Annotating and Automatically Extracting Task Descriptions from Shared Task Overview Papers in Natural Language Processing Domains

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Anna Martin

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Ted Pedersen

May 2022

© Anna Martin 2022

Acknowledgements

There are many who have supported me in my efforts that I would like to thank.

In particular I would like to thank my thesis advisor Dr. Ted Pedersen. I have learned an incredible amount from him in multiple capacities through attending his courses as a student, as a teaching assistant, and through his guidance on my thesis research. I am grateful for his continuous feedback, patience, and editing assistance throughout the writing process. I am also grateful for his help through the PhD application process.

I would also like to thank Director of Graduate Studies Dr. Arshia Khan for her guidance during seminar I and II. She helped me learn how to organize literature reviews and to articulate the rationale and impact statements for my work.

I am quite grateful to Dr. Jennifer D'Souza for her help in developing our corpus guidelines and annotations.

Thank you to my thesis committee Dr. Alexis Elder and Dr. Andrew Sutton for their valuable time and feedback on my work. Thank you also to Dr. Elder for helping me find resources to support my Ethical Considerations section.

Many thanks to my family for their support through the last two years, particularly Tony Boyle.

Dedication

I dedicate my master's thesis to my grandfather Hal Martin, whose lifelong scholarship and contributions to mathematics set an example that I hope to follow all my life.

Abstract

The rapid growth rate of scientific literature makes it increasingly difficult for researchers to keep up with developments in their field. This is a problem that can be addressed by structuring academic papers according to information units that go deeper than keywords. The need to efficiently structure scholarly documents so that they are machine operable necessitates the creation of machine readers to extract and classify bits of fine grained scientific information. This process requires the development of gold standard corpora of annotated scholarly work. For this thesis we developed a gold-standard corpus of task description phrase annotations from Shared Task Overview papers and trained a text classifier on the resulting dataset. The annotation project consisted of: developing a set of annotation guidelines; reading and annotating the task descriptions of 254 Shared Task Overview papers published in the ACL Anthology; validating our guidelines by measuring the Inter-Annotator Agreement; and digitizing the resulting corpus such that it could be used as a resource in machine learning projects. The resulting dataset comprises 254 full text papers containing 41,752 sentences and 259 task descriptions. In our second and final validation we achieved a strict score of 0.44 and a relaxed score of 0.95, measured using Cohen's kappa coefficent. We then used this resource to facilitate the training and development of a classifier to perform automatic identification of shared task descriptions. For preprocessing, we improved the balance between negative and positive samples by eliminating every paper section that does not contain a task description. During our machine learning experiments we trained and validated 18 different sentence classification models using a variety of text encodings and hyperparameter settings. The best performing model was SciBERT, which achieved an F1 score of 0.75 when applied to the reduced test set.

Contents

List of Tables

List of Figures

1 Introduction

Due to the rapid growth rate of scientific literature, it is necessary to find ways to improve how scientific information is structured to make it easier for researchers to stay up-to-date in their fields. To do this, natural language processing and information extraction techniques can be applied towards structuring scientific data (such as concepts, terms, and relations between concepts and terms) into machine-actionable form. Information that is organized in a knowledge graph is much easier for machines to process and use for downstream tasks than raw text taken from PDFs. Once the scientific information is used to populate scientific knowledge graphs, it can be more easily accessed to provide advanced functionality for digital libraries to improve the literature discovery phase of research.

Digital libraries are essential for widening access to scholarly work by providing online access to peer-reviewed academic journals. However, the discovery process of the research cycle has become more difficult; keyword searches of digital libraries can yield thousands of results, many of which may not be relevant to the researcher. Furthermore, the interface between resource and researcher might not provide enough information to the reader to help them decide if the resource is worth reading.

Digital Libraries such the Open Research Knowledge Graph[1] seek to improve this aspect of the research process by storing academic papers in knowledge graphs that contain both bibliographic information and scientific entities that outline the contributions of the paper. Populating this graph by hand is time consuming, so NLP

[1] https://www.orkg.org/orkg/

information extraction techniques must be used to develop machine readers to annotate the papers for inclusion in the knowledge graph. In order to train such machine readers, it is necessary to have access to hand-annotated, domain-specific datasets. This thesis contributes to scholarly document processing research by providing a new resource for training machine readers. We build on our previous work (Martin and Pedersen 2021), which comprised a contribution to the shared task SemEval-2021 Task 11: NLP Contribution Graph (D'Souza, Sören Auer, and Pedersen 2021). The NLP Contribution Graph shared task addressed the need for datasets and benchmarks for structuring scholarly contributions for inclusion in research knowledge graphs. For more information on this shared task and some of the participating systems, see Section 2.5.

This thesis' primary contribution is the creation of a gold-standard corpus of task description phrase annotations from Shared Task Overview papers. We assembled our task description corpus by searching the Association for Computational Linguistics (ACL) Anthology[2] for shared task description papers, including all SemEval task description papers from the year 2001 to 2021, all CoNLL[3] shared tasks 2000-2020, and shared tasks from workshops hosted by the AACL[4], ACL, EACL[5], EMNLP[6], and NAACL[7] conferences. The dataset contains a total of 254 shared task description papers, each with task description phrases annotated.

Our secondary contribution is the development of a machine learner trained on our corpus to extract task descriptions. We preprocessed the dataset by filtering out sections that did not contain task descriptions to improve the balance between positive

[2] https://aclanthology.org/

[3] https://www.conll.org/

[4] https://www.aclweb.org/portal/

[5] https://2021.eacl.org/

[6] https://2021.emnlp.org/

[7] https://naacl.org/

and negative samples. We ran experiments using 18 different classifiers including five single-learner linear models, three ensemble learners, six types of neural network architectures, and four BERT variants. We trained each of the non-neural learners on four different kinds of text encodings and trained the BERT models exhaustively on 12 different combinations of hyperaparameter settings. Each of the models was trained ten times with a different validation set made up of 10% of the training data. The scores were averaged across all ten runs and the mean score and standard deviation reported in the Results section (Section 5). Each of these experiments was repeated with a variation on the dataset that included features that describe positional and contextual metadata for each sentence. The best performing model was retrained on the training data and tested on an unseen test set resulting in an F1 score of 0.75. The most important takeaway from these experiments is that choices made around how the data is prepared are crucial, particularly for a small corpus with very few positive samples.

2 Background

The research area of Scientific Document Processing (SDP) is concerned with finding approaches to improve the retrieval and structuring of information from scholarly papers in science domains. This concern has largely been motivated by the rapid growth rate of scientific literature, which has been estimated in Larsen and Ins 2010 to be between 2.7 and 13.5% from 1997 to 2006. In Ware and Mabe 2015, it is estimated that roughly 3 million scientific articles are published yearly. Some domains are growing even faster: Dhawan, Gupta, and N. K. Singh 2020 analyzed machine learning research's global output between 2009 and 2018 and estimated a growth rate of nearly 28% per year. According to yearly publication data presented in Li et al. 2020, the mean annual growth rate in the deep learning domain between 2013 and 2019 is 152.9%. This rapid growth has created a ”bottleneck” effect for researchers (D. Buscaldi 2018), and necessitates the application of natural language processing (NLP) and information extraction (IE) techniques towards structuring scientific and bibliometric data into machine-actionable forms. One such approach is to populate or enhance knowledge graphs with scientific entities and relations extracted from scholarly documents. Such scientific knowledge graphs can then be used to power applications that improve access to scientific literature, such as Digital Libraries (Ammar et al. 2018, S. Auer et al. 2020).

2.1 Information Retrieval Origins

Researchers are focused today on automating the process of extracting fine-grained information from documents, due to the impracticality of human-annotating the vast and growing amount of scholarship that exists. But this is not a new impulse; the growth in size of collections of unstructured text in the twentieth century caused a similar pressure that led to the development of information retrieval (IR) systems. According to Sanderson and Croft 2012, initial developments in IR occurred in the 1950s after the invention of the UNIVAC computer. These developments included the use of keywords to index documents in a catalog, and retrieving documents based on ranked relevance of documents to the search query based on statistical measures and keywords.

Many of the methods for improving IR in the 1960s and 1970s described by Sanderson and Croft 2012 are still relevant today (and in fact were applied during the experiments performed for this thesis: see Section 4.2.1). One need was to improve the way that documents were ranked so that ranking algorithms would return more relevant results for a given query. These techniques included: vectorizing the document and query and using the cosine coefficient between the two vectors to determine their similarity; ”relevance feedback”, which assigned relevance to documents based on past searches; document clustering based on similarity; and term frequency - inverse document frequency (tf-idf), (a statistic that measures how relevant a word is to a document based on the word's frequency across the collection and within the document).

The tradition of holding shared tasks to facilitate the creation of datasets and benchmark improvements in NLP also dates back to the 20th century. The TextREtrieval Conference (TREC)[1] has been held annually since 1992 and was originally designed to support researchers in testing IR systems on large scale datasets.

2.2 Scholarly Information Extraction Today

Effective and accurate extraction of scientific information from scholarly documents comes with a set of unique challenges. While modern language models work quite well on generic IE tasks such as those included in the General Language Understanding Evaluation (GLUE) (A. Wang, A. Singh, et al. 2018) and SuperGLUE (A. Wang, Pruksachatkun, et al. 2019) benchmarks, these tools can fall short in the domain of scientific literature, due to three main challenges.

Firstly, the length of documents processed in SDP tend to be longer than the documents provided in traditional NLP tasks. Language models such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2019a) are trained on sentence-level data. The corpora used in the GLUE and SuperGLUE benchmarks present training data at the phrase, sentence, and paragraph level, so BERT and BERT variants perform quite well. However, the analysis of scientific documents requires systems to develop an understanding of the work at the document level, and so requires additional techniques to allow systems to understand long passages of text.

Secondly, most modern language models are trained on generic sources of information, such as novels, Wikipedia, news sites, and Reddit (Devlin et al. 2019a, Radford et al. 2019). The datasets for traditional benchmarks are made up of similar sources, so are simple to process due to the large amount of shared vocabulary with models such as BERT. In contrast, the vocabulary of scientific discourse is highly specialized

[1] https://trec.nist.gov/

to various domains, can have different meanings across domains, and changes rapidly with scientific progress. Scientific vocabularies make heavy use of numbers, symbols, mathematical and scientific notation, abbreviations, and acronyms. Symbols and words may have no meaning outside of the context of the paper (Head et al. 2021), or may defined in a separate document.

Lastly, full understanding of scientific documents must be multi-modal. Significant details may only be present in figures and tables (Milosevic et al. 2016). Furthermore, scholarly discourse is intertextual, requiring an understanding of external scientific context and an ability to process citations and references.

2.3 Scholarly Document Processing Workshops

Multiple workshops have emerged to meet these challenges. The first was the *1st International Workshop on Mining Scientific Publications* (WOSP) held in 2012 (Knoth, Zdráhal, and Juffinger 2012), which recognized the growing importance of Digital Libraries and the insufficiency of libraries that only provide access to scientific content. The organizers of this workshop sought to connect researchers who were interested in developing systems for analyzing and extracting information from databases of scientific publications and using that information to develop tools for improving research. In 2016 the first workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL) was run, with similar motivations to the WOSP workshops (Cabanac et al. 2016).

Shared tasks such as a few hosted by the International Workshop on Semantic Evaluation (SemEval)[2] and the CL-SciSumm task (Chandrasekaran, Yasunaga, et al. 2019) have also served as venues for encouraging and documenting progress on

[2] https://semeval.github.io/

SDP topics. SemEval-2017 Task 10 (Augenstein et al. 2017), SemEval-2018 Task 7 (Gábor, Davide Buscaldi, et al. 2018), and SemEval-2021 Task 11 (D'Souza, Sören Auer, and Pedersen 2021) have provided datasets and benchmarks for scientific entity and relation extraction, while the 2019 CL-SciSumm task provided an opportunity for participants to approach the problem of scientific document summarization (Chandrasekaran, Yasunaga, et al. 2019).

In 2020, the first *Workshop on Scholarly Document Processing* was held with an emphasis on bringing together researchers across multiple domains in NLP, Machine Learning, and Artificial Intelligence (Chandrasekaran, Feigenblat, et al. 2020). The first *SciNLP: Natural Language Processing and Data Mining for Scientific Text* workshop was held in the same year, focused primarily on the extraction and representation of information from scientific text[3]. Both of these workshops were held a second time in 2021. SciNLP 2021 was focused on understanding scientific text, and included our poster presentation describing our Shared Task Description annotation project (Martin, D'Souza, and Pedersen 2021).

2.4 Scientific Information Extraction Corpora

Along with these workshops, numerous corpora have been created for the purpose of providing training data and benchmarks for the development of scientific information extraction systems. The characteristics of these datasets vary. Some are generated automatically with light manual annotation, while others are purely hand-annotated by domain experts. The number of documents per dataset and the research domains that are included vary as well. Most of the corpora include some kind of schema for annotating entities and relations, although there is some small

[3] https://scinlp.org/

variety in the entity and relation types across datasets.

Early work in corpus development for information extraction was oriented towards biomedical domains. The original GENIA corpus (Kim, Ohta, et al. 2003) was motivated by interest in using NLP techniques for text mining from biology texts, and the subsequent need for domain-specific annotated training datasets. The GENIA corpus provided 2,000 MEDLINE[4] abstracts, annotated by two biology domain experts with 100,000 term annotations. The CRAFT dataset (Bada et al. 2011) was created to provide a gold standard resource to help address the increasing growth rate of biological research. Similarly to the GENIA annotation project, the dataset was annotated by biology domain experts. However, the guidelines followed by the annotators were developed by an expert in knowledge engineering and biomedical ontologies and an expert in computational linguistics. This resources provides 97 full-length articles containing 99,907 annotations of 4,319 unique concepts taken from 9 different biomedical ontologies and terminologies.

In response to previous effort towards biology text mining, Zadeh and Handschuh 2014 developed the ACL RD-TEC corpus to make it easier for computational linguists to do terminology extraction research by removing the need to consult with experts from other domains. The ACL RD-TEC is a dataset for benchmarking the extraction and classification of computational linguistics terminology and comprises 10,922 articles from the ACL anthology reference corpus (Bird et al. 2008a) with 83,845 annotated terms. This resource was updated to provide classifications of identified scientific terms into seven categories (*method, tool, language resource, language resource product, model, measures and measurements,* and *other*) (Zadeh and Schumann 2016), allowing for entity recognition and classification to be a new potential downstream task.

[4] https://www.nlm.nih.gov/medline/index.html

Since then, numerous corpora have been hand-annotated by experts in computational linguistics and NLP domains. The ScienceIE corpus (Augenstein et al. 2017) was developed for the SemEval-2017 Task 10 and contains paragraph-level annotations of keyphrases of types *material, process,* and *task*, and *hyponym-of* and *synonym-of* relations. The data was manually annotated by undergraduate student volunteers in relevant disciplines. The dataset contains paragraphs from 500 papers in computer science, physics, and material science domains taken from ScienceDirect.[5].

The dataset created for SemEval-2018 task 7 (Gábor, Davide Buscaldi, et al. 2018) comprised data from both the ACL-RelAcS (Gábor, Zargayouna, et al. 2016) and ACL RD-TEC 2.0 (Zadeh and Schumann 2016) corpora. The authors used all 300 abstract annotations from the ACL RD-TEC 2.0 dataset, expanded it to 500 by following the authors' annotation guidelines, then used the annotations of the same 500 abstracts from the ACL-RelAcS dataset. Entities and relations are annotated, with relation types including *usage, result, model, part_whole, topic,* and *comparison*.

