Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Harvesting the Web for building Large-scale Argumentation Graphs

Master's Thesis

Anh Phuong Le

1. Referee: Prof. Dr. Benno Stein
2. Referee: Prof. Dr. Andreas Jakoby

Submission date: September 30, 2020

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, September 30, 2020

.....

Anh Phuong Le

Abstract

Knowledge graphs have been used successfully in the development of many artificial intelligence systems. In computational argumentation, the emergent area in natural language processing, a new argumentation graph has been constructed by Al-Khatib et al. [2020]. The graph models argumentation knowledge as an 'effect' relation (edge) between concepts (nodes). The effect relation has two categories: positive and negative. Al-Khatib et al. [2020] proposes an approach to acquire the knowledge automatically applying machine learning models. This method takes a *claim* as input and outputs whether the claim has an 'effect' relation, the relation category, and concepts. However, the mentioned approach shows some shortcomings, such as dealing only with the claim inputs, that diminish the applicability of constructing a large argumentation knowledge graphs. To overcome these shortcomings and construct a large argumentation knowledge graph, we introduce a new approach to mine *effect relations* between concepts. The new approach, different than Al-Khatib et al. [2020], covers more complex input types and employs the recent advances in text classification. Our implementation includes two main steps: building a new dataset utilizing the paradigm of distance supervision, and developing neural-based classifiers applying an active learning technique to filter the dataset and improve the classification effectiveness. The results of the evaluation experiments show that our approach helps the classifiers to learn new patterns and be able to handle sentences with complex structure.

Contents

Acknowledgements

I would like to take this chance to thank my advisors, Dr. Khalid Al-Khatib and Dr. Michael Voelske, who have provided me with their experience and insights into the problems that I face through the thesis. Without their constant support, I would not achieve this stage of my work. Moreover, I am grateful to be taken as a student in Bauhaus University Weimar in general, and doing my thesis under Webis group, with Prof. Benno Stein as my first supervisor. At the same time, I would like to express my gratitude towards Prof. Andreas Jakoby, who have encouraged me a lot during my Master study and agree to be my second supervisor.

I would never have come to Germany for studying without unlimited love and financial support from my parents, who work hard to offer me a nice living condition as today. Everyone in my family has an impact to me, which creates the person I am today. For that reason, I dedicate my work to my family members, my beloved grandfather, my father, my mother, my lovely brothers and my grand mother.

Last but not least, my journey towards finishing my Master is never complete without all the time well spent with friends. This is my opportunity to appreciate them, especially Yen Dang, who has gone through all the ups and downs during my development of wisdom.

Chapter 1

Introduction

Knowledge graphs play a vital role in tackling various tasks in artificial intelligence such as question answering, recommendation systems, information discovery, text generation and search engines. Several knowledge graphs which are used to incorporate facts or common sense knowledge such as *Google* knowledge base, *DBpedia*, and *ConceptNet*, have been used widely in research and industry. This is stemmed from the fact that knowledge graphs can represent a diverse set of knowledge, capturing a wide range of entities, including real-world objects, events, situations and abstract concepts, along with the relations between them, while offering a reliable, explainable and reusable way to obtain and describe the knowledge.

In this thesis, we leverage the power of knowledge graphs to address a significant aspect of natural language communication: **argumentation**. Argumentation is an active part of our lives; it is involved in every logical reasoning process we do, from individuals' decision-making to groups' debates on certain topics. The construction of argumentation knowledge graphs is a promising approach; it will pave the way for later stages of the computational argumentation life cycle, i.e. improving arguments quality assessment, argument retrieval and synthesis.

Nevertheless, the automation of knowledge acquisition in knowledge graph construction is not a simple task. It requires a fitting model and an end-to-

end competent automated system, from discovering knowledge, recognizing the participating entities to assigning the appropriate semantic relations among the entities.

An early step towards building an **argumentation knowledge graph** has been proposed by Al-Khatib et al. [2020], in which a new graph model is proposed, with '*concepts*' as nodes and positive or negative '*effect*' relations as directed edges between them. Based on this model, the authors successfully mined 1,736 claims containing the defined relations from *online debate portals*. However, there are several shortcomings in this work that limits the applicability of building large-scale argumentation graphs. In particular, (1) the knowledge is mined from one component of argumentative text: claims, which are often expressed in the form of simple and short sentences, while most of the effect relations occur inside the logical reasoning process, i.e. many effect relations appear in the premises of arguments (the propositions that lead to the final conclusion). (2) Looking at the distribution of relation types in the used dataset, we observe a significant imbalance between effect relation types (74% positive - 23% negative relation), which restricts the effectiveness of the statistical models developed using this dataset. (3) The classifiers created for automatic knowledge acquisition are rather preliminary; they are trained using a set of selected linguistic features including lexical, syntax, sentiment and semantic ones. With the current leap in NLP Deep Learning models, it is reasonable to make use of them and strive for better effectiveness of these classifiers.

To overcome the outlined shortcomings, in this thesis, we implement several techniques to improve the mining of the effect of relations from web text. Our contribution is *two-folds*. (1) We construct *a new dataset of annotated* effect relations from argumentative web text, which supplement the current dataset of claims in terms of complexity, type balancing and generalization capability. (2) We develop *several classifiers* using the latest state-of-the-art models for automatic knowledge extraction, including the *detection* of *effect* relations and the *classification* of the *relation types*; these classifiers can work with sentences that have *complex structures* and contain *multiple* effect relations.

To build the dataset, we apply the techniques of **distance supervision** (Zeng et al. [2015]) and **active learning** (Settles [2009]). In the distance supervision step, we use the *Hadoop* framework to transform text into 'simple' sentences, and then filter concepts in a knowledge graph and extract those sentences which include these concepts as candidates with potential effect relations. Subsequently, using the active learning technique, we train multiple classifiers based on several state-of-the-arts *deep learning* models and use them to select new instances (sentences) for relation's manual annotation. Lastly, we train new classifiers using the newly developed dataset and analyze their performance. Figure 1.1 shows the main steps of our procedure.

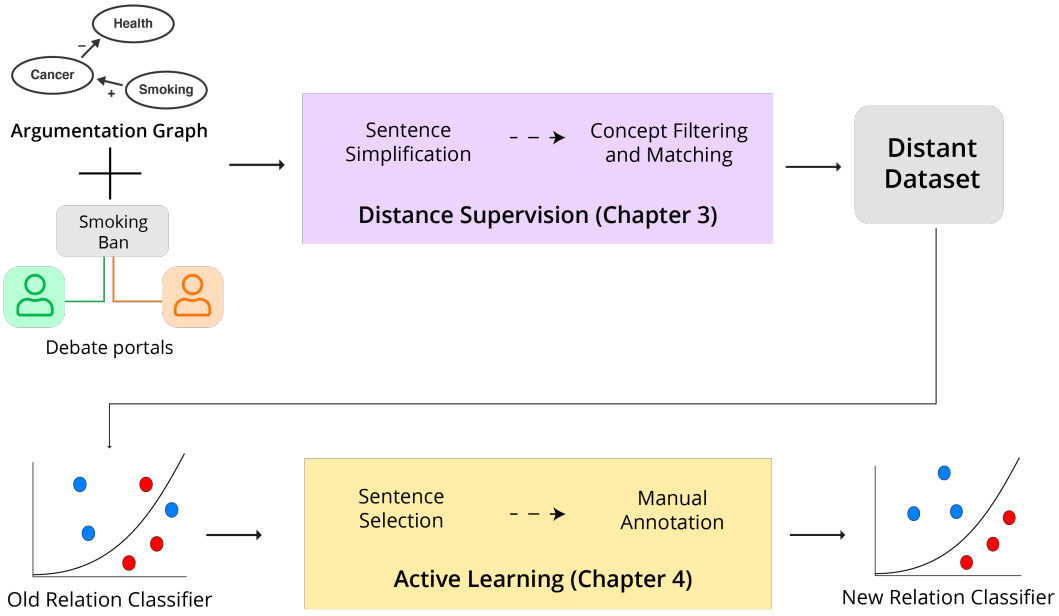

Figure 1.1: Proposed procedure of extracting effect relations

The results show that our classifiers have very high effectiveness, achieving F1 score of up to *89%* for effect detection and *92%* for relation type detection. We also found that masking the concepts in the training sets of the learning models would help the classifiers give more concentration on the relation patterns. Overall, the newly created dataset has improved the generalization of our classifiers.

The structure of this thesis is as follows. **Chapter 2** gives more details

about the basic foundation of the studied task and methods. Beside revisiting literature of *argumentation theory*, *knowledge graph*, *distance supervision* and *active learning*, we introduce the research by Al-Khatib et al. [2020] on *Argumentation Knowledge Graph*. **Chapter 3** describes our process of extracting candidate sentences with effect relation from argumentative debate portals texts. We, first, simplify the texts converting them into simple sentences. Then, we filter pairs of concepts from the argumentation knowledge graph. These pairs are aligned with the simplified sentences in order to find matching instances (candidate sentences). **Chapter 4** demonstrates how we train several classifiers and develop masking and filtering strategies of the candidate sentences. The filtered sentences are then labelled using crowd-sourcing and the results are aggregated to train a new classifier. **Chapter 5** explains our evaluation experiments on the *old* (Al-Khatib et al. [2020]) and the *new* annotated dataset. In addition to training and testing within and across the datasets, we also inspect the distribution of words' importance toward classifiers' prediction. Finally, **Chapter 6** concludes our achievement and discusses future improvement.

Chapter 2

Background and Related Work

This chapter describes the relevant work related to this thesis. First, section 2.1 briefly discusses the role of arguments in our daily life and current academic work on computational argumentation while section 2.2 describes the basics and categorization of research on knowledge graph. Subsequently, section 2.3 summarizes the current works relating to arguments analysis using knowledge graph and gives details on the argumentation knowledge graph model defined by Al-Khatib et al. [2020]. To prevent confusion and mixed assumptions regarding certain concepts, we also introduce the necessary explanation of certain terminologies. Besides revisiting the importance of argumentation and knowledge graph, we dedicate the last three sections of this chapter for discussing the relevant works regarding our methodologies. To add more details, section 2.4 reviews Distant Supervision – a well-known method for extracting relations from text, section 2.5 describes Active Learning – a popular machine learning approach where classifiers actively select new instances for training and section 2.6 presents Transformer-based neural networks – current state-of-the-art deep learning models in the field of Natural Language Processing.

2.1 Argumentation

In daily lives, humans construct *arguments* during debates to support individuals' or groups' decision making. It requires 2 competitive sides providing arguments for and against a proposition, and eventually the stronger side wins (Bjork [1994], Walton et al. [2008]). This process is formalized in researches using *argumentation schemes*. Basic arguments (defined by Walton et al. [2008]) comprise a collection of statements (premises) to support a claim (conclusion). To make a strong argument, one often employs causation (cause-to-effect and effect-to-cause – Hahn et al. [2017]). Figure 2.1 describes an example of an argumentative text which contains several cause-effect arguments. In the field **Computational Argumentation**, we deal with analysis (Argumentation Mining) and generation (Argumentation Synthesis) of arguments, which decomposes into mining, assessment, retrieval, inference, generation and visualization. Some of the works surrounding this topic to be listed are identifying location of argument and classifying types (premise vs conclusion), and relations (support vs attack) of argument units (Stab and Gurevych [2017], Persing and Ng [2016]); classifying stance of argument (Toledo-Ronen et al. [2016]); assessing quality of argument (Wachsmuth et al. [2017a]); building an argument search engine (Wachsmuth et al. [2017b]); synthesizing arguments (Wachsmuth et al. [2018], El Baff et al. [2019]).

