRC task. A major challenge with this approach is generating a dataset of QA pairs. Liu et al. [55] propose a method combining template based and unsupervised machine translation for question generation. Du and Cardie [26] follow a template approach and show that more natural the constructed questions better the event extraction performance. However, none of these methods directly aim to address the long-range dependency problem using simplification.

Automatic text simplification (ATS) systems aim to transform original texts into their lexically and syntactically simpler variants. The motivation for building the first ATS systems was to improve the performance of machine translation systems and other text processing tasks, e.g. parsing, information retrieval, and summarization [11]. In the context of extraction, Zhang et. al. [100] show that pruning dependency trees to remove irrelevant structures

can improve relation extraction performance. Efforts have been made to incorporate syntactic dependencies into models in an effort to mitigate this problem 2018, 2020, 2016. Recently, Mehta et al. [65] have used sentence simplification as a preprocessing step for improving machine translation. Edit-based simplification has been investigated to a great degree to improve the readability of the text [2, 25, 46]. To the best of our knowledge this is the first work that studies sentence simplification for improving MRC-based event extraction.

Chapter 5

Conclusion

In this dissertation, we studied three problems and current state-of-the-art approaches to event extraction in Section 1.5 and proposed novel solutions to each of the problems. Below is the conclusion of this dissertation with respect to the goal and contributions described in Section 1.6:

1. (Chapter 2) We performed a comprehensive analysis of attention mechanisms as used for text classification. We introduced a compact multi-head attention mechanism and illustrated its effectiveness on text classification benchmarks. The proposed method computes multiple attention distributions over words which leads to contextual sentence representations. The results showed that this mechanism performed better than several other approaches with fewer parameters. We also verified that the model captured context-dependent word importance.

2. (Chapter 3) We presented a hierarchical MIL framework for event detection. We used attention based models for the task because of their ability to attend to words and sentences while constructing sentence and document representations respectively. Within the hierarchical models we used LAMA attention mechanism proposed in Chapter 2 which computes multiple attention distributions over words which leads to contextual sentence and document representations. Our results showed that this mechanism when applied at the event extent level makes for rich representations as shown by the ef-

fectiveness on downstream task of event extraction (specifically, population type and protest type prediction). Finally, we saw how transfer learning between related tasks can help improve subtask performance.

3. (Chapter 4) We demonstrated how the long-range dependency problem ubiquitously faced by event extraction models including MRC-based formulations can be addressed by using syntactic disambiguation specifically in a zero-shot setting. We also demonstrated how this disambiguation can be guided by the MRC system itself and can be controlled for different argument roles.

5.1 Future work

We highlight some of the immediate future directions below. With respect to improving the mechanism LAMA and hence further improving event detection performance, we envision three directions for future work;

1. Currently, LAMA relies on RNNs that makes it slower due to their sequential nature computation. We seek to investigate ways of adapting the proposed mechanism in self-supervised language models without dependency on RNNs by incorporating positional embeddings [87] for faster and efficient learning of language representations;

2. Currently, the model uses a single global context vector as the query vector that conflates the entire sentence into one vector which could lead to information loss. Transformer models on the other hand use query every token in the sequence for each candidate token. Even though this may help develop direct connections between relevant words it might get redundant. In the future work, we seek to investigate ways of incorporating phrase-level queries as a middle ground between a single global

context vector like the proposed approach and fine-grained queries like Transformer providing a balance between complexity and context

3. Another important research question concerns with the making the different attention heads more discernible with respect to each other for better interpretability. One of the obstacles in learning attention in an unsupervised way is that there is no implicit mechanism to impose structure on different rows and it merits further research.

With respect to improving the the proposed MRC-guided disambiguation approach we envision the following directions for future work.

1. Albeit, the proposed method can be useful when offline computations can be afforded, it may be difficult to deploy such systems in real-time use cases. Reasons include – 1) MRC system inference time while running the algorithm; 2) call latency if using APIs and 3) search algorithms such as beam search are sequential and can be computationally expensive. A more efficient way of performing syntax disambiguation would be guide the MRC model during the pretraining or finetuning stage. This can be done by using attention masks over the deleted words obtained using the the disambiguation algorithm.

Bibliography

[1] David Ahn. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia, July 2006. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W06-0901.

[2] Fernando Alva-Manchego, Joachim Bingel, Gustavo Paetzold, Carolina Scarton, and Lucia Specia. Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://www.aclweb.org/anthology/I17-1030.

[3] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. 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 344–354, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1034. URL https://www.aclweb.org/anthology/P15-1034.

[4] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2017.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv e-prints*, abs/1409.0473, September 2014. URL https://arxiv.org/abs/1409.0473.