The SciERC corpus (Luan et al. 2018) contains 500 paper abstracts from 12 artificial intelligence conference proceedings from the Semantic Scholar Corpus (Ammar et al. 2018). A domain expert annotated scientific entities of types *task, method, metric, material, other-sciterm,* and *generic*, relations of types*compare, part-of, conjunction, evaluate-for, feature-of, used-for,* and *hyponym-of*, and coreferences. 12% of the data was annotated by four other domain experts to determine intra-annotator agreement. The kappa coefficient calculated to indicate intra-annotator agreement was 76.9% for entities, 67.8% for relations, and 63.8% for coreferences. In addition to describing the SciERC corpus in (Luan et al. 2018), Luan et al. also built a multi-task system to extract the scientific entities, relations, and coreferences from abstracts, achieving an F1 score of 64.2 for entity recognition, 39.3 for relation extraction, and 48.2 for coref-

[5] https://www.sciencedirect.com/

erence resolution. Lastly, they created a scientific knowledge graph by applying their model trained on the SciERC dataset to 110,000 abstracts in artificial intelligence domains.

The TDMSci corpus (Hou et al. 2021) contains expert annotations of 2,000 sentences taken from 30,000 NLP papers published in the ACL Anthology. The sentences were taken from the full text of the papers, not including abstracts, and contain mentions of the three types of entities annotated: *task, dataset,* and *metric.*

While our work is focused on the creation of gold standard (human annotated) datasets, it is interesting to note that the development of some corpora is aided using automatic annotation techniques such as term classification. The benefit of this approach is the ability to create larger corpora more quickly with less labor from expert annotators. However, automatically annotated datasets can be noisier than hand-crafted ones (such noise can be a benchmarking feature for evaluating system robustness, as seen in SemEval-2018 Task 7 (Gábor, Davide Buscaldi, et al. 2018)).

The ACL-RelAcS corpus (Gábor, Zargayouna, et al. 2016) contains 11,000 annotated abstracts and introductions from the Association for Computational Linguistics (ACL) Anthology Corpus (Bird et al. 2008b). Annotations of scientific concepts and relations were created automatically using terminology extraction techniques and pre-existing ontologies. There are 21 types of relations in the dataset: *antonyms, co-hyponyms, is-a, affects, based-on, char, compare, composed-of, datasource, method-applied, model, phenomenon, problem, propose, study, tag, taskapplied, usedfor, uses-information, yields,* and *wrt.*

Schumann and Alonso 2018 created a labeled word list by automatically generating semantic class labels using logistic regression. They captured over 22,980 scientific terms in NLP domains from 11,000 papers taken from the ACL Anthology Reference Corpus (Bird et al. 2008b). The type labels include *technologies, tools, language* *resources, language resource products, models, measures*, and *other*.

The S2ORC corpus (Lo et al. 2020) contains 91.9 million academic papers with bibliometric metadata and mentions of tables, figures, and citations annotated. Papers from the Semantic Scholar literature corpus Ammar et al. 2018 were preprocessed to extract paper metadata, citations, references, bib entries, figures, tables. Links between bib entries, figures, tables in full text are resolved.

The SciREX corpus (Jain et al. 2020) uses the Papers with Code dataset[6] to learn an annotation scheme using distant supervision. Noisy automatic labeling is applied to 438 machine learning papers, which are then manually annotated to correct mistakes made by the automatic system. Four types of entity mentions are annotated, *method, task, metric,* and *dataset*, as well as salient entities, binary relations, and 4-ary relations.

2.5 The NLP Contribution Graph Shared Task

The SemEval-2021 Task 11 (NLP Contribution Graph) (D'Souza, Sören Auer, and Pedersen 2021) provided the corpus that serves as the main source of inspiration for our annotation project described in Chapter 3. It is similar to our work in that full paper texts and sentence-level annotations were provided, but is more complicated containing three levels of granularity. The NLPContributionsGraph corpus contains the full texts of 442 papers in NLP domains. The papers were annotated by domain experts at the sentence-level, phrase-level, and subject-predicate-object level. Sentences were annotated with whether they described a contribution of the paper. Then, contribution sentences were annotated with the scientific terms and relations that describe the contribution. Lastly, those terms and relations were structured into

[6] https://github.com/paperswithcode/paperswithcode-data

triples and classified into twelve information units: *ablation analysis, approach, baseline, code, dataset, experimental setup, experiments, hyperparameters, model, research problem, results,* and *tasks.*

Seven teams participated in this shared task. All seven solutions for phase 1 of the task (classifying sentences as contributing or non-contributing sentences) used a BERT-based solution of some kind (for an explanation of BERT see Section 2.6). The best performing team, UIUC_BIONLP (Liu, Sarol, and Kilicoglu 2021) enhanced the sentence samples with contextual data, adding section header names and sentence positions as features. We took a similar approach in the classification project described in Chapter 4.

Our team developed a pipeline incorporating a variety of techniques. We used DeBERTA (He et al. 2021) for multi-class sentence classification, then used dependency parsing to extract phrases from the sentences identified as contribution sentences and format into subject-predicate-object triples. Seven teams submitted solutions to the first phase, then eight teams made submissions to phase 2 parts 1 and 2. We ranked fifth for the first competition phase, sixth for phase 2 part 1, and fifth for phase 2 part 2. For more information, please read our system description paper (Martin and Pedersen 2021).

2.6 Transformer Architectures

Transformer architectures have pushed the state-of-the-art for many natural language processing tasks, such as the GLUE and SuperGLUE benchmarks (A. Wang, A. Singh, et al. 2018; A. Wang, Pruksachatkun, et al. 2019). Transformers rely on something called attention. The attention mechanism, first used for computer vision starting in 2014 (Robert-Inacio and Yushchenko 2014), allowed the model to focus on different parts of an image to learn which parts of the image are the most rewarding to focus on.

Similarly, the attention mechanism can be used in language models by allowing the model to focus on different parts of the input sequence. This is powerful, because it allows the model to "remember" salient parts of the sequence selectively. Attention was first used to build transformer networks for language modeling in 2017, when it was applied towards machine translation tasks in *Attention is All You Need* (Vaswani et al. 2017). Prior to the invention of the Transformer model, the state-of-the-art for language processing was recurrent neural network (RNN) models with bidirectional long short term memory (Bi-LSTM) cells. RNNs are not very efficient, struggle with long term memory, and cannot be parallelized because they take each word from the input data sequentially (one at a time). Transformer networks address these issues by computing the entire sequence simultaneously using attention blocks.

The transformer network proposed in 2017 is made up of two parts, corresponding to the encoder and decoder structure of RNNs. The input to the encoder is an entire sequence of words. Each input word is embedded and given a positional encoding which keeps track of the position of each word within the sequence to prevent the input from becoming a bag of words. Then the input is fed into an attention block. This block contains multi-head attention, which takes the input embeddings and computes the attention between every position, replacing the original word embedding with an embedding that combines information from pairs of words. This allows the length of the maximal path between any two words in a sequence to never be greater than 1, which is why Transformer models can handle distant semantic dependencies better than RNNs.

The attention block also contains an add-and-norm layer. A residual connection is added taking the original input to the output of the multi-head attention layer, and then normalized so that the values have a mean of 0 and a variance of 1. This step is important because it helps to minimize the number of steps required to optimize the network using gradient descent, by ensuring that the range of values in each layer does not change too much. The output is fed into a feed-forward network, and then another add-and-norm layer. The final output is a sequence of embeddings with one embedding per position in the sequence, which captures the original word plus all of the information from the other words in the sequence. The decoder uses masked multi-head attention to combine output words with the input words received from the encoder. In masked multi-head attention, some of the values in the sequence have their probabilities nullified. This is used in decoding, because the output values must only consider information from previous outputs, so future outputs must be masked to prevent them from being selected. For a full description of the original Transformer for neural machine translation, see Vaswani et al. 2017.

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2019a) was created by Google based on the encoder half of the original Transformer model. Its motivation is similar to the motivation behind bi-directional recurrent neural networks, which is that it is useful to be able to use not just the previous words in a sequence but also the future words in a sequence to predict a given word. The main idea is that the encoder has to learn so much about the input sentence that it makes sense to use it as a stand-alone model separate from the decoder. The authors of BERT took just the encoder model, expanded it to 12 transformer layers instead of 6, and then trained it on a huge dataset of novels and wikipedia pages. It was trained on two tasks: the masked language model, where the model is given a sentence with one word masked and has to predict what word goes in the blank, and the next sentence prediction task, where the model must determine whether one sentence immediately follows another sentence. BERT developed a rich statistical understanding of human language as a byproduct of learning the training tasks, an understanding which can be fine tuned for other tasks by downloading its pretrained weights and adding a feed forward neural network on top to train it on the domain-specific training data.

Since BERT's invention in 2019, many other pretrained BERT variants have been created. For our experiments we used the original BERT and SciBERT (Beltagy, Lo, and Cohan 2019). SciBERT was trained at the Allen Institute for AI on a large corpus made up of scientific papers rather than the more generic books and wikipedia corpora of the original BERT, so SciBERT is a good choice for information extraction tasks in science domains.

3 Annotation Project

This chapter describes the annotation project we performed to develop our gold standard corpus NLPSharedTasks. The aim of this annotation project was to develop a gold standard corpus of shared task overview papers with annotations of shared task descriptions. A shared task description is a span of text containing information on an NLP or computational linguistics task to be performed by participating systems. This information must describe in brief what is to be done to accomplish the task, and may also contain details on the dataset the task is performed over. First we provide a Data Statement following the guidelines by Bender and Friedman 2018. Then, the overview of our methodology is broken up into three sections describing the corpus selection, annotation process, and final set of guidelines. The chapter concludes with a discussion of our findings, including our inter-annotator agreement (IAA) scores and some dataset statistics.

3.1 Data Statement

Following is the Data Statement for our corpus NLPSharedTasks, version 1.

3.1.1 Curation Rationale

Our corpus contains the full texts of 254 Shared Task Overview papers published in the ACL Anthology between the year 2000 and 2021. The criteria for inclusion are:

- The paper was written by the organizers of a Shared Task

- The paper provides a description of the Shared Task, including details on the dataset the task is performed over, the task to be implemented by participating systems, and an overview of participating systems

- The Shared Task described in the paper was hosted by some research workshop in the domain of computational linguistics or natural language processing (NLP)

These criteria ensure that the papers included in the corpus are likely to contain a Shared Task Description phrase. The ACL Anthology was chosen as the source because it provides a catalog that is easy to browse for qualifying candidates for inclusion. Furthermore, choosing a single anthology to draw from provided some consistency of paper style and organization. The starting year (2000) was chosen because the formatting of papers describing earlier initiatives was too dissimilar.

3.1.2 Language Variety

The papers included in NLPSharedTasks are in English as used in scientific communication in linguistics, computer science, and natural language processing domains.

3.1.3 Speaker Demographic

The demographics of the paper authors are unknown. The speakers are likely researchers and students of computational linguistics and natural language processing.

3.1.4 Annotator Demographic

The annotation was performed by two English-speaking annotators well versed in a broad range of NLP topics. Annotator 1 is a graduate student in computer science with a B.S. in computer science, and annotator 2 is a post doctoral researcher in data science with a PhD in computer science. Both annotators had shared task experience, annotator 1 as a participant and annotator 2 as an organizer of SemEval 2021: NLPContributionsGraph (D'Souza, Sören Auer, and Pedersen 2021). Neither annotator was compensated.

3.1.5 Speech Situation

The papers included in NLPSharedTasks were written between 2000 and 2021 in research settings. The speech included in these papers is written and is assumed to be scripted and edited, as well as peer-reviewed. In the case of multiple authors, it is unknown whether interaction was either synchronous or asynchronous. The intended audience of the papers included in NLPSharedTasks is researchers and practitioners of computational linguistics and natural language processing.

3.1.6 Text Characteristics

The genre of the texts included in NLPSharedTasks can be described as written scientific communication in computational linguistics domains and other fields. As such, scientific vocabulary is used throughout that is specific to these domains and the documents are structured in a formal way. Texts are structured with sections under headers including *Title*, *Abstract*, *Introduction*, *Related Work*, *Task Description*, *Results*, and *Conclusion*, among others.

3.1.7 Corpus Access

NLPSharedTasks corpus is available on GitHub and is licensed under a Creative Commons Attribution 4.0 International License.

3.2 Corpus Selection

The resource we drew from was the annual research workshop SemEval and similar initiatives. These venues host shared tasks that approach a wide variety of semantic problems and provide a rich resource for understanding the state of the art in semantic analysis. We assembled our task description corpus by searching the ACL Anthology for shared task description papers, including all SemEval task description papers from the year 2001 to 2021, all CoNLL[1] shared tasks 2000-2020, and shared tasks from workshops hosted by the AACL[2], ACL, EACL[3], EMNLP[4], and NAACL[5] conferences. The dataset was developed in two stages. The first stage selected only Shared Task Overview papers associated with the SemEval workshop from 2001 to 2020, yielding 165 papers. The second stage required searching the ACL Anthology for Shared Task Overview papers published during other workshops, as well as adding the newly published set of papers from SemEval 2021. The second stage added 89 papers to the dataset. The final dataset contains a total of 254 shared task description papers between the years 2000 and 2021 and encompasses twenty natural language processing research topics (see Figure 3.9).

3.3 Annotation Process

The first annotator extracted one or two task description phrases from every paper in the corpus and generated a set of guidelines for the second annotator to follow. The first annotator created two representative sets of twenty papers each. The first subset was extracted after the first stage of dataset selection and contained only

[1] https://www.conll.org/
[2] https://www.aclweb.org/portal/
[3] https://2021.eacl.org/
[4] https://2021.emnlp.org/
[5] https://naacl.org/

SemEval papers from 2001 to 2020. The second subset was extracted after the second stage of dataset selection and contained papers from the CoNLL workshop, SemEval 2021, and other workshops hosted by the AACL, ACL, EACL, EMNLP, and NAACL conferences. These sets were selected to be representative of the range of annotation difficulty, publication date, and subject matter contained in the corpus. The second annotator annotated these subsets extracting only one phrase from each paper, and intra-annotator agreement was determined using Cohen's kappa coefficient. A strict score and a relaxed score were calculated for each dataset, where the strict score compared the exact phrase spans and the relaxed score compared the sentences in which phrase spans were found. After the first subset was annotated by the second annotator, the guidelines were refined by the first annotator to address ambiguities before releasing the second set to the second annotator.