2.2 Knowledge Graph

Knowledge Graph, in short, is a method of incorporating human knowledge, by constructing one or many graphs. This kind of graphs contains *nodes* – indicating entities, and *edges* – representing relations between entities (Hogan et al. [2020], Ji et al. [2020]). The term knowledge graph is used interchangeably with knowledge base. By **Knowledge Base**, one often refers to *triple form* representation of knowledge, i.e.(*head*, `relation`, *tail*). For example, the sentence '*Hanoi is the capital of Vietnam*' could be expressed as (*Hanoi*, `CapitalOf`, *Vietnam*). Knowledge Acquisition is an essential step

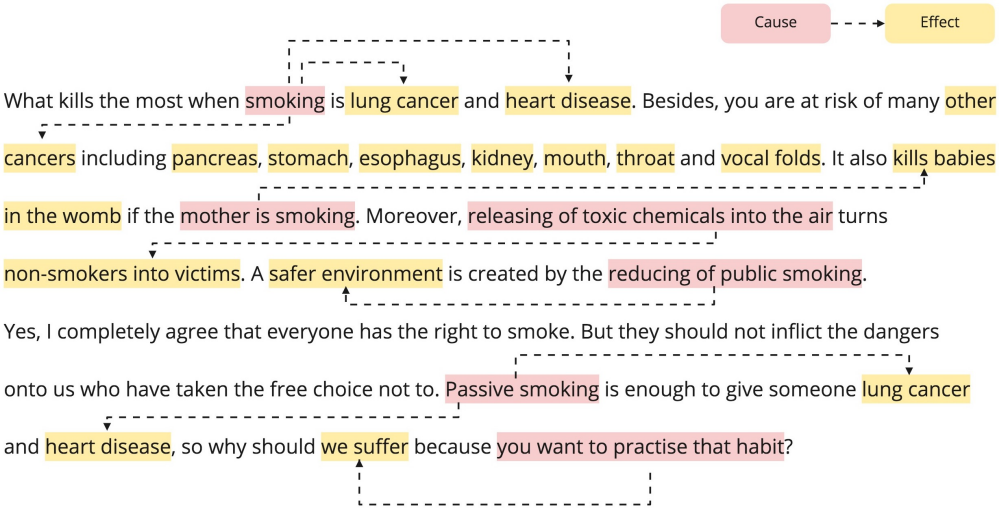

Figure 2.1: Example of an argumentative text taken from the debate *'should smoke be banned in public' – debatewise*

to construct a knowledge graph from unstructured text. It involves **Knowledge Graph Completion** – expanding current graph and discovering new knowledge; **Relation Extraction** – extracting unknown relational facts from text and **Entity Discovery** – entity-related tasks such as recognizing, typing, disambiguating and aligning entities (Ji et al. [2020]). Graph representation provides the advantages of structurally capturing and evolving incomplete knowledge, without the need for defining a schema in advance (Hogan et al. [2020]). Though this idea has been originated since the mid-20, it only becomes a popular term since Google announced its own framework to build a large-scale knowledge graph in 2012 (Hogan et al. [2020], Ji et al. [2020]). Since then, knowledge graph has become an essential focus of industrial and research, with big companies' involvement and various publication (Hogan et al. [2020]). Application of knowledge graphs includes semantic search, deep reasoning, machine reading, entity consolidation, text analytic and etc.(Bonatti et al. [2019]).

2.3 Argumentation Knowledge Graph

Different researches have been carried out to analyze arguments using knowledge graph. For example, Becker et al. [2017] and Kobbe et al. [2019] consider each argumentative unit as a node, and label relations between them as *support* or *attack*. This approach, however, only overlooks the dependencies between text units, while ignoring the specific content inside the arguments. On the other hand, Toledo-Ronen et al. [2016] focuses on *stance classification*, building a knowledge graph of experts' stance towards various Wikipedia concepts. While stance is an important aspect of arguments, a stance alone is insufficient to comprehend. Since arguments are different from facts or common sense, there is no specific answer to a question. Instead, humans make logical statements in order to persuade others (Wachsmuth [2019]). This reasoning is often represented in text as *causality relations* (Guo et al. [2018]). In order to capture the rational embraced by arguments, Al-Khatib et al. [2020] develop a **Argumentation Knowledge Graph model** that represents each node as a *concept instances* and each edge as a *positive* or *negative effect relation* between two concepts. Figure 2.2 shows examples of extracting relations from sentences using a simplified version of the defined model. The *effect relation* specified in this work, when compared to *Causes* – the most similar relation type in ConceptNet (Speer et al. [2017]), is distinctive. First, its knowledge origin is argumentative text, versus the common sense knowledge captured by ConceptNet. Second, the *Causes* relation in ConceptNet only represents relation between two specific events, e.g. *Exercises* vs *Sweat* while the definition of a *concept* in Al-Khatib et al. [2020]'s *Argumentation Knowledge* is rather general. **Concept instances** could be a phrase, an event, or abstract principle or idea. Third, the **Effect relations** are expressed as *directed edge* (**positive** vs **negative**), concisely describing the semantic meaning (increase vs decrease). This argumentation knowledge graph model also proves to be effective in argument generation (Trautner [2020]). In the work done by Al-Khatib et al. [2020], claims are extracted from the corpus by combining several indicators and selected such that they are simple sentences and self-contained units. This is an effective way of obtaining the relations, however, it avoids the complexity

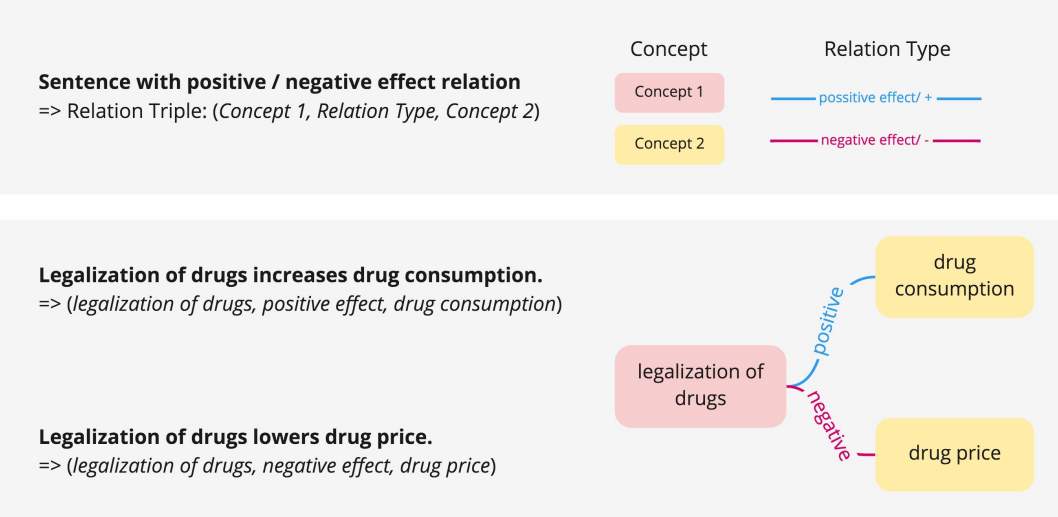

Figure 2.2: A simplified version of argumentation knowledge graph model defined by (Al-Khatib et al. [2020]) – examples of how relation triples are extracted and incorporated into knowledge graph

of argumentation and instead only focuses on a small sampled data set. We further explore this model by looking at a large corpus and automatizing the process of relation extraction.

2.4 Distant Supervision for Relation Extraction

There are several ways to extract relations from text. First, a purely supervised approach develops classifier for relation detection, which is biased towards the handed label data set. Second, a purely unsupervised approach clusters the words string between entities for relations, but the mapping for a large corpus is rather difficult. The third approach bases on bootstrapping from a small amount of seed instances to find new patterns, which also yield new instances from big data set. A notable method of this kind that gains much popularity is called *Distant Supervision* (Mintz et al. [2009]). The typical workflow is:

1. Obtain the entity pairs from a knowledge base.
2. Extract and align entities from text to find relation mentions. **Entity**

mention is a token span of text which represents an entity e while **Relation mention** for some relation instance $r(e_1, e_2)$ is a pair of entities mentions e_1 and e_2 in a sentence s (Ren et al. [2017]).

3. Assign the extracted sentences with candidate relation type(s).
4. Derive unseen relation patterns from newly discovered instances.
5. Assign relation type(s) for remaining sentences in the corpus.

This method makes the assumption that if two entities that have expressed a relation in the seed instances will have the same relation if they both appear in a sentence. Riedel et al. [2010] argue that this assumption does not always apply, since entities related to the same context tend to emerge in the same sentence. Hence, they relax this assumption to *at-least-one* sentence that conveys the relation, which results in multi-instance single-label. Hoffmann et al. [2011] and Surdeanu et al. [2012] further develop this by considering the *overlapping relations*, i.e. different relations which share same pair of entities, hence, multi-instance set-of-labels. With the rising popularity of neural networks, several researchers attempt to apply it into this task in different ways. For example, to overcome noisy label problem, Zeng et al. [2015] and Lin et al. [2016] train a *Convolution Neural Networks* to weight the relevance of sentences in multi-instance learning (Smirnova and Cudré-Mauroux [2018]). On the other hand, Han et al. [2018] implement *Attention Network* to extract head entity, relation and tail entity in a hierarchical manner. From our initial speculation, distant supervision promises to be a suitable approach, given that we want to acquire the new relations' instances and patterns automatically. As we are approaching a newly defined relation model, these previous approaches are only kept as reference, as different problems arise during the implementation.

2.5 Active Learning

Active Learning is a popular method to achieve better accuracy with fewer training instances. The idea behind it is that a machine learning algorithm can actively query unlabelled instances to be labelled by an oracle, e.g human

annotator; as a result, the algorithm can maximize its learning capacity. This approach is particularly helpful for information extraction, where obtaining labels is a time-consuming and costly process. Angluin [1988] shows that by carefully selecting training instances, the number of data required for training could be reduced. Later, Cohn et al. [1994] introduces 'active learning' term to describe the process of algorithms selecting from a set of potential training instances. In this selection process, there requires some *informativeness score* as criteria for selection. Typically, there are 3 scenarios (Settles [2009]) where the learner will query the labels of instances. First, in *membership query synthesis*, the learner generates an instance from some natural distribution and send to the oracle for labelling. Second, *stream-based selective sampling* is a setting where only one unlabelled instance is queried at a time and the learner could reject it based on its informativeness. Third, *pool-based sampling* means that there is a large pool of unlabelled data and most informative instances are selected.

To evaluate the informativeness of unlabelled instances, several query strategies are proposed. Primarily, in *uncertainty sampling* (Lewis and Gale [1994]), the learner queries instances which they have the least confident predictions. A variety of approaches are corresponding to uncertainty sampling, e.g. least confidence sampling, margin and ratio of confidence sampling and entropy-based sampling. Another popular query selection framework is *query-by-committee* (QBC) (Seung et al. [1992]) algorithm. This method proposes training a set of classifiers on the labelled dataset and allowing them to vote on the label of querying instance. Subsequently, the instances with most disagreement have the most informativeness. Some variation of constructing committees is through ensemble learning methods such as bagging and boosting. In our work, as pool-based sampling is applied to select many sentences for annotation, we use a combination of least confidence sampling and query-by-committee, with each committee is a classifier trained on a different pre-trained model.