[8] Elizabeth Boschee, Premkumar Natarajan, and Ralph Weischedel. *Automatic Extraction of Events from Open Source Text for Predictive Forecasting*, pages 51–67. Springer New York, New York, NY, 2013. ISBN 978-1-4614-5311-6. doi: 10.1007/978-1-4614-5311-6_3. URL https://doi.org/10.1007/978-1-4614-5311-6_3.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[10] John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. Simplifying text for language-impaired readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 269–270, Bergen, Norway, June 1999. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E99-1042.

[11] Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1041–1044. Association for Computational Linguistics, 1996.

[12] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[13] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 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, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1017. URL https://www.aclweb.org/anthology/P15-1017.

[14] Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1267–1276, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1158. URL https://www.aclweb.org/anthology/D18-1158.

[15] Junyoung Chung, Caglar Gulcehre, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[16] M. Denil, L. Bazzani, H. Larochelle, and N. Freitas. Learning where to attend with deep architectures for image tracking. *Neural Computation*, pages 2151–2184, 2012.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[19] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[20] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016.

[21] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2): 31–71, 1997.

[22] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1415–1425, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1148. URL https://www.aclweb.org/anthology/D14-1148.

[23] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2327–2333. AAAI Press, 2015. ISBN 9781577357384.

[24] George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf.

[25] Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3393–3402, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1331. URL https://www.aclweb.org/anthology/P19-1331.

[26] Xinya Du and Claire Cardie. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.49. URL https://www.aclweb.org/anthology/2020.emnlp-main.49.

[27] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468, Austin, Texas, November

2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1044. URL https://www.aclweb.org/anthology/D16-1044.

[28] C. Gulcehre, F. Dutil, A. Trischler, and Y. Bengio. Plan, attend, generate: Planning for sequence-to-sequence models. In *NIPS*, pages 5480–5489, 2017.

[29] Hongyu Guo, Colin Cherry, and Jiang Su. End-to-end multi-view networks for text classification. *arXiv preprint arXiv:1704.05907*, 2017.

[30] Luheng He, Mike Lewis, and Luke Zettlemoyer. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1076. URL https://www.aclweb.org/anthology/D15-1076.

[31] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1693–1701, Cambridge, MA, USA, 2015. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969239.2969428.

[32] Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2160–2170, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1201. URL https://www.aclweb.org/anthology/P18-1201.

[33] Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings*

of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pages 2127–2136. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/ilse18a.html.

[34] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, page 1148–1158, USA, 2011. Association for Computational Linguistics. ISBN 9781932432879.

[35] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, 2016.

[36] Vidur Joshi, Matthew Peters, and Mark Hopkins. Extending a parser to distant domains using a few dozen partially annotated examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1199, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1110. URL https://www.aclweb.org/anthology/P18-1110.

[37] Katharina Kann, Sascha Rothe, and Katja Filippova. Sentence-level fluency evaluation: References help, but can be spared! In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 313–323, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/K18-1031. URL https://www.aclweb.org/anthology/K18-1031.

[38] Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[39] Been Kim, Cynthia Rudin, and Julie Shah. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 1952–1960, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969045.

[40] Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 1–9, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W09-1401.

[41] Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. The Genia event extraction shared task, 2013 edition - overview. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 8–15, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-2002.

[42] Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*, 2016.

[43] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014. URL http://aclweb.org/anthology/D/D14/D14-1181.pdf.

[44] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[45] Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606. ACM, 2015.

[46] Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. Iterative edit-based unsupervised sentence simplification. *arXiv preprint arXiv:2006.09639*, 2020.

[47] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.

[48] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

[49] H. Larochelle and G. Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *NIPS*, pages 1243–1251, 2010.

[50] Kalev Leetaru and Philip A. Schrodt. Gdelt: Global data on events, location, and tone. *ISA Annual Convention*, 2013.

[51] Qi Li, Heng Ji, and Liang Huang. 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)*, volume 1, pages 73–82, 2013.

[52] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

[53] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv e-prints*, 1703.03130, March 2017. URL https://arxiv.org/abs/1703.03130.

[54] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[55] Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.128. URL https://www.aclweb.org/anthology/2020.emnlp-main.128.

[56] Xiao Liu, Zhunchen Luo, and Heyan Huang. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1156. URL https://www.aclweb.org/anthology/D18-1156.

[57] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, et al. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[58] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[59] Weiyi Lu and Thien Huu Nguyen. Similar but not the same: Word sense disambiguation improves event detection via neural representation matching. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4822–4828, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1517. URL https://www.aclweb.org/anthology/D18-1517.

[60] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[61] Jie Ma, Shuai Wang, Rishita Anubhai, Miguel Ballesteros, and Yaser Al-Onaizan. Resource-enhanced neural model for event argument extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3554–3559, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.318. URL https://www.aclweb.org/anthology/2020.findings-emnlp.318.