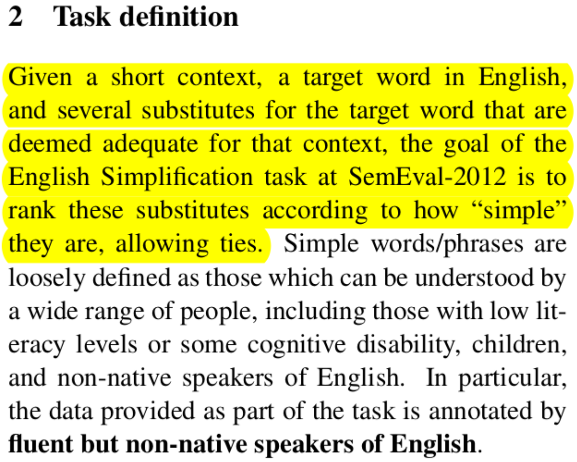
Through the course of writing and revising the guidelines all 254 papers had to be re-annotated four times, yielding a data set that is consistent but time-intensive to produce. Annotator 1 spent roughly 12 minutes per paper during the first round of annotations. Subsequent annotations were less time intensive as they only required evaluating whether the chosen task description still met the requirements rather than reading each paper all the way through.

3.4 Guidelines

This section provides a detailed overview of the final iteration of the guidelines we developed. It covers the definitions of *task description* and task description subtypes *full task description*, *partial task description*, and *multiple subtasks description*. The annotation specifics are described by sets of rules. The first set governs how the task description phrase boundaries should be selected. The second dictates how ambiguities should be resolved, describing four different sources of ambiguities and tips for determining the correct annotation for each scenario.

3.4.1 What is a Task Description?

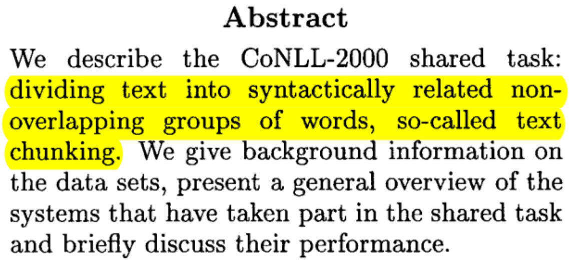
We define a task description phrase as a span of text containing information on the task that must be performed by participating systems. A *full task description* phrase will contain information on the input data and a brief description of what the participating system must accomplish with the input data (see Figure 3.1). There are a variety of ways that paper authors describe the task at hand, which necessitates annotating a broad range of task description types. For example, many papers do not contain a single span that meets the definition of a full task description phrase. For this reason, the annotator may also extract a *partial task description* phrase that only describes the task to be performed by participating systems (see Figure 3.2). However, the extracted task description phrase should make sense out of context; a phrase whose wording is too vague or dependent on previous content to be understood out of context would not be extracted in our annotation process. Task description phrases do not need to provide an exhaustive description of the task, but they should not be overly vague, generic, or confusing. A third type of task description, called *multiple subtasks description* can be extracted when the phrase covers multiple subtasks in a single continuous sequence, as in Figure 3.7. Such a task description is permitted even if the content spans multiple sentences.



2 Task definition

Given a short context, a target word in English, and several substitutes for the target word that are deemed adequate for that context, the goal of the English Simplification task at SemEval-2012 is to rank these substitutes according to how “simple” they are, allowing ties. Simple words/phrases are loosely defined as those which can be understood by a wide range of people, including those with low literacy levels or some cognitive disability, children, and non-native speakers of English. In particular, the data provided as part of the task is annotated by **fluent but non-native speakers of English**.

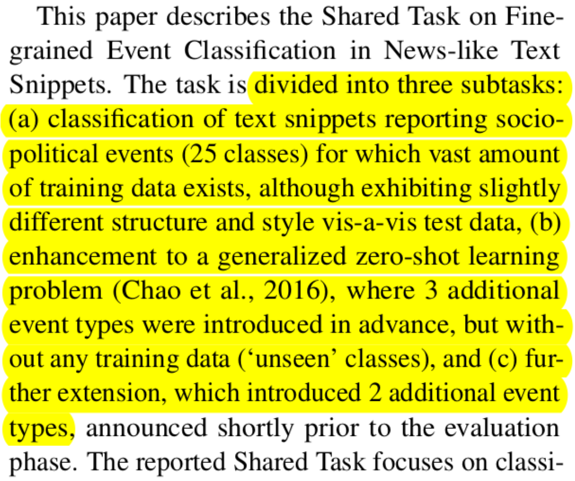
Figure 3.1: A full task description containing a description of the input (”Given a short context, target word in English, and several substitutes for the target word”), and a description of what participating systems must do (”rank these substitutes according to how ”simple” they are, allowing ties”). From *SemEval-2012 Task 1: English Lexical Simplification*, Specia, Jauhar, and Mihalcea 2012.



Abstract

We describe the CoNLL-2000 shared task: dividing text into syntactically related non-overlapping groups of words, so-called text chunking. We give background information on the data sets, present a general overview of the systems that have taken part in the shared task and briefly discuss their performance.

Figure 3.2: A partial task description containing a description of what participating systems must do (”dividing text into syntactically related non-overlapping groups of words, so-called text chunking”). From *Introduction to the CoNLL-2000 Shared Task Chunking*, Tjong Kim Sang and Buchholz 2000.



This paper describes the Shared Task on Fine-grained Event Classification in News-like Text Snippets. The task is divided into three subtasks: (a) classification of text snippets reporting socio-political events (25 classes) for which vast amount of training data exists, although exhibiting slightly different structure and style vis-a-vis test data, (b) enhancement to a generalized zero-shot learning problem (Chao et al., 2016), where 3 additional event types were introduced in advance, but without any training data ('unseen' classes), and (c) further extension, which introduced 2 additional event types, announced shortly prior to the evaluation phase. The reported Shared Task focuses on classi-

Figure 3.3: A task description containing a description of each subtask. From *Fine-grained Event Classification in News-like Text Snippets - Shared Task 2, CASE 2021*, Haneczok et al. 2021.

3.4.2 Task Description Phrase Boundaries

This is a phrase-level annotation task that allows for the extraction of multiple consecutive phrases. This introduces ambiguity in choosing how much of a sentence to include in the extracted phrase. The overall aim is to extract enough of the sentence or consecutive sentences to describe the task so that it makes sense out of context. Edges of the sequence may be trimmed if they are not necessary to understand the task at hand, but internal phrases should not be cut because the extracted sequence must be continuous. The following rules are provided to help the annotator determine how much of a sentence to trim:

1. For compound sentences, only clauses which contain information that specifically describes the task or the given task data should be examined.
2. If the extracted clause is a dependent clause, it might begin with a phrase that describes the relationship between the extracted clause and the trimmed

independent clause, such as phrase beginning with a relative pronoun (e.g. that, which). This phrase should be trimmed if it makes the extracted sequence awkward out of context.

3. When the sentence mentions task participants or participating systems (e.g. ”This task asks participants”, ”Participating systems must”), that mention should usually be removed if it is not relevant to the task description and if it is positioned at either end of the sequence.

4. If the trimmed sequence starts with an infinitive verb, the ”to” should be trimmed.

5. If the last word in the span is followed by a period, then the period should be omitted unless the phrase span covers more than one consecutive sentences.

6. When the phrase is extracted from a title, the part of the title with the year, task number, and workshop or shared task name should be removed, as well as any extraneous information that does not provide information about the task.

Following is an example applying rules 1-5 to determining phrase boundaries for a single sentence, “This paper presents the Graded Word Similarity in Context (GWSC) task which asked participants to predict the effects of context on human perception of similarity in English, Croatian, Slovene and Finnish.”

- **Guideline 1:** The independent clause “This paper presents the Graded Word Similarity in Context (GWSC) task” does not provide any information about the task itself, so it can be removed.

  Result: “which asked participants to predict the effects of context on human perception of similarity in English, Croatian, Slovene and Finnish.”

- **Guidelines 2 and 3:** The phrase “which asked participants” describes the relationship between clauses, so it does not make sense without the context of the full sentence. The fact that the task asked participants to do something is a given, so it need not be included.

  Result: “to predict the effects of context on human perception of similarity in English, Croatian, Slovene and Finnish.”

- **Guideline 4:** The resulting phrase begins with an infinitive verb, so “to” should be trimmed.

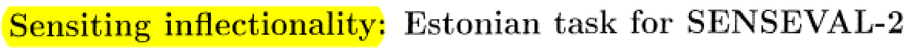
  Result: “predict the effects of context on human perception of similarity in English, Croatian, Slovene and Finnish.”

- **Guideline 5:** The phrase ends with a period, which should be omitted.

  Final Result: “predict the effects of context on human perception of similarity in English, Croatian, Slovene and Finnish”

Rule 6 can be applied to titles such as ”SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering”. “SemEval-2015 Task 4” should be removed. The name of the task “TimeLine:” should also be removed, because it does not provide information on the task. The result is “Cross-Document Event Ordering”.

For downstream phrase-level extraction tasks, the guidelines describing task description phrase boundaries may be encoded into hand-written rules to aid in boundary identification.



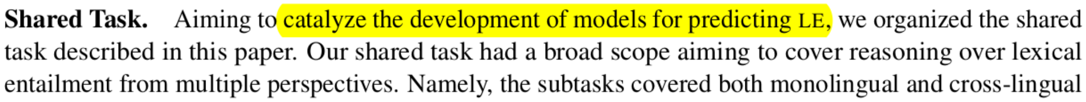
Sensiting inflectionality: Estonian task for SENSEVAL-2

Figure 3.4: This is an example of a phrase that could be misconstrued as a task description, but only describes the topic addressed by the shared task (”sensiting inflectionality”). This phrase should not be extracted as a task description phrase. From *Sensiting Inflectionality: Estonian Task for SENSEVAL-2*, Kahusk, Orav, and Õim 2001.

3.4.3 Ambiguous Cases

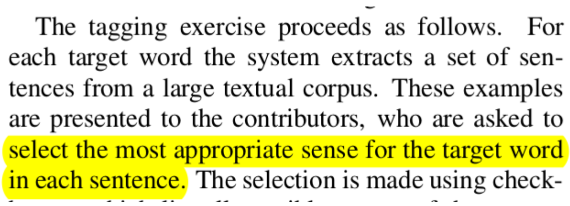
What is Not a Task Description?

Although there is a broad range of specificity and detail that is permitted in task descriptions, we discovered that there are some phrases that may appear similar to task description phrases that should not be extracted. Task description phrases will often mention the research area, but a phrase that only describes the research area is insufficient if it does not contain information on the task to be performed (see Figure 3.4). One other pitfall we observed is the fact that sometimes paper authors use language when describing the aim, goal, or “task” of the task organizers (see Figure 3.5) or dataset annotators (see Figure 3.6) that makes it seem like they are describing the task to be performed by participating systems. A phrase describing the organizers’ aim or the dataset creation task would not be extracted as a task description phrase according to our guidelines. Other misleading phrases include phrases in Introduction/Background sections which describe a general NLP task in order to provide the background information the reader needs to understand the task at hand (see Figure 3.7). Such a phrase would also not be extracted, because it is not describing the actual task to be performed.



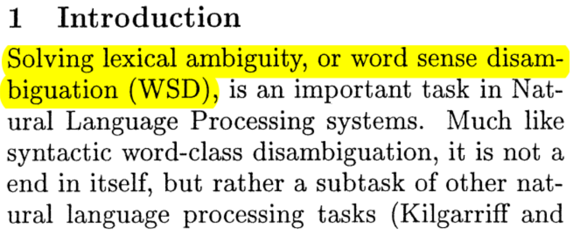
Shared Task. Aiming to catalyze the development of models for predicting LE, we organized the shared task described in this paper. Our shared task had a broad scope aiming to cover reasoning over lexical entailment from multiple perspectives. Namely, the subtasks covered both monolingual and cross-lingual

Figure 3.5: This is an example of a phrase that could be misconstrued as a task description, but actually describes the aim of the task organizers. This phrase should not be extracted as a task description phrase. From *SemEval-2020 Task 2: Predicting Multilingual and Cross-Lingual (Graded) Lexical Entailment*, Glavaš et al. 2020.



The tagging exercise proceeds as follows. For each target word the system extracts a set of sentences from a large textual corpus. These examples are presented to the contributors, who are asked to select the most appropriate sense for the target word in each sentence. The selection is made using check-

Figure 3.6: This is an example of a phrase that could be misconstrued as a task description, but actually describes the task for the annotators who helped prepare the task's dataset. This phrase should not be extracted as a task description phrase. From *An Evaluation Exercise for Romanian Word Sense Disambiguation*, Mihalcea et al. 2004.



1 Introduction

Solving lexical ambiguity, or word sense disambiguation (WSD), is an important task in Natural Language Processing systems. Much like syntactic word-class disambiguation, it is not a end in itself, but rather a subtask of other natural language processing tasks (Kilgarriff and

Figure 3.7: This is an example of a phrase that could be misconstrued as a task description, but actually describes background information related to the research area. This phrase should not be extracted as a task description phrase. From *Dutch Word Sense Disambiguation: Data and Preliminary Results*, Hendrickx and Bosch 2001.

Papers with Subtasks or Joint Tasks

One characteristic of Shared Task Overview papers that complicates the annotation process is that some papers describe subtasks, joint tasks, and multi-track tasks. Developing a machine reader to determine how many subtasks are described in the paper and to extract a task description for each one from potentially disparate parts of the paper would not be trivial. For this reason, we do not annotate subtask descriptions unless they appear in consecutive sequences of text as in Figure 3.3. When there are multiple tracks for a task, the tracks usually differ in terms of the input language or what kinds of resources are allowed in training the participating system. Because tracks usually approach the same task, we ignore them in this annotation scheme. Joint tasks are different from subtasks in that, while subtasks are usually , joint tasks are stand alone tasks without a hierarchy. When a paper describes two or more stand-alone tasks, then the annotator should attempt to extract a task description for each task, even if the sequences of text are not continuous or found in different sections of the paper.

Subtasks, joint tasks, and task tracks can be easily identified based on vocabulary. This would be a good application of handcrafted rules to help a machine reader determine if a paper contains subtasks or not. Determining whether there are subtasks would be helpful, as papers with subtasks require the annotator to read the paper in a slightly different way.

Choosing Between Two Or More Candidates

Sometimes there may be more than one phrase that qualifies as a candidate task description phrase. We identify four types of ambiguities arising from the case of multiple candidates, and describe how to resolve them.

1. The most common relationship between two candidate task descriptions is that one contains more detail than the other. The additional detail may describe the task that systems must perform, or may provide more information on the input data. Or, the content of the phrases is equivalent, but one candidate has more specific word choices. We selected the candidates that contained greater

detail or specificity.

Example: Using detail to choose between two candidates

Phrases:

(a) "automatically assessing humor in edited news headlines"

(b) "build systems for rating a humorous effect that is caused by small changes in text"

Explanation: both phrases describe the task, but the first is more general and the second is more specific. We chose phrase (b).

2. Sometimes there are two candidates that are similar in content but vary in clarity or technical expressiveness. We selected the candidates that were easiest to read out of context.

   Example: Using clarity of language to choose between two candidates

   Phrases:

   (a) "quantify the degree of prototypicality of a target pair by measuring the relational similarity between it and pairs that are given as defining examples of a particular relation"

   (b) "rate word pairs by the degree to which they are prototypical members of a given relation class"

   Explanation: at first glance phrase (a) seems like it must be a better choice because it seems more detailed. However, the fact that phrase (b) contains "word pairs" rather than "target pair" makes it a slightly more clear task description phrase. In this case, the readability of phrase (b) makes it a better option.

3. Sometimes the candidates contain different but complementary information about the task. In this case, if the candidates are found in consecutive sentences, then we extract both of them as a single sequence, including text between their boundaries to preserve continuity.