2.6 Neural-based Transformer

Transfer Learning has gained a huge attention during recent years in the field of Natural Language Processing (NLP) (Malte and Ratadiya [2019]). Normally, in a pre-training phase, a *Language Model* (LM) is learned in different ways to understand the probability distribution over sequences of tokens regarding the language (Jozefowicz et al. [2016]). Subsequently, the general LMs are applied to fine-tune on specific tasks, such as classification and question-answering with relatively lower requirement in terms of data size and training time. One significant example of such a pre-trained model is **BERT** (Devlin et al. [2018]), which combines several state-of-the-art techniques including Masked Language Modeling and Multi-headed self-attention from the so-called *Transformer model* (Vaswani et al. [2017]) for two unsupervised tasks of predicting the masked tokens and classifying pairs of adjacent sentences. When the authors introduced BERT, it had broken records in 11 NLP tasks over *GLUE* and *SQuAD* benchmarks. Following the renowned success of BERT, *HuggingFace Transformer* (Wolf et al. [2019]) offers an unified API for easy application with a typical NLP pipeline and access to several state-of-the-art architectures with tutorials and scripts for users. Besides BERT, several notables architectures available in HuggingFace Transformers are **RoBERTa** (Liu et al. [2019]) – a replication of BERT with tuning hyper-parameters and larger training set, **DistilBERT** (Sanh et al. [2019]) – a small BERT version trained by BERT itself using knowledge distillation (Hinton et al. [2015]), **ALBERT** (Lan et al. [2019]) – a lite BERT that applies parameter sharing among continuous layers and **XLNET** (Yang et al. [2019]) – a model built upon Transformer-XL (Dai et al. [2019]) and Permutation Language Modelling.

Chapter 3

Dataset Construction using Distant Supervision

In this chapter, we describe our process of applying the Distant Supervision method to extract sentences which have high potential of containing some effect relation. First, in section 3.1, we provide the details on the datasets used in our work. Then, section 3.2 explains how we simplify text from big dataset to sentences in local and distributed system. Subsequently, in section 3.3, we discuss how the concept pairs from annotated relation instances are processed and how matching of these concepts is implemented.

3.1 Corpus

In the following, we provide a comprehensive description of the corpus used for the experiments.

3.1.1 Manually annotated dataset

We inherit the manual annotated dataset from Al-Khatib et al. [2020]. It contains 4,722 claims extracted from *Debatepedia* – an online debate portal. Each claim contains a simple sentence structure (subject, verb, object) and is

a self-contained unit. The annotation is performed by crowd-sourcing method. Each annotator reads the sentence carefully and answers whether it contains a *positive* or *negative* effect relation, and point out the two concepts participating in the relations. In addition, relevant entities which are obtained by *Tagme* and *Babelfy* relating to each concept are also identified. For example, *legalizing drug* is combined from two entities: *legalization* and *drug*. The final annotation is aggregated from 5 annotators' results. We utilize this annotated dataset for two purposes. First, we train statistical models to label the sentences for detecting effect relation and classifying relation types. Second, we extract pairs of concepts that participating in the relations and align those pairs with unlabelled sentences to find new instances with relations.

3.1.2 Args.me dataset

The Args.me dataset is developed for the purpose of argument search engines by Ajjour et al. [2019]. It contains numerous debates regarding controversial topics crawled from 5 popular debate portals, including Debatewise (14,353 arguments), IDebate.org (13,522 arguments), Debatepedia (21,197 arguments), and Debate.org (338,620 arguments). The arguments are retrieved by heuristics created specifically for each debate portal. In total, it has the size of 8 GB and 387,606 arguments. After processing, we obtained **27,133,477** sentences.

3.2 Data Simplification

To form strong arguments, ones often write text with a logical structure and closely related sentences. In addition, the sentences are most of the time compound, complex and run-on sentences. These types of sentences could comprise of several connecting clauses and different expression could refer to the same entity. For that reason, a simple splitting of paragraph into sentences would risk losing important context and make it difficult to understand for human and machine. As a result, we implement simplification steps which

not only transform long text to sentences but also simplify and resolute co-reference. After this process, we obtain self-contained sentences.

3.2.1 Text Simplification using Graphene

Since the text from debate portals have high complexity, it is necessary to implement sentence simplification before it is possible to extract any relations. In this work, we use Graphene (Cetto et al. [2018]), an Open Information Extraction tool. This is specifically built for the purpose of relation extraction. We employ the first phase in their pipeline – *Discourse Simplification*. In this step, it recursively removes the clauses and phrases that represent the context into unique *context* sentences, leaving only core information in the *element* sentence. Hence, the input is transformed into multiple sentences of canonical form, each contain a separated fact. Besides, co-reference resolution is incorporated in the process – the expression referring to the same entity is kept when sentences are disconnected. This makes the sentences still understandable when standing alone, without context. In the following, Figure 3.1 shows one example output of the process.

From our assessment, this simplification results are helpful for our downstream task – *Effect Relation Extraction*. To simplify the Args.me corpus, we

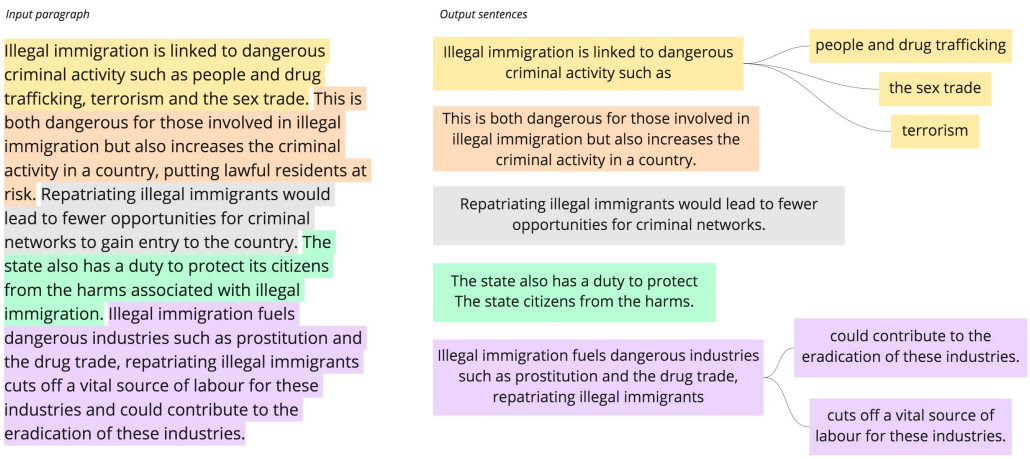

Figure 3.1: Example of simplified sentences from a paragraph

attempt to run the process in parallel in a local machine of 24 cores. However, this operation takes great time to run, as a result, we rely on our *Hadoop* cluster.

3.2.2 Distributed Simplification Process

We use the cluster system that is set up in our labs, which includes 2,000 CPUs. To run simplification in cluster, the program is adapted based on *MapReduce* interface. Since *Hadoop* stores lines of a text file in a distributed manner, we prepare the input file such that each line is a paragraph. For JSON file format, we use python *jsonslicer* library to parse the big JSON file and extract the paragraphs. The program is run successfully when all the necessary libraries with the main code compiled into a big jar file. Some implementation details are followed. First, *Graphene* is initialized in the set up of MapReduce. Then, each implemented Mapper simplifies a paragraph separately. To prevent app crash or time out due to some abnormality, we place a separate thread to check the execution time of each simplification task and skip if it takes too long. Additionally, we create a class to filter out sentences with some abnormality, e.g. too long (more than 90 tokens), too short (less than 5 tokens), weird structures (high frequency of commas and brackets). Last but not least, to speed up running the task, we use 20,000 Mappers and 100 Reducers. The processing of args.me dataset, in the end, only took a few days.

3.3 Concept Matching

Concept Matching is the main step of Distant Supervision. In this process, we first filter our current relation instances, hence extracting the most prominent relations between pairs of concepts. After that, we align this concept pairs with our simplified datasets to find sentences potentially contain effect relation. Finally, we look at the results and do some quality assessment to further filter out noisy instances.

3.3.1 Relations Filtering

As described in 3.1.1, the manual annotated dataset is used as *seed* instances for the Distant Supervision. After some careful inspection, we decide to implement further processing of the dataset to improve the outcome of concept matching. There are two main reasons accordingly. First, it is often the case, that the concept is expressed in a long sequence of text. For example, the effect relation from the sentence '*By legalizing drugs, the state can regulate the sale*' annotated is (*legalizing drug*, `positive relation`, *the state can regulate the sale*). With this pair of concepts, it is not feasible to match fully the second concept, but rather only the most important keywords from the concept, e.g. *sale regulation*. Second, following the setup of the annotation process, sometimes multiple concepts are mixed together in one concept. For instance, sentence '*Legalizing medical marijuana causes crime and safety problems*' express two relations (*legalizing medical marijuana*, `positive relation`, *crime*) and (*legalizing medical marijuana*, `positive relation`, *safety problems*); however, the annotation is (*legalizing medical marijuana*, `positive relation`, *crime and safety problems*). These kinds of instances in the annotated dataset limit the process of concept matching.

To overcome this shortcoming, the author carefully reviews the list of instances; meanwhile extracting the keywords from long concepts and separate relation with multi-concept to multiple relations. In addition, concepts are grouped to the same *base*. For example, *headscarf* is mentioned in the dataset several times, with various representation, e.g. *burqa, veil wearing, full veil, face covering, hijab*. After this process, we obtain 1,764 relations and 1,930 base concepts.

In the meantime, we take note of several examples, which could help to further improve our *Effect Relation Model*. First, after the extraction, how could we formulate the concept to the end user? A sequence of tokens could be shortened using keywords and various forms of words, such as, *Noun + Noun, Adjective + Noun or Noun + Verb-ing*. The second interesting aspect is the prospect of *inferring* a relation. For example, if two concepts

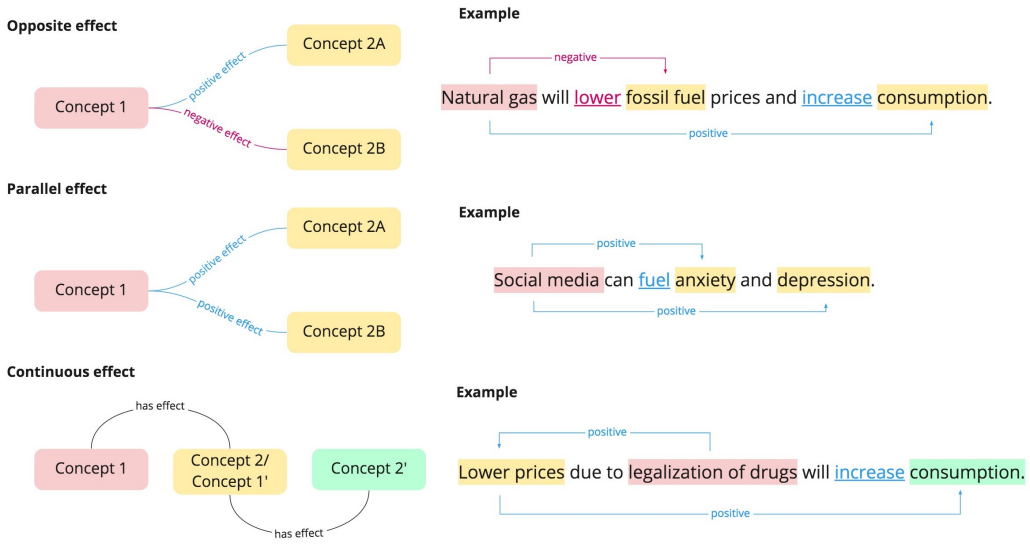

Figure 3.2: Complex effect relations and examples

are related, e.g. *gay marriage* vs *legalizing gay marriage* or two concepts are somehow expressing opposite meaning, e.g. *banning gay marriage* vs *legalizing gay marriage*, taking into account the connection between concepts could facilitate the correct side classification of the existing relation in the sentence and the deduction of new effect relation. Let's take the sentence '*Legalizing marijuana would make roads more dangerous*' as an example. We extract relation (*legalizing marijuana*, `positive relation`, *dangerous road*) from this sentence. Therefrom we could infer 2 more relations, such as, (*legalizing marijuana*, `negative relation`, *safety*) or (*banning marijuana*, `positive relation`, *safety*). Last but not least, the complex effect relation is also considered. Hereby, in Figure 3.2, we list several cases where complex effect relation could occur in a sentence. After the revision step, we sort all the concept by frequency, the most frequent concept appears in 40 relations.