[62] Andrew L. Maas, Raymond E. Daly, et al. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

[63] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natu-*

ral Language Learning, pages 523–534, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D12-1048.

[64] David McClosky, Mihai Surdeanu, and Christopher D Manning. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1626–1635, 2011.

[65] Sneha Mehta, Bahareh Azarnoush, Boris Chen, Avneesh Saluja, Vinith Misra, Ballav Bihani, and Ritwik Kumar. Simplify-then-translate: Automatic preprocessing for black-box machine translation, 2020.

[66] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[67] V. Mnih et al. Recurrent models of visual attention. In *NIPS*, pages 2204–2212, 2014.

[68] Thien Huu Nguyen and Ralph Grishman. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 365–371, 2015.

[69] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 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, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1034. URL https://www.aclweb.org/anthology/N16-1034.

[70] Adam Pauls and Dan Klein. Large-scale syntactic language modeling with treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguis-*

tics (Volume 1: Long Papers), pages 959–968, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P12-1101.

[71] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[72] Matthew Peters, Mark Neumann, et al. Deep contextualized word representations. In *NAACL, Volume 1 (Long Papers)*, pages 2227–2237, June 2018.

[73] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

[74] Benjamin J. Radford. Automated dictionary generation for political eventcoding. *Political Science Research and Methods*, 9(1):157–171, 2021. doi: 10.1017/psrm.2019.1.

[75] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://www.aclweb.org/anthology/D16-1264.

[76] Naren Ramakrishnan, Patrick Butler, Sathappan Muthiah, Nathan Self, Rupinder Khandpur, Parang Saraf, Wei Wang, Jose Cadena, Anil Vullikanti, Gizem Korkmaz, et al. 'beating the news' with embers: forecasting civil unrest using open source indicators. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1799–1808. ACM, 2014.

[77] Giuseppe Rizzo and Raphaël Troncy. Nerd: A framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations*

at the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 73–76, 2012.

[78] Parang Saraf and Naren Ramakrishnan. Embers autogsr: Automated coding of civil unrest events. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 599–608, 2016.

[79] Philip A. Schrodt. Automated production of high-volume, near-real-time political event data. 2011.

[80] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[81] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *ArXiv*, abs/1611.01603, 2017.

[82] Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. RBPB: Regularization-based pattern balancing method for event extraction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1224–1234, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1116. URL https://www.aclweb.org/anthology/P16-1116.

[83] Dinghan Shen, Wang, et al. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th ACL (Volume 1: Long Papers)*, pages 440–450, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1041.

[84] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[85] Beth M. Sundheim. The Message Understanding Conferences. In *TIPSTER TEXT PROGRAM PHASE II: Proceedings of a Workshop held at Vienna, Virginia, May 6-8, 1996*, pages 35–37, Vienna, Virginia, USA, May 1996. Association for Computational Linguistics. doi: 10.3115/1119018.1119025. URL https://www.aclweb.org/anthology/X96-1006.

[86] Joshua Tenenbaum and William Freeman. Separating style and content. In M. C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1997. URL https://proceedings.neurips.cc/paper/1996/file/70222949cc0db89ab32c9969754d4758-Paper.pdf.

[87] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[88] David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1585. URL https://www.aclweb.org/anthology/D19-1585.

[89] Daisy Zhe Wang, Yang Chen, Sean Goldberg, Christan Grant, and Kun Li. Automatic knowledge base construction using probabilistic extraction, deductive reasoning, and

human feedback. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 106–110, 2012.

[90] Fulton Wang and Cynthia Rudin. Falling Rule Lists. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 1013–1022, San Diego, California, USA, 09–12 May 2015. PMLR. URL http://proceedings.mlr.press/v38/wang15a.html.

[91] Wei Wang. *Event Detection and Extraction from News Articles*. PhD thesis, Virginia Tech, 2018.

[92] Wei Wang, Yue Ning, Huzefa Rangwala, and Naren Ramakrishnan. A multiple instance learning framework for identifying key sentences and detecting events. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 509–518, 2016.

[93] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

[94] Boyi Xie, Rebecca J. Passonneau, Leon Wu, and Germán G. Creamer. Semantic frames to predict stock price movement. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–883, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-1086.

[95] K. Xu et al. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015.

[96] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[97] Mo Yu, Matthew R Gormley, and Mark Dredze. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1374–1379, 2015.

[98] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering.

[99] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[100] Yuhao Zhang, Peng Qi, and Christopher D. Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1244. URL https://www.aclweb.org/anthology/D18-1244.

[101] Yang Zhao, Zhiyuan Luo, and Akiko Aizawa. A language model based evaluator for sentence compression. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 170–175, 2018.