   Example: Using consecutive sentences

   Phrases:

   (a) "Annotate instances of nouns, verb, and adjectives using WordNet 3.1"

   (b) "Label each instance with one or more senses, weighting each by their applicability"

   Explanation: Both of these phrases together are sufficient to describe the task at hand, but neither is sufficient by itself. Fortunately, they appear in consecutive sentences; we extracted both of these phrases and the text in between.

   Result (extracted sequence in italics:

   This task required participating systems to *annotate instances of nouns, verb, and adjectives using WordNet 3.1 (Fellbaum, 1998), which was selected due to its fine-grained senses. Participants could label each instance with one or more senses, weighting each by their applicability.*

4. At other times, the candidates contain complementary information but are not found in consecutive sentences. In this case, if one of the phrases is coherent and complete enough to stand alone as a task description, then we extract it.

5. Occasionally, a paper will contain multiple candidates that have no significant difference in their content or quality. This happens most often when the paper

authors reuse text from the abstract in the body of the paper. We extracted any of the candidates in this scenario.

Example: choosing between two equivalent candidates

Phrases:

(a) "Given a set of documents and a set of target entities, the task consisted of building a timeline for each entity, by detecting, anchoring in time and ordering the events involving that entity"

(b) "Given a set of documents and a set of target entities, the task consists of building a timeline related to each entity, i.e. detecting, anchoring in time, and ordering the events in which the target entity is involved"

Explanation: both phrases are equally good candidates, so either may be extracted.

Resolving ambiguities requires nuance. It may not be possible to come up with detailed enough rules to choose between two good task description candidates. This is a problem that would be served better with a machine learning approach.

The Null Case

Rarely, a paper does not contain a phrase that sufficiently describes the task as a whole. This occurs in the following scenarios:

1. The paper only provides description phrases of subtasks, without a description of the task as a whole.

2. The task is an iteration of a task from a previous year, and the description paper does not contain a suitable description phrase, being highly referential to the previous year's task description.

3. There is a phrase that describes the task, but its language is too indeterminate to serve as a stand alone task description. An example of such a phrase is *"unify the somewhat divergent research efforts and to address certain recognized data issues that have developed in the nascent phase of this subfield"*; this phrase should not be extracted as the task description phrase.

In any situation where a task description phrase cannot be found, we used a portion of the title of the paper if the title contained a phrase describing the task. If no task description phrase could be found in the body of the paper and the title did not describe the task, then we would use NULL as the task description.

3.5 Annotation Results

We calculated the inter-annotator agreement between annotator 1 and annotator 2 using Cohen's kappa statistic (Cohen 1960). Inter-annotator agreement is a measure that describes how well two or more annotators are able to apply the same label to each sample in a dataset. Cohen's kappa coefficient is a statistic that measures agreement in such a way that any agreement occurring by chance is eliminated. This is performed by first calculating the proportion of samples on which the annotators agreed (p_o) and the proportion of units where chance agreement is expected (p_e). Then κ is calculated by scaling the difference between p_o and p_e by $1 - p_e$ (see Equation 3.5) (Cohen 1960).

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{3.1}$$

k can range between -1 and 1, where $\kappa \leq 0$ indicates indicates no agreement and $\kappa = 1$ indicates perfect agreement. The closer to 1, the more reliable the inter-annotator agreement is: $0.01 \leq \kappa \leq 0.20$ indicates slight agreement; $0.21 \leq \kappa \leq 0.40$

fair agreement; $0.41 \leq \kappa \leq 0.60$ moderate agreement; $0.61 \leq \kappa \leq 0.8$ substantial; and $\kappa \geq 0.81$ indicates near perfect agreement (McHugh 2012).

Cohen's kappa is a useful measure because it corrects for how frequently two annotators might agree by chance. For example, if two annotators were to randomly assign one of two labels to a set of samples, there is a 25% chance that the annotators will agree for any given sample. This does not mean that the annotators are *reliably* agreeing 25% of the time, so a score that is reduced by the amount of chance agreement would be more meaningful (in this case their score would be zero).

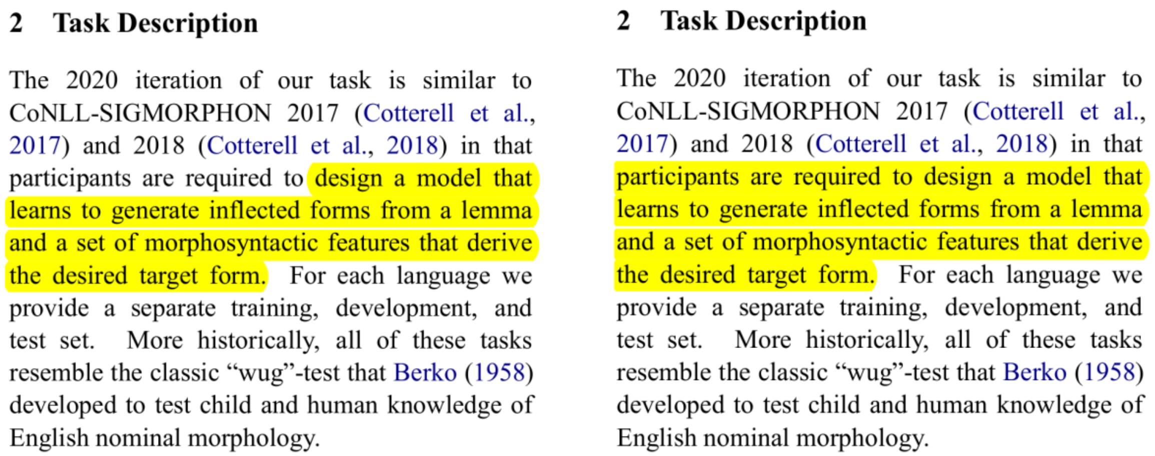
Measuring inter-annotator agreement for our dataset is important because it helps us understand how consistent our guidelines are and how consistently the annotator follows them. It is also important for downstream machine learning tasks that might be performed using our corpus (like the task we performed in Chapter 4). Understanding how well a human can predict an annotation for a given dataset helps to evaluate how well a machine performs the same task.

Two subsets of 20 papers each were selected to determine inter-annotation agreement in two stages. The first annotator could choose multiple candidate phrases that matched the criteria and were equivalent in quality. The second annotator chose only one phrase per paper. A strict score and a relaxed score were calculated for each set. The strict score was calculated by finding Cohen's kappa coefficient of the second annotator's set and the first annotator's set, where the choice that best matched the other annotator's choice was selected. Two phrases only counted as a match if they were the same. The relaxed score was calculated by counting phrases as a match if they overlapped.

Set #	Strict Score	Relaxed Score
Set 1	.383	.6401
Set 2	.4374	.9488

Table 3.1: Inter-annotator agreement measured with Cohen's kappa coefficient

The strict Cohen's kappa coefficient for the first subset was 0.383, and the relaxed Cohen's kappa coefficient was 0.6401, indicating fair to substantial agreement (Viera, Garrett, et al. 2005). After we made revisions and clarifications to the annotation guidelines, we annotated the second subset, and achieved a strict score of 0.4373 and a relaxed score of 0.9488, indicating moderate to almost perfect agreement. The difference between the strict and relaxed scores indicates that, though the annotators often selected the same sentence to extract the task description phrase from, mutually choosing equivalent phrase spans is somewhat difficult. For example, from the paragraph in Figure 3.8, annotator 1 extracted "design a model that learns to generate inflected forms from a lemma and a set of morphosyntactic features that derive the desired target form", and annotator 2 extracted "*participants are required to* design a model that learns to generate inflected forms from a lemma and a set of morphosyntactic features that derive the desired target form".



2 Task Description

The 2020 iteration of our task is similar to CoNLL-SIGMORPHON 2017 (Cotterell et al., 2017) and 2018 (Cotterell et al., 2018) in that participants are required to design a model that learns to generate inflected forms from a lemma and a set of morphosyntactic features that derive the desired target form. For each language we provide a separate training, development, and test set. More historically, all of these tasks resemble the classic "wug"-test that Berko (1958) developed to test child and human knowledge of English nominal morphology.

2 Task Description

The 2020 iteration of our task is similar to CoNLL-SIGMORPHON 2017 (Cotterell et al., 2017) and 2018 (Cotterell et al., 2018) in that participants are required to design a model that learns to generate inflected forms from a lemma and a set of morphosyntactic features that derive the desired target form. For each language we provide a separate training, development, and test set. More historically, all of these tasks resemble the classic "wug"-test that Berko (1958) developed to test child and human knowledge of English nominal morphology.

Figure 3.8: The left example shows annotator 1's choice and the right example shows annotator 2's choice. From *SIGMORPHON 2020 Shared Task 0: Typologically Diverse Morphological Inflection*, Vylomova et al. 2020

3.6 Corpus Statistics

In this section we provide quantitative information on our corpus. One benefit of annotating shared task overview papers published over a long period of time is that this resource could potentially be used to study NLP research progress and trends. For this reason, we provide some basic statistics on the content of the papers included (see Section 3.6.1). We also provide statistics on the extracted task descriptions in Section 3.6.2, as such information may be useful to others for building task description extraction systems.

3.6.1 Characteristics of Shared Task Overview Papers

The 254 shared task overview papers collected for this dataset encompass a wide variety of research topics. We identified 20 distinct topics (see Figure 3.9), although there is some overlap. For example, note that the topic **classification** appears to contain 5 papers. There are more classification tasks found in the dataset, but they fell into other categories such as **sentiment analysis**, **social factors**, and **time and** space.

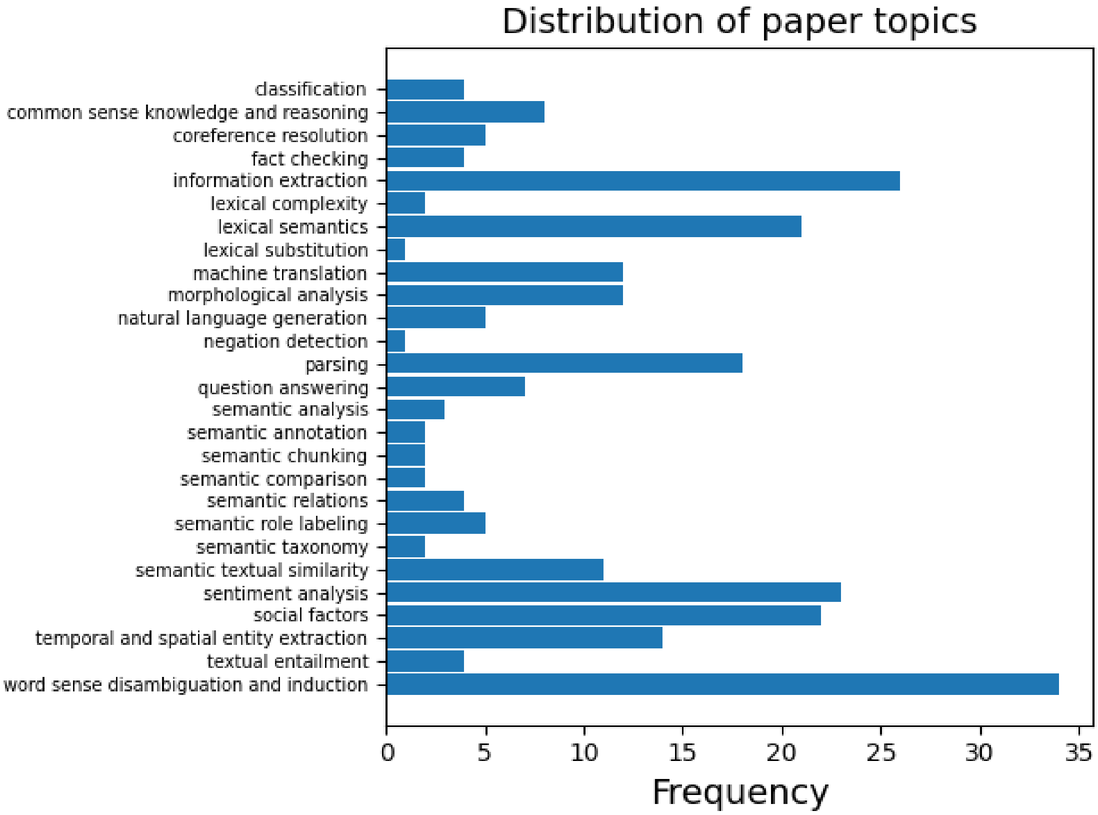


Figure 3.9: Distribution of Paper Topics

The distribution of papers over year of publication can be seen in Figure 3.10. Note that year 2001, 2004, 2007, and 2010 appear to be outliers. This is because the majority of our dataset was taken from the SemEval workshops, which were not held in 2000, 2002, 2003, 2005, 2008, 2009, or 2011.

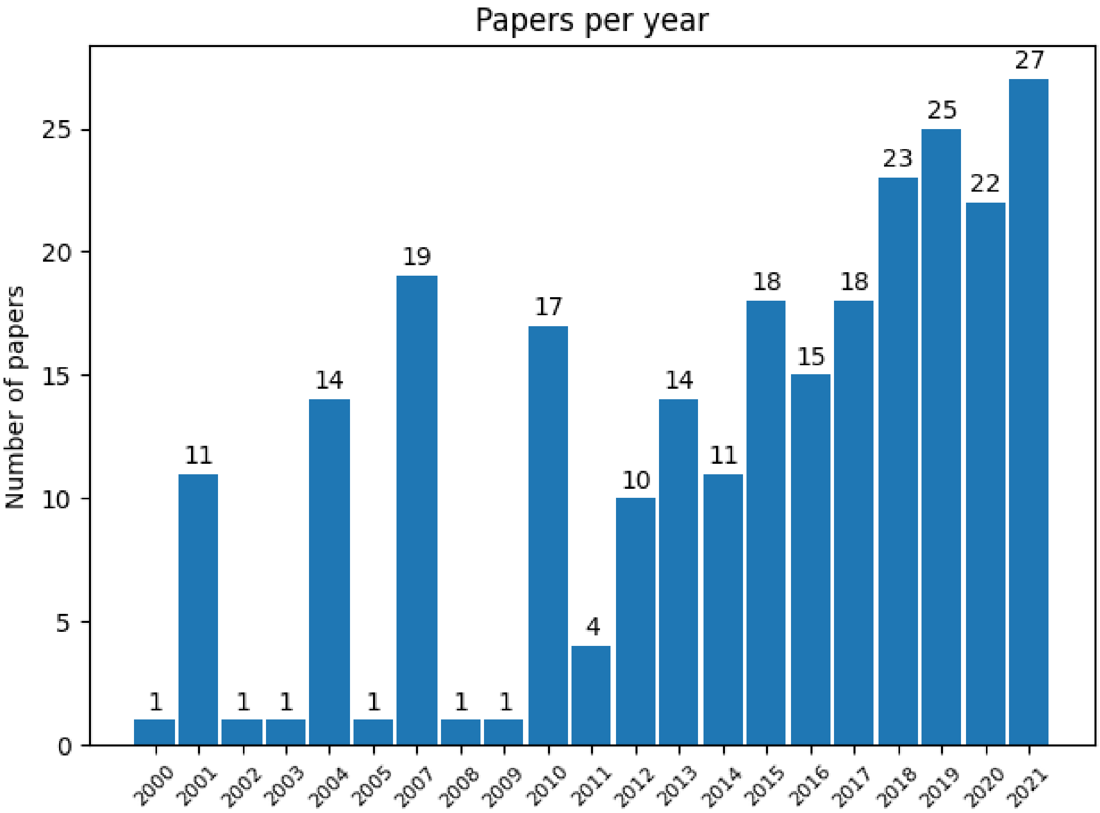


Figure 3.10: Distribution of Publication Year

An interesting characteristic of these shared tasks is that not all tasks are novel. It is fairly common for tasks to be re-run for several years. This allows participants to improve benchmarks by building on previous work. It also allows task organizers to add to the complexity of the task.