3.3.2 Matching of Concept Pairs

After Relation Filtering is the process of matching the concept pairs from our annotated dataset to the big corpus. We stem all the tokens using Porter

Table 3.1: Concept matching statistics

Number of matching sentences						
	debateorg	debatepedia	debatewise	idebate	parliament	sum
full	24,064	2,650	466	831	2	**27,793**
two-third	47,257	1,660	312	465	0	49,694
half	133,1995	40,171	23,654	32,743	257	1,428,820

Stemmer and compare tokens from each sentence with tokens from all concepts 1. If there is a match, we take the base of the found concept 1, and find all concepts 2 that has some effect relations with any in the group of similar concepts. Each match is some overlapping tokens between sentence and concept. We match base on 3 levels: *full-*, *two-third-* and *half-* string matches. The result of this process is described in Table 3.1. Due to the limitation of man power, we only take attention to approximately 28,000 matches from full match. Inspecting some examples of this dataset, we realize the need to filter out noisy sentences.

3.3.3 Results' Noise Reduction

Before filtering, we group all matches by matched concept pairs and order them by frequency. Additionally, we remove duplicates, all links and special characters in the sentences. Some glance through the instances give us ideas about whether to filter or to keep the concept pairs. Some pairs which often produce noise in the data are those with overlapping tokens (e.g. *immigrant*

Table 3.2: Concept matching after noise reduction statistics

Number of *full matched* sentences after noise reduction					
debateorg	debatepedia	debatewise	idebate	parliament	sum
9,302	613	241	173	0	10,329

law and *immigrant*); irrelevant pairs (e.g. *individual* and *corporation*) and infrequent pairs (too specific concepts). Thereafter, we group by sentences and sort them by frequency of concept pairs for ease of observation. To evaluate this filtering process, the author checks in a random sample of 100 instances from the dataset *before* and *after* filtering. The results shows a leap in percentage of sentences with some effect relation from 56% to 70%. Table 3.2 shows some results statistics.

Chapter 4

Relation Classification with Active Learning

This chapter explains our process of applying active learning for training classifiers and querying instances for annotation. In section 4.1, we compare different types of classifiers training on several models. From this comparison, we develop a filtering strategy to select sentences for public annotation. The crowd-sourcing annotation process is described in section 4.2.

4.1 Training classifiers

In essence, we train several classifiers using ktrain library and apply it to the *args.me matched concepts sentences* dataset from previous step, evaluate the prediction, compare them to choose some sentences for annotation. These sentences should either contain unseen patterns or complex effect relations, which could later enhance the performance of the classifiers.

4.1.1 Training setup

The *ktrain* library provided by Maiya [2020] is considerably helpful in training classifiers. It offers ease of access to a variety of pre-trained models by

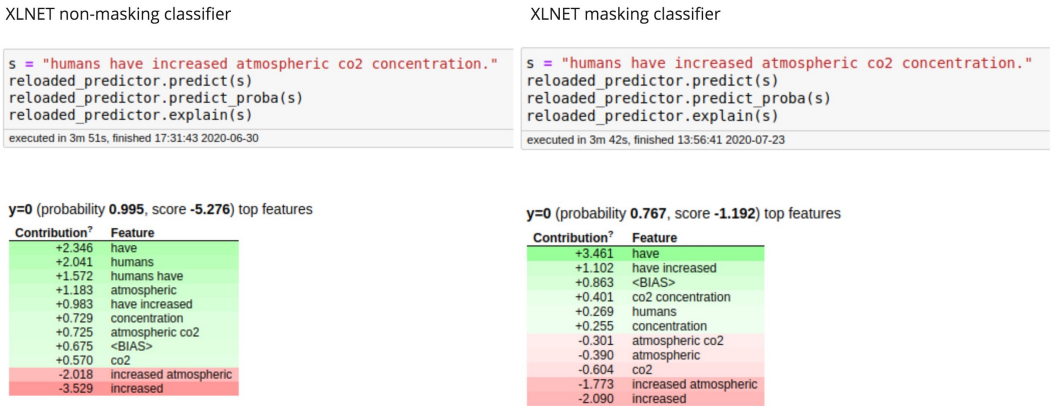

Figure 4.1: Example when masking classifier make better prediction than non-masking classifier

Hugging Face (Wolf et al. [2019]) and several machine learning baselines. In the following we describe our general steps taken to train a classifier using ktrain. First, the dataset is separated into 80% training and 20% testing. At the next step, ktrain assists in pre-processing and loading the selected pre-trained model. Subsequently, we simulate the training for some epochs and inspect the loss to find the most suitable learning rate for the dataset. For comparison, $4e^{-5}$ is chosen as the common learning rate for the experiments. Moreover, we use function *fit-one-cycle* to train. The major step is to train using the training set and evaluate the classifier with the test set. After training, it is also essential to find the model's top loss and analyze whether and how it fails for such cases. The contribution of n-grams towards the prediction is explained using **lime** (Ribeiro et al. [2016]).

4.1.2 Comparison of the Classifiers

Using the described method, we focus on training 2 types of classifiers on the manual annotated dataset:

- **Effect Detection** Classifiers: detect if there exists at least one effect relation in the sentence.
- **Relation Type** Classifiers: classify if the effect relation detected in

the sentence belongs to negative or positive type. This classifier could not apply to sentences with mixed (both positive and negative) effect relations.

The similar methodologies and experiments are carried out for training both types of classifiers. We first train an *effect detection* classifier using **DistilBERT**, which shows a promising result – F_1 score = 0.88 (compared to 0.81 in Al-Khatib et al. [2020] paper). After some inspection of n-grams contribution to each prediction, we suspect that the model not only learns the relation pattern, but also the mentions of entities. As a result, we implement masking to cover the concept entities with a constant string and retrain the classifier.

Masking of sentences The masking could not be done perfectly. Though most of the time, relation patterns appear as *verbs* in sentence, sometimes they are *noun phrase* as well. Therefore, we should avoid removing those from our sentences. First, we use *DBpedia Spotlight* (Mendes et al. [2011]), which disambiguate, extract and map entities from sentences to DBpedia URIs. The threshold is set to 0.3. Second, we employ *Spacy* (Honnibal and Montani [2017]) to find NER and noun phrases in the sentence. We combine all found text sequences and mask two-third of them. In Table 4.1, we show comparison of different pre-trained models when we run it with **masking** and **non-masking**. From our observation, the classifiers which are trained on masked sentences do not perform as well as those without masking; however, it makes more emphasis on *relation patterns* rather than falsely assume the *concept mentions* as representation of 'effect' relation. One example is shown in the Figure 4.1. The classifiers trained on masked sentences support the selection of sentences for annotation to improve our current classifiers.

4.1.3 Uncertainty Sampling

At this stage, we have trained some reliable classifiers on the *claims*; subsequently, we apply them on the *concept matching args.me dataset* to understand how the classifiers fit with the new dataset. As this ones are sentences extracted

Table 4.1: Comparison of classifiers

	F_1 score						
Effect Detection	DistilBERT	ALBERT	BERT	RoBERTa	XLNET	NBSVM	Fasttext
non-Masking	0.88	0.88	0.88	**0.89**	**0.89**	0.81	0.79
Masking	0.84	0.62	0.85	**0.86**	**0.86**	0.79	0.79
Relation Type	DistilBERT	ALBERT	BERT	RoBERTa	XLNET	NBSVM	Fasttext
non-Masking	0.90	0.79	0.90	**0.93**	**0.91**	0.88	0.86
Masking	0.89	0.79	0.79	0.79	**0.86**	**0.88**	0.87

from *full text of arguments*, as opposed to only *claims* in the *manual annotated datasets*, we expect the classifiers will fail where it comes to more complex and unseen pattern. For that reason, *active learning* method is utilized, which basically rules out those sentences with prominent labels, confidently predicted by the classifiers. Furthermore, those sentences that produce uncertain predictions are kept for crowd-sourcing annotations, which in the succeeding steps will help generalize and improve the classifiers' accuracy. Overall, after this step, we keep about **2,000** sentences from 10,000 matched sentences.

Classifier Confidence Calibration To obtain a general overview of the classifiers accuracy, we generate the confidences of **XLNET** and **ROBERTa**– the best *effect detection* classifier for *non-masking* and *masking* prediction and put them into bins from 1 to 9, with 1 as *really sure* that the sentence has *no effect relation*. From our observation, only if the classifiers are absolutely confident about the *non-effect*, otherwise, there is chance that the predicted sentence still displays some effect relation. Moreover, if the classifiers detect some effect relation (confidence score from 5 to 9), the sentence should definitely express that as well. This observation helps us to come up with a good strategy for *uncertainty sampling*.

Filtering Strategies With the objectives of acquiring sentences containing *unseen* or *complex relation patterns* and more *negative effect relation* examples

to eventually improve our classifiers' effectiveness and balance of the annotated dataset, we implement the following steps to filter out the sentences that have high confidence of prediction (confidence bin of 1 or 9) and obtain consistent prediction from our best classifiers (comparison in Table 4.1) that are trained on the full 'manual annotated dataset'. Following are the types of instances that are filtered from our 'matching' sentences.

- Those with high confidence of *no effect* (agreed by *masking* and *non-masking effect detection* classifiers): **6103** sentences
- Those with high confidence of *effect* (agreed by *masking* and *non-masking effect detection* classifiers): **1615** sentences
- Those with some *positive effect relation* for sure (agreed by *masking* and *non-masking effect detection* classifiers and best *relation type* classifiers): **1828** sentences

With this filtering methods, we acquire in total **1,937** sentences left for crowd-sourcing annotation. A quick inspection into 100 sentences of these shows a high percentage of instances with unseen and complex effect relation, along with a more frequent occurrence of negative effect relations.

4.2 Crowd-sourcing Annotation

In order to acquire more knowledge instances for automatic knowledge extraction and to train new classifiers to better detect and classify effect relation in more complicated sentences, we conduct crowd-sourcing annotation in *Amazon Mechanical Turk (Mturk)*, a service provided by Amazon which serves our purpose. Mturk allows requester to spread tasks of their own requirements to its remote workers, and assess the annotation results of workers for approval. It also helps us to distribute pays to workers with approved tasks. Since the last annotated dataset is also acquired from Mturk, the interface and instruction is utilized and modified to suit our new requirement.

4.2.1 Annotation Task

A task developed in Mturk for acquiring the annotations from multiple human sources in a short time period. It comprises of a web interface and instruction to guide users through the requirements of the task.

Annotation Interface We inherit the annotation interface from the last annotation task and make some necessary modifications to meet our goal. Since we are dealing with a more complex task compared to the original paper, *multiple-relation* is included. After workers *detect* some *relations*, there appears a box for annotating the first effect relation discovered and also a button to add more relation. It is compulsory to paste the word(s) indicating the *pair of concepts*, the *relation* and select the *relation type* (positive versus negative) of the specific relation that they found. In addition, workers also have the option to remove or modify their current annotation before submitting the task. As the data set is not of high quality – arguments from different internet users, with varying tones and also containing some noise from co-reference resolution process (less than 10%), there is also an option for workers to state whether the sentence contains some errors or not comprehensible. The following Figure (4.2) shows a screenshot of the annotation interface.