Reruns	rerun	novel task
# of papers	65	189

Table 3.2: Distribution of reruns

3.6.2 Task Description Characteristics

One of the most consistent patterns observed is that task descriptions tend to appear under the same limited set of section headers (Figure 3.11). While they are most commonly found in the abstract, they also frequently appear in introduction sections. Unsurprisingly, sections with titles such as ”Task Description” or ”Task Overview” often contain task description phrases suitable for our project. Rarely, papers may not contain a good task description until the conclusion or discussion section. And there were thirteen papers that did not contain a task description in the body, but had a title that was sufficient.

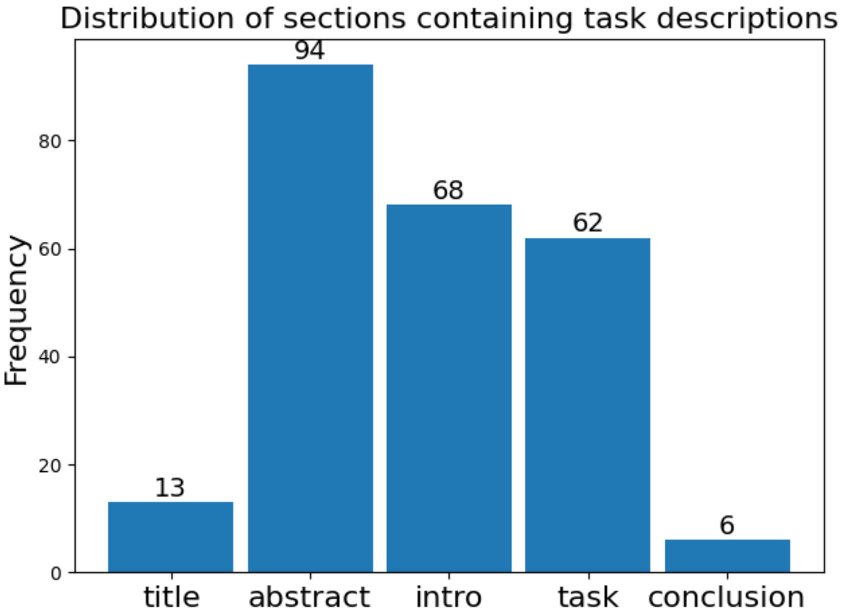


Figure 3.11: Distribution of sections containing task descriptions

Section 3.4.3 described a number of factors that caused ambiguity during the annotation process. One of these factors is the scenario where there are two or more candidate task descriptions that are all good choices. These ambiguities could be resolved in four ways: by a.) choosing the one with more task-related **detail**; b.)

choosing the phrase with better **readability**; c.) choosing the phrase that works best on its own when they contain complementary but **different information**; and d.) choosing any candidate when the phrases are truly **equivalent**. The frequencies of each of these choices can be seen in Figure 3.12.

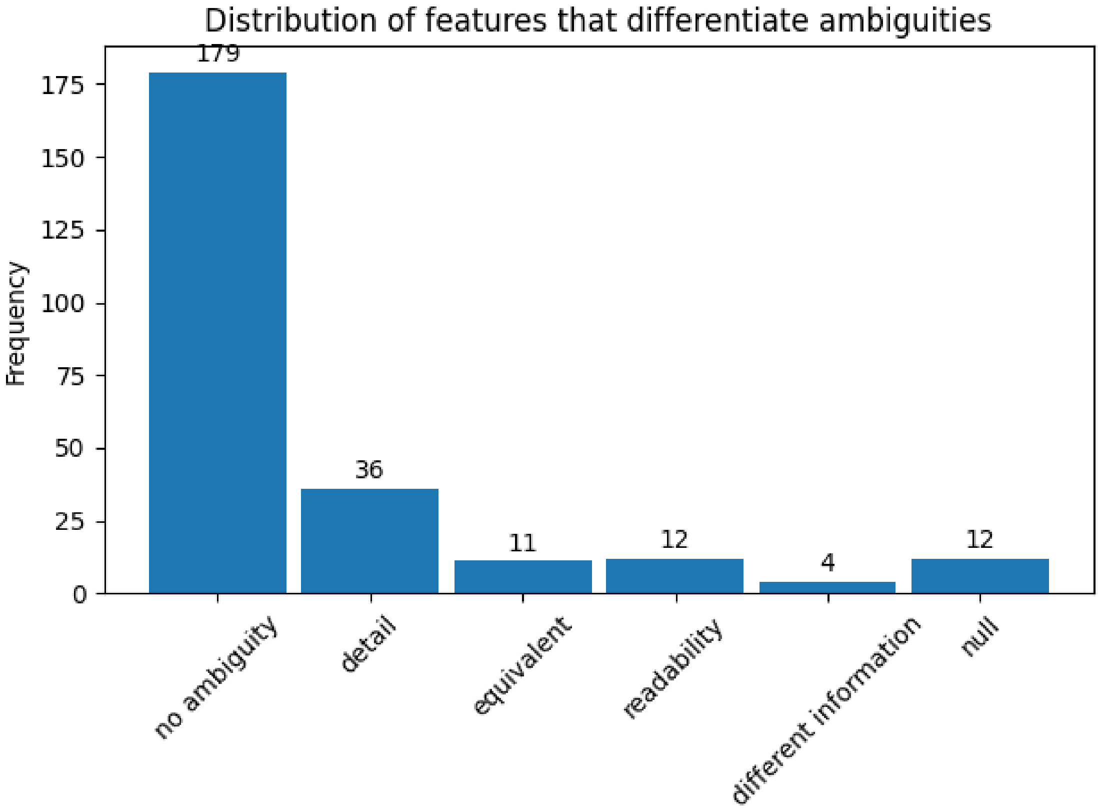


Figure 3.12: Distribution of features that help choose between two or more candidate phrases

In Section 3.4.1 we describe multiple types of task descriptions: full, partial, and multiple subtask descriptions. The counts for each of these types are found in Table 3.3.

phrase type	full	partial	subtask	null
# of extracted phrases	127	104	13	12

Table 3.3: Number of full, partial, subtask, and null task descriptions in 254 shared task overview papers

A complicating aspect of this corpus in terms of information extraction and text classification is the varying lengths of task descriptions. This low-homogeneity can make it more difficult to train traditional classifiers, but is important because it provides a more ”real-world” environment. The extracted phrases span between 1 and 4 sentences, and contain between 3 and 126 tokens (Table 3.4).

feature	mean$^{+}_{-}$ std	max	min
word count	29$^{+}_{-}$21.8	126	3
sentences in span	1.17$^{+}_{-}$.48	4	1

Table 3.4: Mean word count and sentences per task description

4 Classification Project

Sentence classification is an NLP task where, given a set of pre-labeled sentences and a set of classes, a model must assign a class to each sentence without knowing its true label. In binary classification tasks, there are two labels, often described as *positive* and *negative* to indicate that the data can be classified as either X (positive) or not-X (negative). Following testing, each output sample can be described with one of four labels: true positive (TP) if the sample was correctly classified as positive; true negative (TN) if the sample was correctly classified as negative; false positive (FP) if the sample was wrongly classified as positive; and false negative (FN) if the sample was wrongly classified was negative. To explore its utility as a resource for training machine readers, we used our corpus as data for a sentence classification task. This task requires two classes: task description (positive) and non task description (negative).

Despite the fact that task descriptions are defined as sequences that can be longer or shorter than a single sentence, we designed a sentence classification task because we achieved much higher inter-annotator agreement scores when we compared the chosen sentences rather than phrases (see Section 3.5). Classifying sentences is an easier task because we do not have to be concerned with phrase boundaries. Once a machine learner is trained on the training set, it can be incorporated into a machine reader to help select the task description from an unseen shared task overview paper.

4.1 Data Preparation

Scholarly papers are often stored as PDFs, which are not very machine-actionable[1]. For this reason the full text for each paper had to be extracted and stored in a different format. We used GROBID (*GROBID* 2008–2022), a software library for extracting and restructuring scientific papers from PDFs into XML encoded files. After converting NLPTasks into XML documents, we wrote a custom XML parser to extract the text data into plain text files. Headers were captured and entered on a single line. We used an empty newline to separate sections and subsections. Each paragraph is printed on a single line. While we could have left the files in XML format, there is benefit in creating a dataset that is easily human readable. For an example of what it looked like to transform a PDF to a plain text file see Figure 4.1.

One issue we had with GROBID was that it does not handle tables well. The text in table cells gets inserted into the body of the paper wherever the table was inserted, resulting in gibberish often found in the middle of a sentence. We had to manually fix this issue by cleaning the gibberish from task descriptions. We did not clean non task description sentences due to time constraints. It was particularly important to clean the task description sentences because there are so few of them that we did not want to lose any samples.

To prepare our corpus for a sentence classification task, we randomly divided the 254 papers into a training set of 228 papers and a test set of 26 papers. This is a split of roughly 90% training data and 10% test data. We allotted only 10% of the data for testing because we wanted to be able to train and validate our models on as much data as possible due to the relatively small number of task description sentences. The reason that we split the data by paper instead of by sentences is that we wanted to be

[1] Some journals and archives such as arXiv (`https://arxiv.org/`) provide LaTeX source code for papers in addition to PDFs

able to preserve the idea of extracting a single span of text per paper that represents the task description for the shared task.

Each set is stored in a .csv file where each row contains a single sentence and label (1 if task description, 0 if not), as well as a paper_id that associates the sentence with the paper it is taken from. The sentences are stored in the order they appear in each paper. The resulting training set contains 259 positive samples and 41,493 negative samples, and test set contains 34 positive and 4725 negative samples. This is an extremely unbalanced dataset, where less than 1% (0.63%) of the total sentences are positive samples. The reason for this is the annotation goal was to extract a single candidate per paper. However, extra steps must be taken to change the balance enough so that machine classifiers are able to learn how to identify task descriptions.

Datasets designed for classification are unbalanced when there is a large difference in size between the classes. This poses problems for machine learners. When there are many training samples for one class for the machine to learn from, it will be able to detect new members of that class with much greater confidence than for the smaller class. Furthermore, salient characteristics of the small class might be present frequently enough in the large class that they lose meaning for the classifier. The balance must be adjusted prior to classification by either increasing the number of samples for the small class or decreasing the number of samples for the big class (or both). Additionally, metadata from the papers can be added as features to provide relevant context.

We applied a few techniques for increasing the size of our positive sample set and decreasing the size of our negative sample set. The approach that yielded the best results was to use the hierarchical structure of academic documents to annotate our positive samples with contextual metadata and to make informed choices about which negative samples should be removed. Section 4.1.1 describes this approach in detail.

Other less effective experiments can be found in Appendix A.

4.1.1 Leveraging Paper Context and Hierarchical Structure

Incorporating Positional Data

Scholarly papers tend to have a predictable structure. Task description overview papers always start with an abstract and introduction, which tend to be followed by task description and dataset preparation sections, before describing the system solutions and reporting results. There are patterns within sections as well; for example, introductions that contain a task description phrase often contain the phrase near the end of the section. For this reason, we added positional data as features to the dataset following the example of Liu, Sarol, and Kilicoglu 2021.

We added a section header feature to the dataset by iterating through the plain text files and capturing the header for each section. Each sentence's position was quantified with four values: the sentence index relative to its section, the sentence index relative to the full paper, the quadrant of the section that the sentence is found in, and the quadrant of the paper that the sentence is found in. These five features were added as columns to the training and test .csv files (see Table 4.1 for an example of what a data instance looks like).

sentence id	paper id	header	local pos	global pos	local pct	global pct	sentence	label
5274	S10-12	abstract	2	3	3	1	The task involves recognizing textual entailments based on syntactic information alone.	1

Table 4.1: Example of a data sample

Eliminating Sections

In addition to annotating the training and test data with positional information, we made the decision to eliminate any section that does not contain a task description for each paper. For example, a paper with the task description in the introduction would be represented in the dataset only by the sentences found in the introduction. This approach has two benefits. First, it improved the ratio of positive to negative training and test samples. Second, it addressed the following problem. Because the goal was to extract a single sequence from each paper, some papers have negative samples that would actually qualify as task descriptions if a better candidate had not been found. It was detrimental for the classifiers to be learning negative samples that meet the criteria for task descriptions. Eliminating sections allowed us to remove some of those perplexing sentences. The resulting training set contains 259 positive samples and 2,304 negative samples, and the resulting test set contains 34 positive samples and 293 negative samples. After reducing the dataset, 11.28% of the total data is positive which, while still unbalanced, is more manageable than the previous 0.64%.

One problem with manually removing samples from the dataset based on which sections contain task descriptions and which do not is that the test set had to be examined. This is a problem because the reduced test set is less ”real world”; in a non-experimental setting, the machine reader should be able to extract a task description from a whole paper, since it does not know ahead of time which section contains the task description. To address this issue, we tested our model on three versions of the test set. The first was manually reduced the same way as the training set. The second had sections automatically removed by training a BERT model on section headers seen in the training set. This model was then applied to the test set to classify section headers as either likely or unlikely to contain a task description. This is a more fair test set because one could apply this classifier to any unseen papers to filter out paper sections. The third set is the full test set without any data removed.

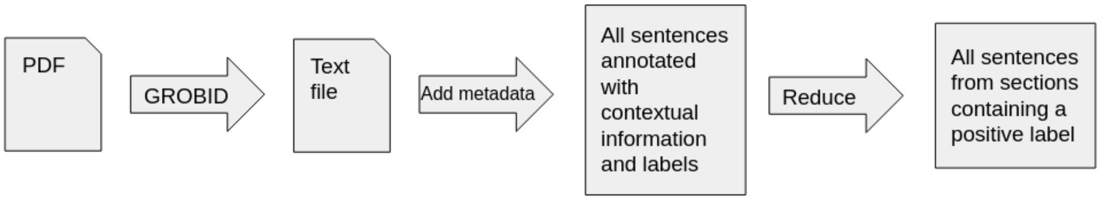

PDF
GROBID
Text file
Add metadata
All sentences annotated with contextual information and labels
Reduce
All sentences from sections containing a positive label

Figure 4.2: Dataset extraction process

4.2 Experimental Setup

We ran an exhaustive set of experiments on the training data to compare different classification algorithms, text encodings, and hyperparameter settings. Each combination of classifier, encoding, and hyperparameters was run ten times, with a different subsection of the dataset set aside for validation data each time. The average of each set of ten runs was calculated and reported.

4.2.1 Preprocessing

Natural language must be encoded into some kind of numeric representation before it is input into a machine learning model, as vectors of numbers can be more easily operated on. There is an additional benefit to encoding text in that the encoding can actually store more information about each sentence, word, or character's context than the raw strings can. Information such as the distribution of a word and its proximity to other words can be captured in vectors or learned weights called word embeddings, and documents can be represented by matrices of vectors or embeddings[2]. Following is a brief description of the encodings used in this work.

Count Vectorization, which is similar to one-hot encoding, can be used to represent words and documents. A span of text is represented by creating a matrix where each row represents a unique word in a predetermined vocabulary. The element associated with each word is set to be the literal count of that word in the sequence, and words that do not appear in the sequence are set to zero. This approach encodes only the raw frequencies of words.

Tf-idf (term frequency-inverse document frequency) is a statistical measure than can be used to weight each token in the matrix according to how frequent the token is in a given sequence, and how many sequences in the corpus contain the token (Manning, Raghavan, and Schütze 2008). The tf-idf score of a token t in a document d is calculated by multiplying the frequency of t in d by the inverse frequency of documents that contain t. Both factors can be normalized by using the log of each term, resulting in Equation 4.2.1 (where N is the total number of documents in the corpus).

$$tfidf(t,d) = \log_{10} freq(t,d) \times \log_{10} \frac{N}{freq(d,t)} \tag{4.1}$$

The effect of this encoding is that tokens that are very frequent in one document but relatively infrequent in others will receive a higher score, indicating their importance to the documents they appear in.

In our experiments we used three variations on the tf-idf encoding method. For each version, token t represents a different unit: words, character-level ngrams, and word-level ngrams. An ngram is a sequence of n consecutive bits of text[3]. So word-level 2-grams in the phrase ”consecutive bits of text” would be ”consecutive bits”,

[2] See chapter 6 of Speech and Language Processing for more information on word embeddings: https://web.stanford.edu/~jurafsky/slp3/6.pdf

[3] For more on ngrams see https://web.stanford.edu/~jurafsky/slp3/3.pdf

”bits of”, and ”of text”. Character-level 2-grams of the word ”text” would be ”te”, ”ex”, and ”xt”. Our experiments used 2- and 3-grams.

Word embeddings are similar to the vector representations described above, but are shorter and contain more nuanced information. Word embeddings are created by training a model on some natural language processing task, and saving the the weights for each learned word in a dense vector.

For our experiments we generated count vectorized, character-level tf-idf, and word-level tf-idf encodings, and pretrained word embeddings for the dataset using fasttext (Bojanowski et al. 2017) and BERT (Devlin et al. 2019b) embeddings.

4.2.2 Classification Algorithms

We trained a number of non-neural and neural classification algorithms and fine-tuned two BERT variants on the dataset. Each classifier was tested on at least one type of encoding. For a summary of the hyperparameter settings and encodings used for each classifier, see Table 4.2.

4.2.3 Training Loop

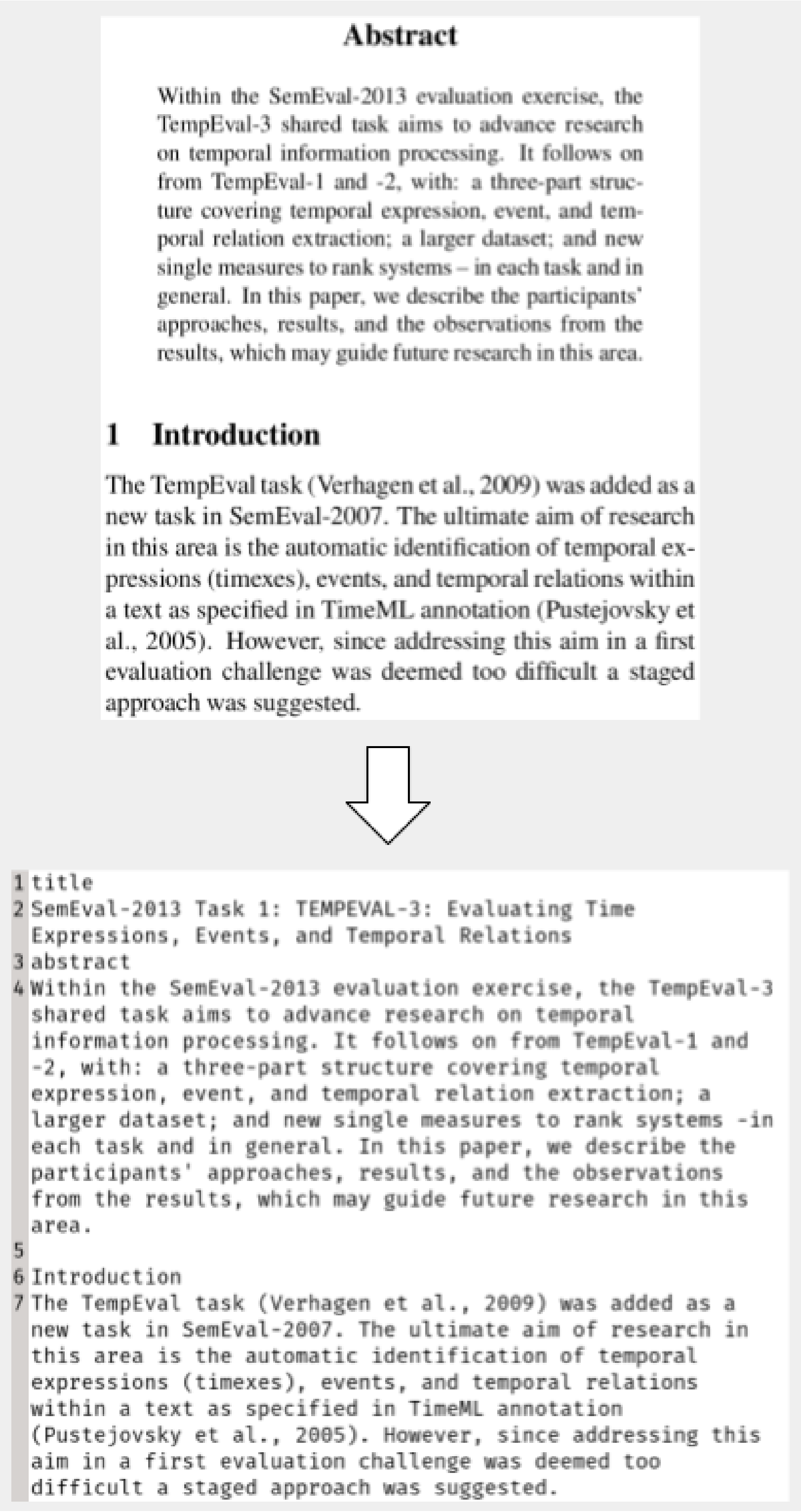
For the non-BERT algorithms each classifier-encoding pairing learned the training dataset ten times. In between runs, the data was shuffled and a new validation set containing 10% of the training data was selected. The precision, recall, and F1 score was recorded for each training run. Then the mean scores and standard deviation were calculated for each classifier-encoding pair. These scores were saved in a text file with automatically generated LaTeX score so that they might be easily reported in this document. The procedure for the two BERT models was similar, but instead of training different classifier-encoding pairings, we trained for every possible hyperparameter combination. The procedures described in this paragraph were performed twice, once on the dataset annotated with contextual data and once on just the sentences and their labels. The procedure for training the non-BERT algorithms can be seen in Algorithm 1 and the BERT experimental procedure can be seen in Algorithm 2.

Algorithm 1 ML experiments

```
procedure TRAIN_LOOP(training_data, classifiers)
    encodings ← [count_vectors, tf_idf, embeddings]
    total_results ← empty_list

    for each classifier in classifiers do

        for each encoding in encodings do
            sub_results ← empty_list

            for i in range[0, 10) do
                training_data ← shuffle(training_data)
                training, validation ← split_data(training_data)
                train_model
            precision, recall, f1 ← validate_model
            sub_results.append([precision, recall f1])
        sub_results ← mean(sub_results)
        total_results.append(sub_results)
return total_results
```

Algorithm 2 BERT experiments

```
procedure TRAIN_LOOP(training_data)
    bert_models ← [bert, scibert]
    batch_sizes ← [16, 32]
    learning_rates ← [2e-5, 3e-5, 5e-5]
    epochs ← [2, 3, 4]
    total_results ← empty_list

    for each bert in bert_models do

        for each batch_size in batch_sizes do

            for each learning_rate in learning_rates do

                for each num_epochs in epochs do
                    sub_results ← empty_list

                    for i in range[0, 10) do
                        training_data ← shuffle(training_data)
                        training, validation ← split_data(training_data)

                        for e in range[0, num_epochs) do
                            train_model
                        precision, recall, f1 ← validate_model
                        sub_results.append([precision, recall f1])
                    sub_results ← mean(sub_results)
                    total_results.append(sub_results)
return total_results
```



Abstract

Within the SemEval-2013 evaluation exercise, the TempEval-3 shared task aims to advance research on temporal information processing. It follows on from TempEval-1 and -2, with: a three-part structure covering temporal expression, event, and temporal relation extraction; a larger dataset; and new single measures to rank systems – in each task and in general. In this paper, we describe the participants' approaches, results, and the observations from the results, which may guide future research in this area.

1 Introduction

The TempEval task (Verhagen et al., 2009) was added as a new task in SemEval-2007. The ultimate aim of research in this area is the automatic identification of temporal expressions (timexes), events, and temporal relations within a text as specified in TimeML annotation (Pustejovsky et al., 2005). However, since addressing this aim in a first evaluation challenge was deemed too difficult a staged approach was suggested.

```
title
SemEval-2013 Task 1: TEMPEVAL-3: Evaluating Time
Expressions, Events, and Temporal Relations
abstract
Within the SemEval-2013 evaluation exercise, the TempEval-3
shared task aims to advance research on temporal
information processing. It follows on from TempEval-1 and
-2, with: a three-part structure covering temporal
expression, event, and temporal relation extraction; a
larger dataset; and new single measures to rank systems -in
each task and in general. In this paper, we describe the
participants' approaches, results, and the observations
from the results, which may guide future research in this
area.

Introduction
The TempEval task (Verhagen et al., 2009) was added as a
new task in SemEval-2007. The ultimate aim of research in
this area is the automatic identification of temporal
expressions (timexes), events, and temporal relations
within a text as specified in TimeML annotation
(Pustejovsky et al., 2005). However, since addressing this
aim in a first evaluation challenge was deemed too
difficult a staged approach was suggested.
```

Figure 4.1: An illustration of the conversion of scholarly PDFs to plain text files using GROBID

Classifier	Hyperparameters	Encodings
Basic Linear Models		
Complement Naive Bayes	Default	count vectors, tf-idf
Logistic Regression	maximum iterations = 1000	
SVM	Default	
KNN	number of neighbors = 20, weight function = distance	
Stochastic Gradient Descent	loss function = modified huber, max iterations = 1000, early stopping	count vectors, tf-idf, fasttext embeddings
Ensemble Models		
Random Forest	Default	count vectors, tf-idf
Gradient Boosting	number of boosting stages = 1000, early stopping	count vectors, tf-idf, fasttext embeddings
XGBoost	number of gradient boosted trees = 1000	
Neural Networks		
Neural Network	layers = 2, activation = sigmoid	fasttext embeddings
Deep Neural Network	layers = 3, activation = relu, sigmoid	
RNN	layers = 4, activation = sigmoid	
RNN with LSTM	layers = 3, activation = sigmoid, dropout = 0.2	
RNN with BiLSTM	layers = 3, activation = sigmoid, dropout = 0.2	
CNN	layers = 10, activation = relu, sigmoid, dropout = 0.2, pool size = 4	
Pretrained Models		
BERT	epochs = 2, 3, 4, batch size = 16, 32, learning rate = 2e-5, 3e-5, 5e-5, optimizer = AdamW	BERT embeddings
SciBERT	epochs = 2, 3, 4, batch size = 16, 32, learning rate = 2e-5, 3e-5, 5e-5, optimizer = AdamW	SciBERT embeddings

Table 4.2: This table contains a list of all classifiers used in our training experiments, their hyperparameter settings, and the software used to implement them.

5 Results

We evaluated our system using precision, recall, and F1 score. These three metrics are calculated using three counts: the number of positive samples correctly classified as positive (true positives, or tp); the number of negative samples incorrectly classified as positive (false positives, or fp); and the number of positive samples incorrectly classified as negative (false negatives, or fn). The precision score shows what proportion of predicted task descriptions were correct and is calculated by dividing the number of true positives by the total number of predicted positives:

$$\text{Precision} = \frac{tp}{tp + fp} \tag{5.1}$$

The recall score shows what proportion of true task descriptions were correctly identified and is calculated by dividing the number of true positives by the number of actual positives:

$$\text{Recall} = \frac{tp}{tp + fn} \tag{5.2}$$

Precision measures how close a classifier gets to only identifying positive samples, and Recall measures how close a classifier gets to identifying all of the positive samples. An ideal measure for evaluating classification results would be able to take both of these concepts into account. So, the F1 score is calculated by finding the harmonic mean of the Precision and Recall:

$$F1 = 2 \times \left(\frac{(\text{Precision})(\text{Recall})}{\text{Precision} + \text{Recall}} \right) \tag{5.3}$$

Confusion matrices are provided for selected experiments. These serve as lookup tables for the number of true positives, true negatives (tn), false positives, and false negatives. Each row of the confusion matrix represents the number of actual samples for a class and each column represents the number of predicted samples for a class. Because our task is a binary classification problem there are only two classes, positive (task description) and negative (not a task description). The sum of both values in row 1 is equal to the total actual positives, and the sum of both values in row 2 is equal to the total actual negatives. Similarly, the sum of both values in column 1 is equal to the total number of predicted positives, while the sum of both values in column 2 is equal to the total number of predicted negatives. The sum of all four cells is equal to the total number of validation samples. See Table 5.1 for an example of a binary confusion matrix.

	Predicted		
True	TP	FN	Positive
	FP	TN	Negative
	Positive	Negative	

Table 5.1: Example confusion matrix

Each classifier was evaluated by calculating the F1 score for each run and finding the mean F1 score across ten runs. For all classifiers, two versions of the data were used. First each classifier was trained on sentences and positional features. Then each classifier was trained on just the sentences, to explore whether the inclusion of contextual information was beneficial.

For the non-BERT classifiers, results were recorded for each combination of classifier and encoding (except for the neural networks, which only used embeddings). For the BERT-based classifiers, results were recorded for each combination of three hyperparameters: the number of epochs, the batch size, and the learning rate. Results were recorded for every hyperparameter combination for each classifier. Each combination was run ten times, and the mean and standard deviation of ten runs were recorded.

The full training results can be found in Appendix C. In Section 5.2, for the non-BERT classifiers results are reported only for the encoding that yielded the highest F1 score. For the BERT classifiers, results are reported for the best hyperparameter combination for each model. In Section 5.2, results are reported for both versions of the dataset.

Because the training results are averaged over ten runs, the confusion matrices also had to be averaged over ten runs. Due to rounding, calculating the precision, recall, and F1 score from the confusion matrices sometimes results in a value that is off by 0.01 from the official reported training results.

5.1 Baseline

The baseline score was calculated by classifying the first sentence in each paper as positive and all other sentences as negative samples. The reason for this is that choosing one sentence for each paper results in a similar positive to negative sample ratio as with the true data. Additionally, there are more positive first sentences than there would be with a random distribution of positives, so this provides a baseline that is slightly better than baseline. This method resulted in 69 true positives, 141 false positives, 2,163 true negatives, and 190 false negatives. The baseline precision, recall, and F1 scores are .33, .27, and .29.