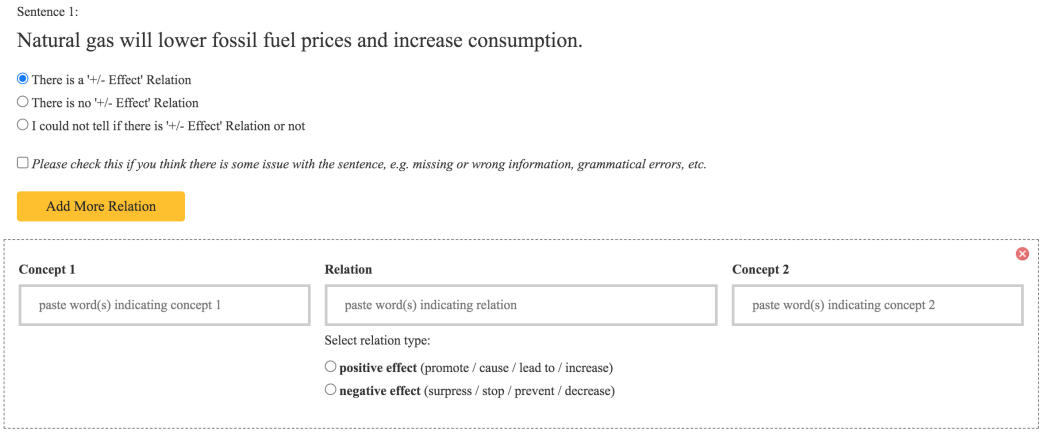

Figure 4.2: Annotation Interface

Annotation Instruction Along with a friendly annotation interface, the task instruction is also essential to help workers understand the task thoroughly, and make the right decision in their work. During the development of annotation task, we introduce it to our friends and colleagues, who check and give feedback about their experience. We arrange our task instruction as followed: First, the definitions of *'effect relation', 'concept', positive - negative 'relation'* are explained, along with prominent examples, which highlight the occurrence of concepts and relations in different colors, so that they look apparent to novice users. Moreover, special instructions are given for some cases, where users may make mistakes, for example, *users should not use their own background knowledge to deduct the relation*, or *they need to consider the position of concept 1 - concept 2 in passive sentence*. Additionally, the fact that *negated statement and neutral relation are not considered as relation* in our task is also stated specifically in the instruction. For complex relations, we show certain types which appear frequently in the dataset, e.g. *parallel, opposite or continuous*; these come with a set of examples along with pre-annotations. Last but not least, 2 video examples of complex effect relation are captured for demonstration purpose, with pop up text to explain step-by-step annotation process. Figures 4.3 shows screenshots of our task description.

4.2.2 Annotation Study

We conduct this process in several steps. First, three experts annotated 100 sentences randomly sampled from the dataset and give feedback for the task. Subsequently, the data from *'filtering'* step (4.1.3) is released for public annotation and the last step is to aggregate the results of annotation.

Expert Annotation is carried out before the actual crowd-sourcing annotation. In this study, 3 experts who understand the task thoroughly use the designed annotation interface in Mturk to annotate 100 randomly sampled sentences from 10,000 sentences of 'concept matching' dataset. After that, we

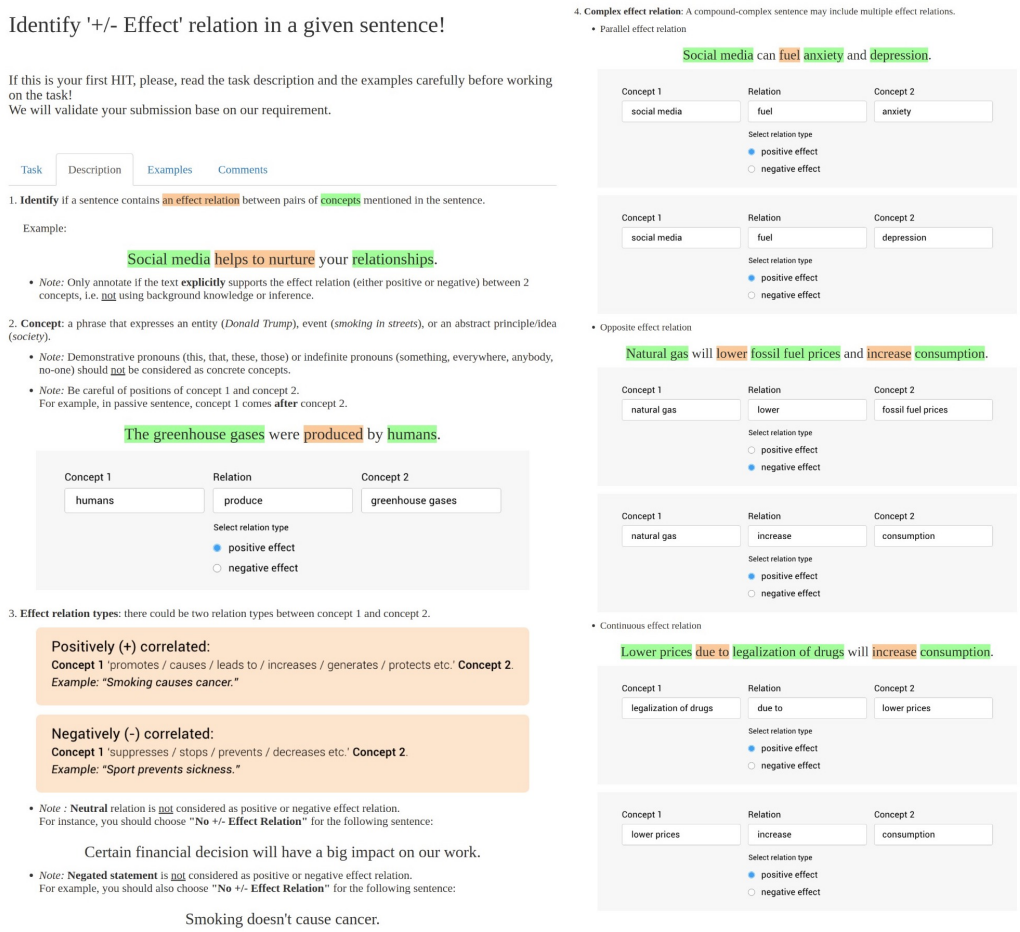

Figure 4.3: Mturk Task Instruction

gather the feedback and suggestion to enhance the interface and instruction. The results of this process show a percentage of approximately 50% effect relation.

Public Annotation We publish 1,937 sentences from filtered-matching sentences in 4 batches, each contain about 500 sentences. Each sentence require 3 workers, who have at least 98% approval rate and come from English speaking countries like USA, Canada and England and the annotation process is closely supervised. For the first batch, we check the distribution of answers from workers, to analyze if they have a good effect detection rate (from 40%

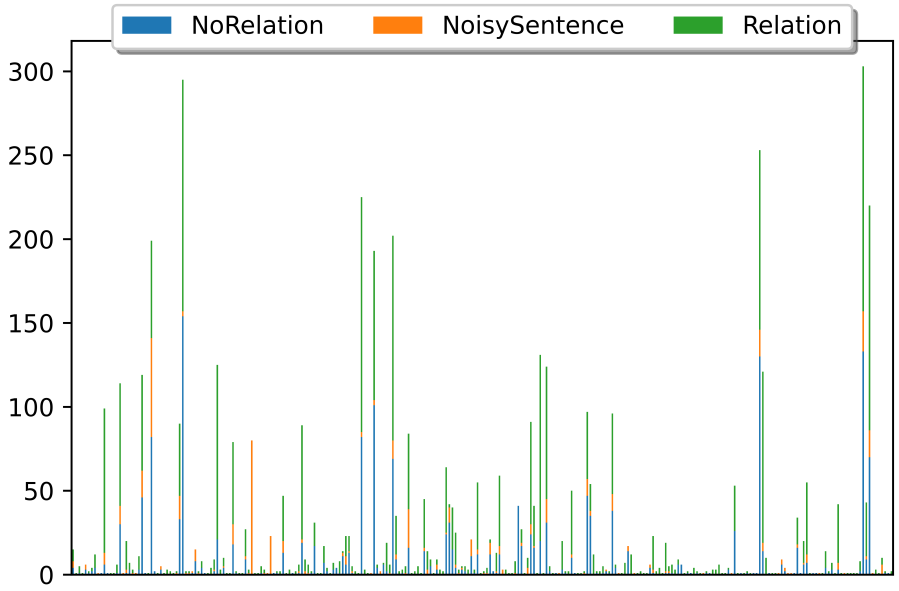

Figure 4.4: Distribution of workers per HITs with their answers for effect detection

to 70%), which align with the annotation from experts. After that, we check the annotations of each worker briefly to evaluate their levels of understanding of the requirement. We approve most the annotations from those with a good distribution of effect relation and high quality annotated HITs. Subsequently, we reject those with 0% of effect detection and large number of annotated HITs, considering them as *spammers*. After obtaining the first version of reviews, we aggregate the results of effect detection and automatically approve all the annotated HITs with same answer as aggregated. We also relax some rejections, where the answers align with the approved answers. For each rejection, we state the feedback clearly in order to communicate with the worker for better annotation quality in the next batches. The rejected HITs are then republished for annotation. In later review cycles, we approve almost all tasks done by workers with high approval rates in the reviewed HITs. We repeat this process 2 to 3 times for each batch, with later batches taking less time due to workers gaining good approval rates and better understanding of the task. The whole process takes 1 week. In total, there are 285 workers working for our

task, who obtain approximately 75% approval rate. In average, the workers take from 15 to 75 seconds to complete a task. The distribution of approved answers per worker for effect detection is depicted in Figure 4.4. As observed, the workers who received good acceptance rate should have high rate of effect detection. Out of 1,912 sentences published for annotation, we obtained **1,485** sentences with sufficient number of workers (3) for aggregation calculation.

4.2.3 Result Aggregation

After obtaining the annotated HITs from workers, we combine them using *majority voting*. We focus on the following aspects of the annotated sentences:

- Whether the sentence express an effect relation – **Effect Detection**
- Whether the sentence express more than one effect relation, i.e. single vs multiple relation(s) – **Multiple Relation Detection**
- Whether the sentence with effect relation contains a positive relation – **Positive Relation Detection**
- Whether the sentence with effect relation contains a negative relation – **Negative Relation Detection**

Agreement scores Table 4.2 describes the Krippendorff agreement scores among experts and workers concentrating on these perspectives. In general, public has lower agreement compared to experts, with an overall fair agreement (0.21 to 0.40). It is also notable that the workers sometimes annotate the incorrect relation type. This could be explained when looking through the annotations, some workers would annotate a positive relation as negative, if this relation is bad in general, e.g. Smoking causes cancer. For this reason, the relation type related sub-tasks from the workers have substantially smaller annotation score compared to the experts, who understand the task thoroughly. Moreover, due to the simple nature of most of other Mturk tasks, the workers also have the tendency not to annotate more than one effect relation. For that reason, the annotated results from public reveal a poor agreement on Multiple Relation Detection.