5.2 Training Results

The best training results came from the cased SciBERT model with a mean F1 score of 0.72. However, SVM and XGBoost performed fairly well with F1 scores of 0.61. The training results for Non-BERT classifiers are presented first, and the results for the BERT models are presented in Section 5.2.2.

5.2.1 Non-BERT Training Results

The mean and standard deviation of the precision, recall, and F1 scores for the non-BERT classifiers are shown in Table 5.2. The first half of the table shows results using the dataset that contains contextual features, and the second half shows the results of training on sentence data only. The best text encoding for each model is reported in this table.

Many more of the classifiers were able to beat the baseline when using the data annotated with contextual features than when using the plain dataset. Only logistic

model	encoding	precision	recall	F1
Training results using data annotated with contextual features				
nn	NA	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
dnn	NA	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
rnn	embeddings	0.34 ± 0.02	0.13 ± 0.1	0.17 ± 0.11
knn	char ngrams	1.0 ± 0.0	0.1 ± 0.01	0.18 ± 0.01
Baseline	**NA**	**0.33**	**0.27**	**0.29**
gradient boost	word ngrams	1.0 ± 0.0	0.19 ± 0.01	0.32 ± 0.02
lstm	embeddings	0.74 ± 0.28	0.26 ± 0.25	0.33 ± 0.25
bilstm	embeddings	0.89 ± 0.12	0.24 ± 0.12	0.35 ± 0.12
random forest	word ngrams	0.78 ± 0.08	0.23 ± 0.02	0.36 ± 0.03
cnn	embeddings	0.67 ± 0.07	0.34 ± 0.12	0.43 ± 0.1
complement naive bayes	one-hot	0.41 ± 0.02	0.68 ± 0.01	0.51 ± 0.01
logistic regression	one-hot	0.73 ± 0.01	0.43 ± 0.02	0.54 ± 0.02
sgd	tf-idf	0.76 ± 0.07	0.5 ± 0.05	0.6 ± 0.04
svm	one-hot	0.68 ± 0.0	0.55 ± 0.01	**0.61 ± 0.01**
xgb	one-hot	0.93 ± 0.01	0.45 ± 0.01	**0.61 ± 0.01**
Training results using text data only				
nn	NA	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
dnn	NA	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
knn	NA	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
random forest	word ngrams	0.63 ± 0.21	0.06 ± 0.02	0.11 ± 0.03
cnn	embeddings	0.61 ± 0.36	0.1 ± 0.09	0.16 ± 0.13
gradient boost	char ngrams	1.0 ± 0.0	0.1 ± 0.02	0.18 ± 0.03
complement naive bayes	one-hot	0.2 ± 0.01	0.37 ± 0.01	0.26 ± 0.01
rnn	embeddings	0.45 ± 0.18	0.21 ± 0.13	0.26 ± 0.11
lstm	embeddings	0.59 ± 0.24	0.22 ± 0.13	0.29 ± 0.14
Baseline	**NA**	**0.33**	**0.27**	**0.29**
bilstm	embeddings	0.54 ± 0.27	0.26 ± 0.16	0.34 ± 0.19
logistic regression	one-hot	0.64 ± 0.0	0.27 ± 0.01	0.38 ± 0.01
svm	one-hot	0.49 ± 0.01	0.31 ± 0.01	0.38 ± 0.0
sgd	one-hot	0.46 ± 0.15	0.41 ± 0.09	0.41 ± 0.05
xgb	one-hot	0.74 ± 0.03	0.39 ± 0.01	**0.51 ± 0.01**

Table 5.2: Mean training results and standard deviation for linear, ensemble, and neural classifiers across ten runs. Only the results for the best encoding are reported.

regression, SVM, stochastic gradient descent, XGBoost, and RNN with BiLSTM improved on the baseline when trained on the plain dataset. The shallow and deep neural networks, RNN, and K-Nearest Neighbor were the only models that4 could not improve on the baseline when trained on the contextual dataset (these models also could not improve on the baseline when trained on the plain dataset).

Across both versions of the dataset, for the non-neural classifiers one-hot appears to be the most successful encoding. Keeping in mind that the char ngrams and word ngrams are variants on tf-idf, four of the classifiers trained on the dataset with contextual features preferred one-hot encodings and four preferred tf-idf. For the classifiers trained on the dataset without contextual features, five of the classifiers preferred one-hot encodings and two preferred tf-idf.

XGBoost performed the best for both versions of the dataset with F1 scores of 0.61 for the dataset with contextual features and 0.51 for the dataset without. SVM performed relatively well for the dataset with contextual features with an F1 score of 0.61 (tied with XGBoost). The basic shallow and deep neural networks were not able to learn either version of the dataset, and k-nearest neighbor was not able to learn the version without contextual features.

Predicted

	Positive	Negative	
True	17	21	Positive
	1	220	Negative

Table 5.3: The average confusion matrix for XGBoost with one-hot encoding trained on the dataset with contextual features.

	Predicted		
True	17	14	Positive
	8	220	Negative
	Positive	Negative	

Table 5.4: The average confusion matrix for SVM with one-hot encoding trained on the dataset with contextual features.

Overall, the neural classifiers performed poorly in comparison with other machine learning paradigms and the linear single learner classifiers performed the best on average as a group (although the ensemble classifier XGBoost performed the best over any other classifier). See Table 5.5 for the mean F1 scores by classifier category. Including contextual features such as header title and positional data appeared to be beneficial. The maximum F1 score for classifiers trained with contextual features is ten points higher than the maximum F1 score for classifiers trained on sentence data only. The mean F1 score across all 14 classifiers trained on the dataset with contextual data is .35 and the mean F1 score across all classifiers trained without contextual features is .24, which is a significant difference.

classifier type	mean F1 for dataset with context	mean F1 for sentence data only
single-learner linear	0.48	0.29
ensemble classifier	0.43	0.26
neural network	0.21	0.18

Table 5.5: Mean F1 score by classifier type

The precision score is higher than the recall score for almost every classifier. This makes sense for a dataset with many fewer positive instances than negative. The precision score is the number of correct labelings of positive samples (true positives) scaled by the sum of true positives and the number of incorrect positive labelings (false positives): precision $= \frac{TP}{TP+FP}$. So the fewer false positives a model returns, the higher the precision score will be. The recall score is the number of true positives scaled by the sum of true positives and the number of incorrect negative labelings (false negatives): recall $= \frac{TP}{TP+FN}$. So the more false negatives a model returns, the lower the recall score will be. Classifiers that are biased towards the class that is over-represented are less likely to falsely assign the under-represented label and more likely to miss members of the under-represented class because they are less likely to assign that label overall.

Notice that k-nearest neighbor and gradient boosting algorithms have precision scores of 1 and very, very small recall scores. This indicates that these classifiers are labeling almost every single sample as negative and have not actually learned how to identify a task description.

	Predicted	
True	Positive	Negative
Positive	1	9
Negative	0	249

Table 5.6: The average confusion matrix for K-Nearest Neighbor with tf-idf (character ngrams) trained on the dataset with contextual features.

	Predicted	
True	Positive	Negative
Positive	4	17
Negative	0	238

Table 5.7: The average confusion matrix for gradient boosting with tfidf (word ngrams) trained on the dataset with contextual features.

Another interesting case is CNB which has higher recall scores than precision scores. The creators of CNB designed it specifically to perform well on unbalanced text data by addressing sources of bias for multinomial Naive Bayes (Rennie et al. 2003). It does appear to be the case that CNB was less biased than some of the other classifiers, as the ratio of true positives to false negatives was lower resulting in a higher recall score.

	Predicted		
True	10	5	Positive
	14	230	Negative
	Positive	Negative	

Table 5.8: The confusion matrix for complement naive Bayes with one-hot encoding trained on the dataset with contextual features.

5.2.2 BERT Training Results

The precision, recall, and F1 scores for four variants of BERT models are shown in Table 5.9. As with the Non-BERT training results, scores are reported for both the dataset with additional contextual features and the dataset containing sentences alone. Only the best combinations of hyperparameters are reported here: for the full results, see Appendix C. All of the BERT results were higher than the baseline.

Unlike for the majority of the algorithms with results reported in Section 5.2.1, the highest performing BERT model scored better on the dataset comprising sentence data only without additional features. The cased scibert model earned an average F1 score of 0.72 on the simple dataset and an average F1 score of 0.69 on the dataset containing contextual features. However, the other three models all returned higher mean scores when trained on the dataset containing contextual features. The mean F1 score across all four models trained on the contextual dataset is 0.7, while the mean score across all four models trained on the simple dataset is 0.68. Notice also that the standard deviations are somewhat high, indicating a not insignificant spread around the mean. From this data it is unclear whether one variant of the dataset is better than the other.

Four of the BERT models trained for four epochs, three of them trained for three epochs, and one of them trained for only two epochs. It is interesting that for four of the models the best choice for number of epochs was less than four, as one would expect the model to continue to improve. A visual inspection of the loss per epoch shows that there is room for the losses for the model trained for two epochs and one of the models trained for three epochs to continue to reduce (see Figure 5.1 for a model trained for two epochs, Figure 5.2 for a model trained for three epochs, and Figure 5.3 for a model trained for four epochs).

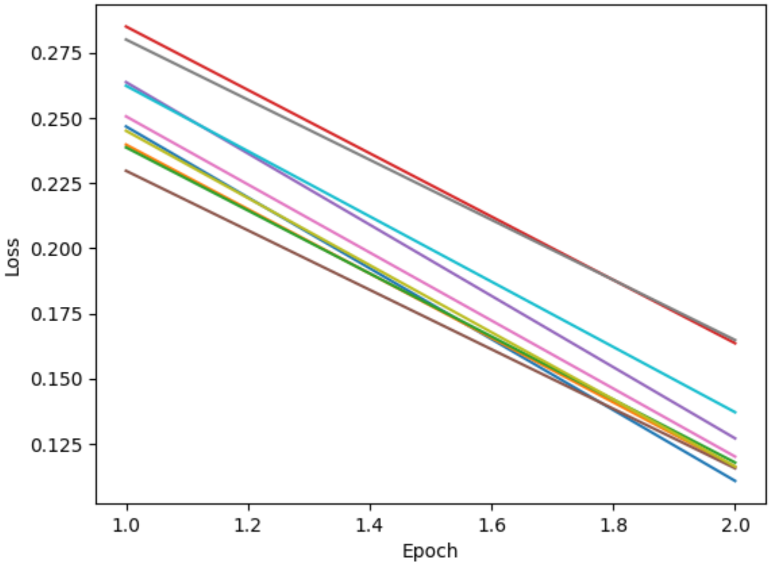


Figure 5.1: Training losses per epoch for scibert_cased trained with contextual data for two epochs with batch size of 32 and learning rate of 5e-05. Each curve shows the losses for one of ten runs.

model	epochs	batch size	learning rate	metric	score
Training results using data annotated with contextual features					
bert-uncased	4	16	2e-05	Precision	0.65 ± 0.14
				Recall	0.72 ± 0.14
				F1	0.68 ± 0.12
bert-cased	4	16	2e-05	Precision	0.69 ± 0.1
				Recall	0.73 ± 0.1
				F1	**0.71 ± 0.09**
scibert_uncased	3	16	3e-05	Precision	0.69 ± 0.03
				Recall	0.73 ± 0.12
				F1	**0.71 ± 0.06**
scibert_cased	2	32	5e-05	Precision	0.7 ± 0.08
				Recall	0.7 ± 0.11
				F1	0.69 ± 0.08
Training results using text data only					
bert-uncased	3	32	5e-05	Precision	0.63 ± 0.08
				Recall	0.7 ± 0.1
				F1	0.66 ± 0.08
bert-cased	4	16	2e-05	Precision	0.65 ± 0.08
				Recall	0.67 ± 0.11
				F1	0.66 ± 0.08
scibert_uncased	3	16	3e-05	Precision	0.7 ± 0.12
				Recall	0.67 ± 0.08
				F1	0.67 ± 0.08
scibert_cased	4	32	5e-05	Precision	0.73 ± 0.11
				Recall	0.71 ± 0.07
				F1	**0.72 ± 0.08**

Table 5.9: Mean training results and standard deviation for four BERT classifiers across ten runs. Only the results for the best hyperparameter combination are reported here.

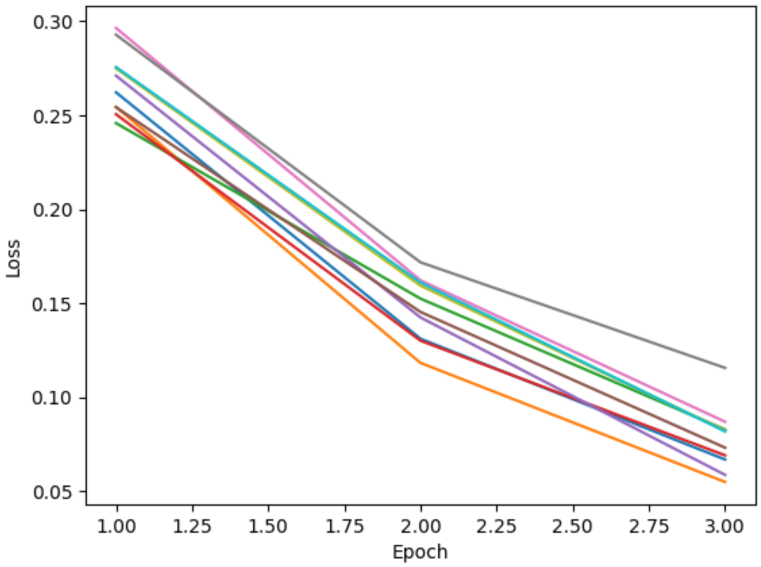


Figure 5.2: Training losses per epoch for bert-uncased trained without contextual data for three epochs with batch size of 32 and learning rate of 5e-05.

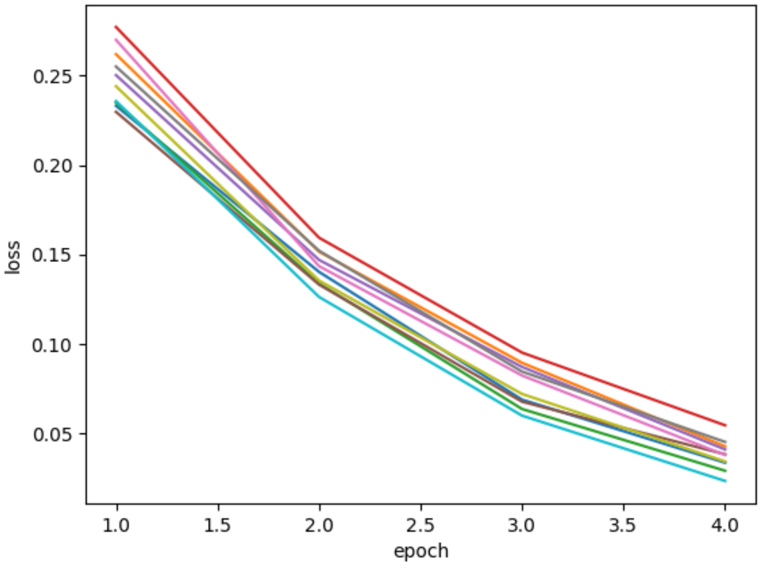


Figure 5.3: Training losses per epoch for scibert_cased trained without contextual data for four epochs with batch size of 32 and learning rate of 5e-05.