Table 4.2: Annotation Agreement

	Krippendorff Agreement scores			
	Effect Detection	Positive Relation Detection	Negative Relation Detection	Multiple Relation Detection
Expert	0.34	0.66	0.70	0.28
Public	0.27	0.31	0.36	0.03

Aggregation results are achieved by majority voting. Besides, since annotating more than one relation is optional and neglected by some workers, we consider a sentence having 'multiple' relation as long as at least 1 worker detect various relations in the sentence. The numbers of occurrence of each types, along with its percentage are gather in Table 4.3. For comparison, we put the annotation results achieved by Al-Khatib et al. [2020] on claims – old dataset and our aggregation results of experts and public annotations on argumentative text – new dataset side by side. As seen, our methods lead to a higher percentage of occurrence of effect relation (62% compared to 37%). Furthermore, new dataset contains nearly three times number of negative relation type, with experts found 60% and public found 62% negative relations among sentences with relations. It is also noteworthy that our updated task put an emphasis on sentences with more complex structures and multiple relations. For multiple relations, we find 25% of sentences with relation express more than one relation.

Analysis of instances We manually examine some examples of the newly annotated dataset and compare them with the old annotated dataset. Some interesting observations are made regarding the complexity and how multiple relations are formulated.

- **Complexity**: When comparing the instances that discuss the same topic, sentences from the new dataset express a much more complex structure. For example, while a claim about the topic 'marijuana' in the

Table 4.3: Aggregated Annotation Statistics

Comparison of annotation results						
	Old dataset		New dataset			
			Experts		Public	
	#	%	#	%	#	%
Effect Detection						
Overall	4740	100	80	100	1324	100
Relation	1736	37	48	60	819	62
No Relation	3004	63	32	40	505	38
Relation Type						
Overall	1736	100	48	100	819	100
If Positive	1287	74	29	60	486	59
If Negative	390	23	29	60	507	62
Multiple Relation						
Overall	-	-	48	100	819	100
Single	-	-	34	71	607	75
Multiple	-	-	14	29	202	25

old dataset could be '*Marijuana* use can lead to *cancer*', the new dataset provides the instance: '*Marijuana* has the capability of stopping *cancer*, curing *glaucoma*, reversing *effects of tobacco* and ameliorating *lung health*, decreasing *anxiety*.'

- **Multiple relations**: The most common multiple-relation type (refer to figure 3.2) in new dataset is parallel relation. This could be expressed through several concepts sharing the same relation signal ('lead to', 'cause', 'reduce', 'prevent') and the concepts are jointly mentioned by the use of connecting signal such as comma sign, 'such as', 'and', etc. Another way of introducing multiple sentences is through using a variety of relation signals. As shown in the previous example, many different cues for negative relations are present such as 'stop', 'cure', 'reverse', 'ameliorate', 'decrease'. By employing numerous effect relation signals, a sentence could also demonstrate mixed or opposite relation

types. For example, sentence '*GM foods* are safe for human consumption, reduce *pesticide*, increase *yield* and decrease *cost*, combat *global warming*.' shows different effects related to 'GM foods', including both positive and negative ones. In addition, passive expression is also often used to link phrases which share the same concept. This is demonstrated in the instance '*Marijuana* can relieve certain types of *pain*, *nausea*, *vomiting* and other symptoms caused by such *illnesses* as *cancer* or by the *harsh drugs*.' Interestingly, the continuous effect relation defined previously is scarcely found in this dataset. Instead, we could loosen the definition by integrating those cases where concepts appear to be more complex or ambiguous. Let's take this sentence as an example: '*Genetic screening* for the embryos can reduce the chance of *giving birth* to *more than one child*; because *clinics* now want to *prevent this by planting one embryo* at a time and they have to do this through *genetic screening*'. Considering this instance, we could view the extracted relation (genetic screening, negative relation, giving birth to more than one child) as a concept (representing a fact) which has some positive effect on 'clinic planting one embryo'. For this specific case, a more detailed study of concept identification could be done to determine a suitable model for these kinds of sentences.

Chapter 5

Evaluation

Based on the aggregation results, we train several classifiers using a pre-trained **XLNET** model, which had the most stable performance in our previous comparison. For each dataset, we arrange the sentences by topics and divide them into 80% training, 20% testing. By comparing classifiers trained on the *old*, *new* and *combined* datasets, with both *masking* and *non-masking* sentences across the different test sets, we obtain a comprehensive overview of our classifiers' performance. We obtain *F1 scores* of up to **89%** for the *effect* detection task and up to **92%** for the *relation type* detection task. In general, the *new* training dataset improves our classifiers' generalization capacity, and the *masking* strategy proves essential for predicting instances with complex patterns.

5.1 Experiment Setup

From previous work by Al-Khatib et al. [2020], we already collected **4,710** labelled instances. We call this the *'old'* dataset. Based on the annotation agreement results described in the previous chapter, we acquire a *'new'* dataset of **1,324** instances with labels. In order to study the impact of the new dataset—along with that of variations in data processing strategy—on classification performance, we set up the experiments outlined below.

5.1.1 Types of Classifiers

In previous experiments, we have trained classifiers for two distinct tasks: First, **detecting** whether or not a sentence has an **effect relation**, and second, classifying whether the detected effect is positive or negative. In this work, we consider the possibility of *multiple effects* occurring in a single sentence. Therefore, we reformulate the second task as two separate problems: that of **detecting positive relations** and that of **detecting negative relations**. Under this formulation, one sentence may express both a positive and a negative relation, simultaneously.

5.1.2 Types of Datasets

Combining datasets: For each classifier, we have basically two datasets, *old* and *new*. Additionally, we combine the training and test sets from each to create new, combined training and test sets. For the *Effect Detection* task, we would like to analyze the impact of the amount of available training data, by splitting the new training set into four parts, and **progressively combining** them with the old training set to train classifiers.

Multiple relations: For the *Relation Type Detection* task, we examine the influence of multiple relations on the *combined* dataset by training two sets of classifiers on the old plus new training datasets: one using only instances with a **single relation**, and another using both **single- and multiple-relation** instances.

5.1.3 Training and Test Sets

Split: To reasonably compare the results of different training runs, we use the same training and test dataset split as was used by Al-Khatib et al. [2020]. All datasets are split into **80%** for training and **20%** for testing.

Topic separation: The old dataset is ordered by topics and divided in such a way that the training and test set's *topics* are *not overlapping*. As a result, the classifiers trained on these samples are considered *topic independent*. To acquire a similar topic separation, we implement a **topic matching** between the *new* dataset and the *old* dataset. Since the sentences in the new dataset are not labeled with specific topics, we employ the *concepts* in those sentences for the purpose of topic matching: we measure the similarity between matched concept pairs in these sentences with the topics in the old dataset, and assign each new-dataset sentence the best matching topic thus identified. Subsequently, we arrange those sentences by the topic order of the *old* dataset. Therefore, we obtain for the new datasets an equivalent topic distribution in training and test sets.

5.1.4 Masking

As masking is introduced in the *active learning* step, complete training and testing of classifiers are carried out to analyze the effect of masking on the performance of classifiers. For the new dataset, as the **relation patterns** are also annotated, we enhance the previous *masking strategy* by **unmasking** tokens which are annotated by workers as a representation of relations in the sentence.

5.2 Evaluation Results

With the experiment setup described above, a number of classifiers are trained and tested across different datasets. By comparing the micro-average F1 scores, we can draw several conclusion.

5.2.1 Effect Detection Classifiers

Table 5.1 shows the results of the experiments on the effect detection classifier. Each table cell shows an F_1-score; the training set varies across the table's

rows, and the test set varies across the columns. For both training and test set, we compare variants containing masked samples (marked M in the table), and containing non-masked samples (marked x). We vary the make up of the training set by comparing using only the *old* training set, only the *new* training set, as well as all of the old training set *combined* with successively larger fractions (25% up to 100%) of the new dataset. For the test set, we compare using either the old or new test set by itself, or both test sets combined. In general, more training data tends to improve classification performance. The classifier trained on both full training sets ("Combined 100%") holds the best record when trained with non-masking sentences and tested with old and combined non-masking sentences (**89%** and **85%** respectively). For a test set with masking, the classifiers trained on the combined training sets with masking hold the best records of **85%**.

Impact of combining datasets: As seen from the table, the results improve gradually when we add more of the new dataset across all test sets, for both masking and non-masking samples. We see the most improvement when testing with the new non-masking dataset, with a leap of 5 percentage points in F1 score when adding 25% of the new dataset, and up to **12 percentage points** when adding all of the new dataset. From this, we conclude that the new training set helps improve the generalization capability of the classifiers.

Impact of masking: Training on the masking dataset often leads to slightly (1 to 4 percentage points) lower F_1-score when testing on the non-masking dataset. However, this type of classifier achieves comparable or better results on the masking test set. The classifier trained on the full combined training set with masking reaches a peak accuracy of **85%** on the old masking test set and **82%** on the combined masking test set. This score could be considered as our classifier's confidence when being applied to sentences discussing new topics and containing unfamiliar concepts.

Table 5.1: Comparison of Effect Detection Classifiers

Training Set	F_1 score		Test Set: Old x	Old M	New x	New M	Combined x	Combined M
	Old	x	0.88	0.84	0.63	0.57	0.82	0.78
		M	0.85	0.83	0.62	0.58	0.80	0.77
	New	x	0.74	0.77	0.71	**0.75**	0.74	0.77
		M	0.67	0.69	0.62	0.70	0.66	0.70
	Old + 25% New	x	0.86	0.83	0.68	0.59	0.82	0.78
		M	0.87	0.84	0.66	0.68	0.83	0.80
	Old + 50% New	x	0.87	0.83	0.69	0.60	0.83	0.78
		M	0.87	**0.85**	0.65	0.68	0.82	0.81
	Old + 75% New	x	0.88	0.80	0.70	0.61	0.84	0.76
		M	0.87	0.83	0.67	0.67	0.82	0.79
	Old + 100% New	x	**0.89**	0.83	**0.75**	0.62	**0.85**	0.78
		M	0.88	**0.85**	0.70	0.70	0.84	**0.82**
	Majority Class Baseline		0.64	0.64	0.53	0.53	0.62	0.62
	Al-Khatib et al. [2020]		0.81	-	-	-	-	-

Impact of the new test set: In general, the test set from the *new* dataset complicates the effect detection task. When using this test set alone, the best F_1-score reached for both masking and non-masking data is only **75%**. The new test set also reduces the score achieved by all classifiers by about 4 percentage points when *combined* with the old test set. This result confirms our success in selecting the instances for labelling during active learning.

Impact of testing within the same and across datasets: It is remarkable to witness a significant *drop* in F1 score (25 percentage points) when testing the classifier trained on the old non-masking training set on the new

test set. More surprisingly, the classifier trained on the new non-masking training set achieves moderate results (71%) when testing with the corresponding test set, but performs comparably or better when tested on the *old* dataset (F1 scores of 74% and 77% on the non-masking and masking test sets, respectively). Though training on a small number of instances (1,055 sentences), the classifier trained only on the new dataset has more stable performance across different test sets, which further strengthens our conclusion that our new classifiers have better generalization capability.

5.2.2 Relation Type Detection Classifiers

Relation Type (Positive or Negative) detection classifiers' F1 scores are shown in Tables 5.2 and 5.3. These classifiers are trained on sentences containing relations from the old dataset (1,339 sentences) and new dataset (651 sentences). Overall, the classifiers work quite well, with the best accuracy achieved when training on the *combined* dataset (**92%** when testing on the old dataset, and **89%** when testing on the combined dataset). Training on multiple relations does not change the general performance of the classifiers by much.

Impact of masking: the classifiers trained on the *masking* datasets work comparably well to the *non-masking* ones. In some cases, they outperform their counterparts. For example, positive and negative relation detectors trained on the new dataset with masking, and the combined dataset with only single relations exceed the same classifiers trained on non-masking sentences when testing with the new dataset (difference of up to 9 percentage points). This shows the importance of masking for relation type detection to help the classifiers focus only on the *relation pattern* itself.

Impact of the new test set: By applying the new test set to different classifiers, we achieve scores of up to **86%** for *positive* relation detection and up to **83%** for *negative* relation detection. Similar to the effect detection clas-

Table 5.2: Comparison of Positive Relation Detection Classifiers

	F_1 score		Test Set: Old x	Old M	New x	New M	Combined x	Combined M
Training Set	Old	x	0.91	0.90	0.64	0.61	0.82	0.81
		M	0.90	**0.91**	0.72	0.74	0.84	0.85
	New	x	0.81	0.81	0.77	0.78	0.80	0.80
		M	0.74	0.87	**0.86**	0.79	0.78	0.85
	Old + New (Single)	x	0.90	**0.91**	0.77	0.77	0.86	0.86
		M	**0.92**	0.89	0.83	**0.84**	**0.89**	0.87
	Old + New (Single + Multiple)	x	0.91	**0.91**	0.74	0.71	0.86	0.84
		M	0.91	**0.91**	0.83	0.82	**0.89**	**0.88**
	Majority Class Baseline		0.79	0.79	0.69	0.69	0.78	0.78
	Al-Khatib et al. [2020]		0.86	-	-	-	-	-

sifiers, combining the new and old test sets *decreases* the performance of the classifiers overall (a drop of 3 percentage points). In more detail, it is noteworthy that testing on the *new* dataset shows that *negative* relation detection has marginally *worse results* compared to *positive* relation detection. One possible explanation for this is the fact that the old dataset is less balanced, positive relations being in a stronger majority compared to the new dataset. Further, *multiple relations* with mixed effect directions might confuse the classifier in detecting whether there is negative relation or not.

Impact of testing within the same and across datasets: When comparing the results of training on the old dataset and testing on the new dataset, we observe a drop of 27 percentage points for the positive relation detector and 21 percentage points for the negative relation detector. Conversely, the classifiers trained on the new dataset of only 651 instances give a more stable prediction with F_1-scores of up to **87%** (*masking positive* detector) and **81%**

Table 5.3: Comparison of Negative Relation Detection Classifiers

	F_1 score		Test Set: Old x	Old M	New x	New M	Combined x	Combined M
Training Set	Old	x	0.90	0.90	0.79	0.76	0.86	0.85
		M	0.90	**0.91**	0.73	0.78	0.85	0.87
	New	x	0.77	0.80	0.75	0.79	0.77	0.80
		M	0.77	0.81	0.81	0.76	0.79	0.79
	Old + New (Single)	x	**0.92**	0.90	0.74	0.74	0.86	0.85
		M	0.90	0.90	0.82	**0.81**	0.87	**0.87**
	Old + New (Single + Multiple)	x	0.91	0.89	**0.83**	0.79	**0.88**	0.85
		M	**0.92**	**0.91**	0.72	0.75	0.86	0.86
	Majority Class Baseline		0.79	0.79	0.66	0.66	0.77	0.77
	Al-Khatib et al. [2020]		0.86	-	-	-	-	-

(*masking negative* detector), respectively. When inspecting the recall of the classifiers on the new test set, we observe that the classifiers trained on the old dataset tend to only recognize *positive* relations (with the recall for *negative* relations only reaching a maximum of *61%*), whereas training on the combined dataset helps the *negative* relation detector achieve a recall of up to *81%*. This improvement in recall illustrates the effect of the better balancing of positive and negative relations inside the new training set.

5.3 Manual Inspection

To confirm that our classifiers learn properly and label instances based on the correct relation patterns, we collect several sentences on the internet related to a new topic ('*Covid-19*') not contained in the dataset, in order to assess the classifiers' efficiency. We load the model from *ktrain* and use the

Figure 5.1: Distribution of tokens' importance toward predictions for correctly labelled sentences by the effect detection classifier trained on the old dataset.

built-in function to interpret how important a token is towards the prediction. We include the figures for ease of comparison. For reference, the more bright and green highlight of the token is, the more contribution it makes towards the prediction. By contrast, tokens highlighted in red indicate support for the opposite label.

5.3.1 Effect Detection

When looking at Figure 5.1, we can see that the classifier trained on the old dataset works quite nicely with those sentences containing familiar relation patterns. For example, it correctly labels the sentences come with these patterns: '*due to*', '*lead to*', '*contribute to*', '*increase*', '*reduce*', '*cause*'. Sometimes, even a new pattern like '*prompt*' is still understood correctly by the classifier. This could be explained by our use of transfer learning, and the fact that the pre-trained models already have a good general comprehension of language regarding synonyms and similar expressions.

There are three instances where the classifier fails to detect the relation,

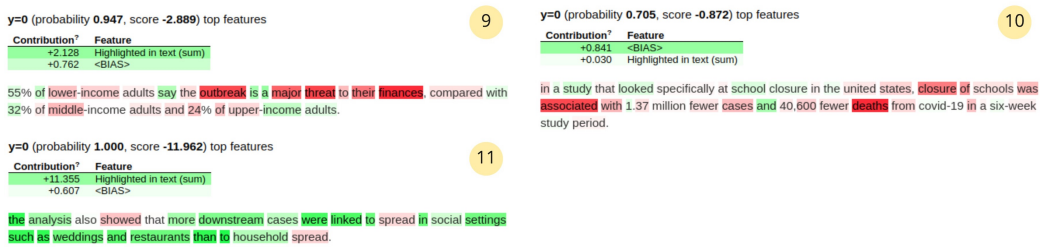

Figure 5.2: Distribution of tokens importance toward prediction of *incorrectly* labelled sentences by *effect detection* classifier trained on *old dataset*

with relations indicated by '*major threat*', '*associated with*' and '*linked to*' (Figure 5.2). For the first 2 patterns, although the classifiers recognize that the mentioned token sequences strongly indicate some relation, it still labels the sentence as '*no relation*'. For the final pattern, the classifier neglects '*linked to*' and considers this as an indicator of '*no relation*'.

Next, we apply the classifiers trained on the *combined* dataset to the sentences with effect relations that the previous classifier *fails* to recognize, and observe whether any improvement is made (Figure 5.3). Interestingly, all three sentences with effect relations are labelled correctly. The patterns mentioned earlier—'*associated with*', '*major threat*' and '*linked to*'—are labelled as indicators of effect relations. However, besides these tokens, several other tokens are also labelled as indicators. This confusion seems to be resolved by training on *masked* sentences. When we predict the labels of these instances with classifiers trained on the combined masked training dataset (Figure 5.4), even without masking the testing instance, the classifier still puts more focus on the *relation pattern* itself, and less emphasis on other tokens.

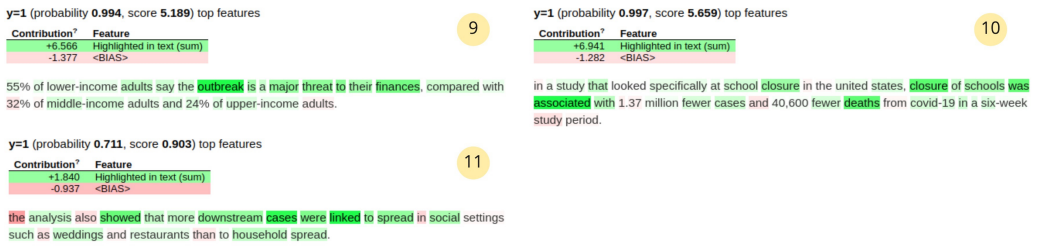

Figure 5.3: Distribution of tokens' importance toward predictions for correctly labelled sentences by the effect detection classifier trained on the combined dataset.

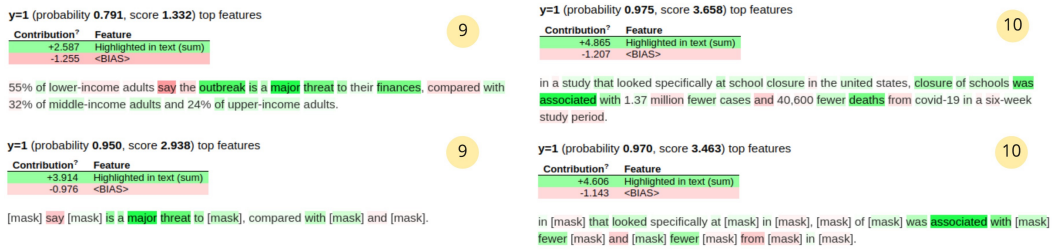

Figure 5.4: Distribution of tokens' importance toward predictions for correctly labelled sentences by the effect detection classifier trained on the combined *masking* dataset.

5.3.2 Relation Type Detection

As we achieve equivalent results for positive and negative relation detection, we analyze only the negative relation classifiers. Another goal of our inspection is to check whether the new classifiers are better at detecting new patterns of negative relations, and at dealing with sentences of with mixed relation types. Based on the word importance distribution produced by the classifier trained on the old dataset (Figure 5.5), we can conclude that this classifier can identify negative relations with familiar patterns such as '*prevent*' and '*reduce*'. However, when it comes to a more complex pattern as in sentence number 3 ('*major threat*') in Figure 5.5, it yields a misleading detection. For sentence number 4, even though '*reduce*' is a clear indicator for a negative relation, the classifier attends to positive-relation tokens such as '*increase*' to produce the label. This is due to the fact that the old dataset only contains sentences with

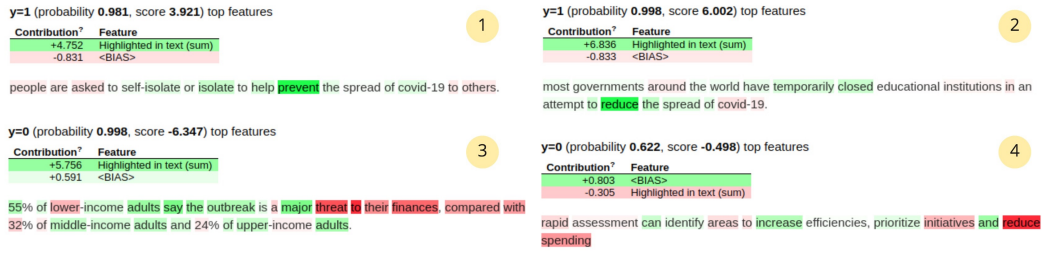

Figure 5.5: Distribution of tokens' importance toward predictions by the negative relation detection classifier trained on the old dataset. Sentences 1 and 2 are correctly labelled; sentences 3 and 4 are incorrectly labelled.

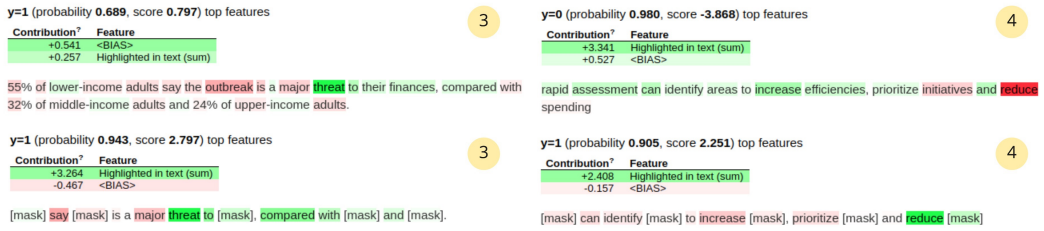

Figure 5.6: Distribution of tokens' importance toward prediction by the negative relation detection classifier trained on the combined dataset. Sentence 3 and the masked version of sentence 4 are correctly labelled.

a single relation, and the classifier trained on this dataset only gives a single answer of either positive or negative relation. On the other hand, the new dataset has sentences with multiple relations, i.e. a single sentence could have both positive and negative relation.

When applying the classifier trained on the combined dataset to the aforementioned sentence number 3, we obtain the correct label (Figure 5.6). Nevertheless, it is noteworthy that the complex sentence structure complicates the prediction, and reduces the classifier's confidence. When the same sentence is masked, we achieve a much higher confidence (0.943 compared to 0.689, Figure 5.7). Remarkably, only the masked version of sentence 4, with multiple relations, earns the correct label from the combined non-masking classifier. This underscores the importance of masking irrelevant tokens not only in the training data but, in the test data as well, especially for those sentences with a more complex structure, patterns, and multiple relations.

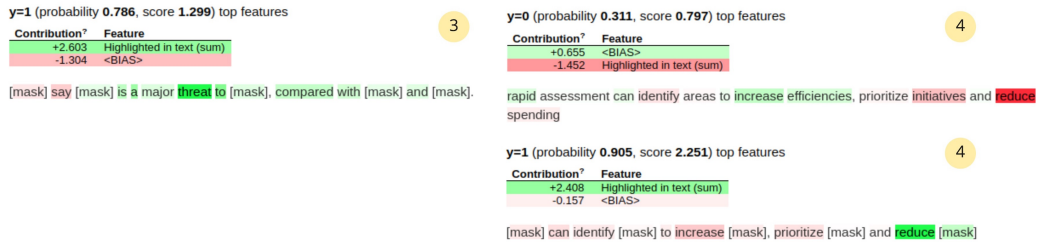

Figure 5.7: Distribution of tokens' importance toward prediction of *correctly* labelled sentences by the negative relation detection classifier trained on the combined masking dataset.

Chapter 6

Conclusion

The focus of this thesis is developing a new dataset of annotated effect relations and training neural-based classifiers on labelled datasets. The datasets and classifiers developed are intended for further automatically extracting new instances of effect relations and classifying relation types.

Our implementation is divided into dataset construction, sentences filtering, crowd-sourcing annotation and evaluation experiments.

Dataset Construction: To build a new dataset, we obtain the manually annotated data from Al-Khatib et al. [2020] as seed instances. After observing the pairs of concepts extracted from these instances, we realize the need to group the concepts and separate the multi-concept to individual ones. This refining step facilitates the alignment between concepts and new sentences. By carefully inspecting the labelled instances, we have developed a modified model for effect relation, which take into consideration 'multiple relations'.

For extracting new relations from arguments, args.me dataset is acquired. Subsequently, long and complex texts are processed and simplified using a distributed system. We align full string stemmed tokens appearing in the concept with sentences to find matching instances. When the matching process is finished, we obtain about 30 thousand sentences, which reduces to 10 thousand sentences after removing noisy instances.

Relation Classification: Assuming that the collected sentences are candidates for potential effect relation, we implement a filtering strategy based on active learning to selectively sample sentences for manual annotation.

We use ktrain (Maiya [2020]), a framework to easily obtaining and fine-tuning pre-trained models to compare the performance of several deep learning models. After the classifiers are trained, distribution of word's importance towards prediction suggests that they sometimes falsely learn to recognize some concepts patterns, instead of the relation patterns. Hence, a masking scheme of these kinds of tokens is implemented. To filter the candidate sentences, we apply the least confidence sampling and disagreement sampling by the best classifiers from the previous step. This process results in 2,000 sentences selected for annotation.

Crowd-sourcing Annotation: An annotation interface with thoughtful instruction is developed for annotating the newly sampled sentences. The annotation interface allows workers to annotate concept pairs, relations and relation types of detected effect relations. Moreover, they could also add multiple relations if that emerges in the sentence. To avoid misunderstanding, terminologies are explained in the description clearly with examples and specific notes are shown for cases where users tend to annotate wrongly.

Before public annotation is carried out, 3 experts use the interface to annotate 100 sentences. Later, the aggregation of these sentences is applied for comparison with the results made by non-expert workers. We implement quality supervision on annotation tasks, in which tasks done by spam workers are usually rejected. When the final annotation results are obtained, we calculate agreement scores and aggregate results. The results achieve a higher percentage of relation instances and more balance between positive and negative relation types.

Evaluation Experiments: Based on the aggregation results, we train several classifiers using a pre-trained XLNET model, which had the most stable performance in our previous comparison. For each dataset, we arrange the

sentences by topics and divide them into 80% training, 20% testing.

We obtain a comprehensive overview of our classifiers' performance through training and testing across different types of dataset. Remarkably, F1 scores reach up to 89% for the effect detection task and up to 92% for the relation type detection task. By comparison of testing results and manually inspecting the word's importance for prediction, we have shown that the new training dataset improves our classifiers' effectiveness, and the masking strategy proves essential for predicting instances with complex patterns.

6.1 Improvements and Future Work

While our classification approach constitutes a notable improvement over previous work, we see various avenues for further refinement. First, the co-reference resolution technique we used is sometimes incorrect, leading to noisy training instances. Here, a superior text simplification framework could be implemented to improve the quality of instances with complex sentences, in particular.

Second, our annotators often gave diverse answers for the rated sentences, but we aggregated by majority vote. Instead, we could aggregate the answers as percentages, and assign these percentages as the target for the classifiers to predict. This way, the annotators' degree of agreement or disagreement could be incorporated into the training process, which in turn could lead to better calibration of the classifiers' confidence.

Third, while we considered different classification tasks in our study—in particular, detecting the presence of an effect relation on the one hand, and classifying the direction of the relation on the other—the classification models we trained to address these tasks are completely separate from each other. As an alternative, a model could be trained jointly for all tasks in a multi-task learning setting. This has been shown to improve the performance of machine learning models in other NLP tasks, and may prove beneficial here, as well.

Fourth, we focus only on training detecting classifiers, and omit the task of

recognizing the concepts in the sentence. Future research could build on an aggregation of our annotated concept pairs, especially for multiple concepts. As a result, the annotations collected as part of our study could lay the groundwork for a full 'effect relation extraction' framework, which would not only detect and classify relations, but also automatically extract the participating concept pairs in unseen relations.

Finally, a continuation of our study could apply our trained classifiers to a much bigger dataset of web text. Such an effort could be undertaken to support a large-scale evaluation study, to gather ever larger amounts of additional training data in a distant-supervision setting, or both.

Bibliography

Yamen Ajjour, Henning Wachsmuth, Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. Data acquisition for argument search: The args. me corpus. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 48–59. Springer, 2019. 3.1.2

Khalid Al-Khatib, Yufang Hou, Henning Wachsmuth, Charles Jochim, Francesca Bonin, and Benno Stein. End-to-end argumentation knowledge graph construction. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*, 2020. (document), 1, 1, 2, 2.3, 2.3, 2.2, 3.1.1, 4.1.2, 4.2.3, 5.1, 5.1.3, 5.1, 5.2, 5.3, 6

Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988. 2.5

Maria Becker, Michael Staniek, Vivi Nastase, and Anette Frank. Enriching argumentative texts with implicit knowledge. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10260 LNCS:84–96, 2017. ISSN 16113349. doi: 10.1007/978-3-319-59569-6_9. 2.3

Rebecca S Bjork. Argumentation and debate: Critical thinking for reasoned decision making. by austin j. freeley, 1994. 2.1

Piero Andrea Bonatti, Stefan Decker, Axel Polleres, and Valentina Presutti. Knowledge graphs: new directions for knowledge representation on the semantic web. *Report from Dagstuhl Seminar 18371*, 2019. 2.2

Matthias Cetto, Christina Niklaus, André Freitas, and Siegfried Handschuh. Graphene: Semantically-linked propositions in open information extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2300–2311. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/C18-1195`. 3.2.1

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994. 2.5

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019. 2.6

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2.6

Roxanne El Baff, Henning Wachsmuth, Khalid Al Khatib, Manfred Stede, and Benno Stein. Computational argumentation synthesis as a language modeling task. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 54–64, 2019. 2.1

Ruocheng Guo, Lu Cheng, Jundong Li, P. Richard Hahn, and Huan Liu. A survey of learning causality with data: Problems and methods. *CoRR*, abs/1809.09337, 2018. URL `http://arxiv.org/abs/1809.09337`. 2.3

Ulrike Hahn, R. Bluhm, and Frank Zenker. *Causal argument*, chapter 25. Oxford University Press, 05 2017. 2.1

Xu Han, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2245, 2018. 2.4

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2.6

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics, 2011. 2.4

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020. 2.2

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017. 4.1.2

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*, 2020. 2.2

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016. 2.6

Jonathan Kobbe, Juri Opitz, Maria Becker, Ioana Hulpus, Heiner Stuckenschmidt, and Anette Frank. Exploiting Background Knowledge for Argumentative Relation Classification. In Maria Eskevich, Gerard de Melo, Christian Fäth, John P. McCrae, Paul Buitelaar, Christian Chiarcos, Bettina Klimek, and Milan Dojchinovski, editors, *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 8:1–8:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-105-4. URL `http://drops.dagstuhl.de/opus/volltexte/2019/10372`. 2.3

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019. 2.6

David Lewis and William Gale. Training text classifiers by uncertainty sampling. In *seventeenth annual international ACM SIGIR conference on research and development in information retrieval*, pages 3–12, 1994. 2.5

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, 2016. 2.4

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 2.6

Arun S Maiya. ktrain: A low-code library for augmented machine learning. *arXiv preprint arXiv:2004.10703*, 2020. 4.1.1, 6

Aditya Malte and Pratik Ratadiya. Evolution of transfer learning in natural language processing. *arXiv preprint arXiv:1910.07370*, 2019. 2.6

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8, 2011. 4.1.2

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009. 2.4

Isaac Persing and Vincent Ng. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394, 2016. 2.1

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. Cotype: Joint extraction of typed entities and

relations with knowledge bases. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1015–1024, 2017. 2

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 4.1.1

Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases*, pages 148–163, 2010. 2.4

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 2.6

Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009. 1, 2.5

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992. 2.5

Alisa Smirnova and Philippe Cudré-Mauroux. Relation extraction using distant supervision: A survey. *ACM Computing Surveys (CSUR)*, 51(5):1–35, 2018. 2.4

Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 2.3

Christian Stab and Iryna Gurevych. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659, 2017. 2.1

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language*

processing and computational natural language learning, pages 455–465. Association for Computational Linguistics, 2012. 2.4

Orith Toledo-Ronen, Roy Bar-Haim, and Noam Slonim. Expert stance graphs for computational argumentation. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 119–123, 2016. 2.1, 2.3

Lukas Trautner. Exploiting Argumentation Knowledge Graphs for Argument Generation. Bachelorarbeit, Bauhaus-Universität Weimar, Fakultät Medien, Computer Science and Media, March 2020. 2.3

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. 2.6

Henning Wachsmuth. Lecture notes in introduction to computational argumentation, April 2019. 2.3

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187, 2017a. 2.1

Henning Wachsmuth, Martin Potthast, Khalid Al-Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59, Copenhagen, Denmark, September 2017b. Association for Computational Linguistics. doi: 10.18653/v1/W17-5106. URL `https://www.aclweb.org/anthology/W17-5106`. 2.1

Henning Wachsmuth, Manfred Stede, Roxanne El Baff, Khalid Al Khatib, Maria Skeppstedt, and Benno Stein. Argumentation synthesis following

rhetorical strategies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3753–3765, 2018. 2.1

Douglas Walton, Christopher Reed, and Fabrizio Macagno. *Argumentation Schemes*. Cambridge University Press, 2008. doi: 10.1017/CBO9780511802034. 2.1

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 2.6, 4.1.1

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019. 2.6

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1753–1762, 2015. 1, 2.4