5.3 Test Results

Tests were run using the cased SciBERT model fine-tuned on the simple dataset over four epochs with a batch size of 32 and a learning rate of 5e-05 (the model with the highest training results). Three versions of the test dataset were used in order to determine how well our system would perform given data of varying levels of preprocessing. The three versions of the test data are:

1. The dataset manually reduced in the same way that the training data is reduced. Only sections that contain a task description are included (for resulting confusion matrix see Figure 5.10);

2. The dataset automatically reduced by learning which section headers are likely to appear over a section containing a task description. Only sections that have a high probability of containing a task description are included (for resulting confusion matrix see Figure 5.11);

3. The full dataset without any sections removed from any papers(for resulting confusion matrix see Figure 5.12).

	Predicted		
	Positive	Negative	
True	24	10	Positive
	6	287	Negative

Table 5.10: The confusion matrix for the test results on the manually reduced test set.

	Predicted		
True	21	8	Positive
	63	1104	Negative
	Positive	Negative	

Table 5.11: The confusion matrix for the test results on the automatically reduced test set.

	Predicted		
True	25	9	Positive
	128	4597	Negative
	Positive	Negative	

Table 5.12: The confusion matrix for the test results on the full test set.

For more information on how these variants of the test data were created, see Section 4.1.1. The manually reduced dataset is the most balanced while the full test set is the least balanced. The scores reflect the variation in proportion of positive to negative samples; the most balanced dataset is associated with the highest F1 score (0.75) and the least balanced is associated with the lowest (0.27).

test dataset	precision	recall	F1
manually reduced	0.8	0.71	0.75
automatically reduced	0.25	0.72	0.37
full test set	0.16	0.74	0.27

Table 5.13: Test results for each version of the test dataset

Surprisingly, the F1 score for the manually reduced dataset (0.75) is higher than the mean training result (0.72). This is surprising because the hyperparameter settings were chosen based only on the training data; the test data was unseen during the process of hyperparameter selection. However, 0.75 is within one standard deviation of the mean training result (standard deviation = ±0.08). The dataset used to train the model used to classify the test set was bigger than the dataset used during training experiments because 10% of it did not need to be set aside for validation. It is possible that, due to the relatively small amount of positive samples, that increasing the training data by a small amount could be enough to improve results on during testing.

5.3.1 Error Analysis

In this section we provide examples of the false negatives and false positives returned by our system when tested on the test set. It is not always clear why our system would label true samples as non task descriptions, although it does seem that shared task overview papers with subtasks may be particularly challenging. In contrast, many of the false positives are near misses, sentences that could have been annotated as positive but were not because there was a better option.

False Negatives

Two of the samples from the test data show one weakness of reformulating the task as a sentence classification task rather than a sequence labeling task. Both of the sentences shown below are labeled as task descriptions not because the entire sentence meets the criteria, but because a phrase from each was annotated as a task description phrase in our gold corpus (task descriptions shown in italics).

> **Example FN-1**
>
> Unsupervised Word Sense Induction and Discrimination (WSID,

> also known as corpus-based unsupervised systems) has followed this line of thinking, and tries to *induce word senses directly from the corpus.*
>
> -*Semeval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems*, Agirre and Etxabe 2007.
>
> **Example FN-2**
>
> The Genia event task, *a bio-molecular event extraction task*, is arranged as one of the main tasks of BioNLP Shared Task 2011.
>
> -*Overview of Genia Event Task in BioNLP Shared Task 2011*, Kim, Y. Wang, et al. 2011.

However, our system labeled both of these sentences as non task descriptions. This illustrates the danger that the language surrounding the task description phrases in each sample can outweigh the language inside the task description phrases.

Other errors reflect the same ambiguities that were difficult for human annotators to resolve, as in the following example containing two consecutive sentences.

> **Example FN-3**
>
> [1]We present a counterfactual recognition (CR) task, the task of determining whether a given statement conveys counterfactual thinking or not, and further analyzing the causal relations indicated by counterfactual statements. [2]In our counterfactual recognition task, we aim to model counterfactual semantics and reasoning in natural language.
>
> -*SemEval-2020 Task 5: Counterfactual Recognition*, Yang et al. 2020.

In the gold standard corpus, we labeled sentence FN-3.2 as the task description rather than sentence FN-3.1 following rule 2 of our guidelines for choosing between two or more valid candidates (see Section 3.4.3), which states that, when there are two candidates that are similar in content but vary in clarity or technical expressiveness, the annotator should select the candidate that is easiest to read out of context. During testing, our system selected sentence FN-3.1 as a task description and labeled sentence FN-3.2 as a non task description. This is an example of a difficult case for the system to distinguish between two potential candidates.

The paper *Overview of the Fourth Social Media Mining for Health (#SMM4H) Shared Task at ACL 2019* (Weissenbacher et al. 2019) was a particularly interesting sample in the test set. The shared task in question contained four distinct tasks and a. valid task description for each task is found in consecutive sentences. For this reason, the annotators labeled each of those sentences as part of the task description, as seen below:

> **Example FN-4**
>
> [1]Task 1 asked participants to distinguish tweets reporting an adverse drug reaction (ADR) from those that do not. [2]Task 2, a follow-up to Task 1, asked participants to identify the span of text in tweets reporting ADRs. [3]Task 3 is an end-to-end task where the goal was to first detect tweets mentioning an ADR and then map the extracted colloquial mentions of ADRs in the tweets to their corresponding standard concept IDs in the MedDRA vocabulary. [4]Finally, Task 4 asked participants to classify whether a tweet contains a personal mention of one's health, a more general discussion of the health issue, or is an unrelated mention.
>
> -*Overview of the Fourth Social Media Mining for Health (#SMM4H)*

Shared Task at ACL 2019, Weissenbacher et al. 2019.

During testing, our system labeled sentences FN-4.1, FN-4.2, and FN-4.3 as task descriptions, but labeled sentence FN-4.4 as a non task description. It is unclear why the system did not label the fourth sentence as a task description but it is encouraging that the system was able to detect a task description spanning multiple sentences.

Our system was not always able to detect descriptions spanning multiple sentences, as seen in Examples **FN-5** and **FN-6**:

Example FN-5

[1]This task required participating systems to annotate instances of nouns, verb, and adjectives using Word-Net 3.1 (Fellbaum, 1998), which was selected due to its fine-grained senses. [2]Participants could label each instance with one or more senses, weighting each by their applicability.

-*SemEval-2013 Task 13: Word Sense Induction for Graded and Non-Graded Senses*, Jurgens and Klapaftis 2013.

Example FN-6

[1]Given two sentences, s1 and s2, an STS system would need to return a similarity score. [2]Participants can also provide a confidence score indicating their confidence level for the result returned for each pair, but this confidence is not used for the main results.

-*SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity*, Agirre, Cer, et al. 2012.

Our system correctly selected sentences FN-5.1 and FN-6.1, but failed to identify sentences FN-5.2 and FN-6.2 as task descriptions. The difference between these examples and **Example FN-4** is that each sentence in **Example FN-4** contains phrases that indicates a task is being discussed (e.g. the word *Task*, the phrase *asked participants*, and the snippet *end-to-end task where the goal was*). However, in **Example FN-5** and **Example FN-6**, the second sentence is auxiliary to the first; it provides detail to the first sentence, but does not stand alone as a task description. Our system detects task descriptions at the sentence level, so was not trained to look for such a relationship between consecutive sentences.

Papers with subtasks presented were difficult cases during the annotation project, and the system sometimes failed to extract task descriptions from these papers in the test data. Examples **FN-7** and **FN-8** show the task descriptions extracted by a human annotator for two papers with subtasks. Our system labeled every sentence from both of these papers as a non task description.

> **Example FN-7**
>
> Nine sub-tasks were included, covering problems in time expression identification, event expression identification and temporal relation identification.
>
> -*SemEval-2015 Task 6: Clinical TempEval*, Bethard et al. 2015.
>
> **Example FN-8**
>
> In this paper, we present two subtasks designed to evaluate such phrasal models: a. Semantic similarity of words and compositional phrases b. Evaluating the compositionality of phrases in context.
>
> -*SemEval-2013 Task 5: Evaluating Phrasal Semantics*, Korkontzelos et al. 2013.

False Positives

The false positives returned from our system tend to be near misses (sentences that could have been chosen during annotation but were rejected due to the presence of a better candidate). Another interesting trend is that the false positives are often adjacent to true positives extracted by the system. While these false positives may be lacking in detail on their own, some of them work quite well as auxiliary sentences to the true positives. Both of these characteristics can be seen in Examples **FP-1**, **FP-2**, and **FP-3**.

> **Example FP-1**
>
> [1]Parser Evaluation using Textual Entailments (PETE) is a shared task that involves recognizing textual entailments based on syntactic information alone. [2]Given two text fragments called ”text” and ”hypothesis”, textual entailment recognition is the task of determining whether the meaning of the hypothesis is entailed (can be inferred) from the text.
>
> -*SemEval-2010 Task 12: Parser Evaluation using Textual Entailments*, Yuret, Han, and Turgut 2010.

Example FP-1 shows all of the sentences our system labeled as task descriptions from the paper *SemEval-2010 Task 12: Parser Evaluation using Textual Entailments* (Yuret, Han, and Turgut 2010). Sentence FP-1.1 is also labeled as a task description in the dataset but sentence FP-1.2 was a false positive. The reason sentence FP-1.2 was not selected during the annotation process was because the sentence provides a general explanation of textual entailment tasks rather than the specific shared task that the participating systems must solve. However, sentence FP-1.2 does provide valuable background information for sentence FP-1.1.

> **Example FP-2**
>
> [1]The goal of the shared task on shallow discourse parsing is to detect and categorize individual discourse relations. [2]Specifically, given a newswire article as input, a participating system is asked to return a set of discourse relations contained in the text. [3]A discourse relation, as defined in the PDTB, from which the training data for the shared task is drawn, is a relation taking two abstract objects (events, states, facts, or propositions) as arguments.
>
> -*The CoNLL-2015 Shared Task on Shallow Discourse Parsing*, Xue et al. 2015.

Example FP-2 shows all three sentences extracted by our system from *The CoNLL-2015 Shared Task on Shallow Discourse Parsing* (Xue et al. 2015). Sentence FP-2.1 is a false positive, but sentences FP-2.2 and FP-2.3 are true positives. Sentence FP-2.1 was not included as a task description in our gold dataset because it is lacking in detail. However, all three consecutive sentences are quite readable out of context and would work well in practice as a long task description.

> **Example FP-3**
>
> [1]This task seeks to evaluate the capability of systems for predicting dimensional sentiments of Chinese words and phrases. [2]For a given word or phrase, participants were asked to provide a real-valued score from 1 to 9 for both the valence and arousal dimensions, respectively indicating the degree from most negative to most positive for valence, and from most calm to most excited for arousal.
>
> -*IJCNLP-2017 Task 2: Dimensional Sentiment Analysis for*

Chinese Phrases, Yu et al. 2017

In **Example FP-3**, both of these consecutive sentences were labeled by our system as task descriptions. However, only sentence FP-3.2 was a true positive. FP-3.2 was selected as the task description during the annotation process because of the detail provided for what the participating system should do for each word or phrase. However, sentence FP-3.2 is lacking in that it does not mention the language of the words and phrases (Chinese). Both of these sentences together form a more complete task description.

In **Example FP-4** we see another instance of a task fragmented into sub-parts that confuses our system. The task contains two tracks, a ”shallow” track and a ”deep” track. However, the task description should try to describe the overall task rather than one of the tracks. The gold label meets this criteria, as seen in sentence FP-4.1. Our system correctly extracted sentence FP-4.1, but incorrectly extracted a sentence that describes the shallow track (sentence FP-4.2).

Example FP-4

[1]The Sr’18 task is to generate sentences from structures at the level of abstraction of outputs in state-of-the-art parsing, encouraging participants to explore the extent to which neural network parsing algorithms can be reversed for generation.

[2]The task amounts to determining the word order and inflecting words.

-*The First Multilingual Surface Realization Shared Task (SR’18): Overview and Evaluation Results*, Mille et al. 2018.

Another issue with this sample is that our system chose two sentences that are not consecutive. This behavior could be fixed with a simple rule that chooses the sentence with the higher probability of being a task description.

6 Conclusion

This section provides an outline of our contributions. Then it summarizes some key takeaways from our annotation project and experiments and outlines some opportunities for future work. Lastly, this section provides a brief discussion of ethical considerations related to the enhancement of digital libraries with artificial intelligence.

6.1 Thesis Contributions

The contributions of this thesis are as follows:

- A new corpus of 256 Shared Task Overview papers with annotated task descriptions;
- Code for preprocessing, training, and testing machine learners for binary text classification tasks;
- An evaluation of 18 models for binary sentence classification on a corpus of shared task overview papers with unbalanced data;
- The publication of our system description for SemEval-2021 Task 11: NLPContributionGraph (Martin and Pedersen 2021);
- The poster presentation of our annotation project for SciNLP-2021 (Martin, D'Souza, and Pedersen 2021).

6.2 Discussion of Results

Overall the machine learning results were encouraging. In particular, the false positives returned by our system were all sensible and met many of the task description requirements. The false negatives show that our system does not pick up on task description phrases that are embedded in longer, more general sentences. It also shows that our system struggled with papers containing subtasks or multiple tracks. This makes sense, as those scenarios were some of the hardest to deal with during the annotation project. One of the biggest takeaways from the machine learning results is that preprocessing is incredibly important. Decisions on how to construct and filter the datasets were much more impactful on final results than the specific hyperparameters chosen.

6.3 Future Work

As noted in Section 5.3.1, most of the false positives extracted by our system were adjacent to true positives and provided additional task information that was useful and relevant. A future annotation project could be designed that is roughly based on our guidelines but more permissive in terms of the sentences to be extracted. Instead of prioritizing conciseness, this project would prioritize extracting as much information as is needed to formulate a more complete summary of the shared task. This resource could then be used as material for an extractive task summarization project.

There are a few potential improvements that could be applied to our machine learning process. More could be done to take advantage of the innate hierarchical structure of scientific papers. Future work should be done to explore how the formatting and organization of papers can be leveraged to provide useful information to machine readers. Another future project could apply intermediate pre-training to SciBERT by fine-tuning it on a different dataset that contains task annotations before fine-tuning it on our corpus. Lastly, there is more work to be done to determine the best way to filter the test data so that it is as ”real world” as possible while still being balanced enough for machine readers to be able to extract task descriptions.

6.4 Ethical Considerations

What unintended consequences might arise from the enhancement of digital libraries with AI-driven tools? If artificial intelligence is to continue to be deployed more and more in the interfaces between humans and scientific information, it is important to consider how that could impact human behavior and the research process. This illusion can lead people to stop searching for more information on a topic sooner than they should. In the case of digital libraries, there are two groups of people whose behavior may be affected AI-enhanced tools: the readers using the digital library in their research; and the writers of work that will eventually be hosted by the digital library.

Nguyen 2021 describes how the exaggerated illusion of clarity in systems can cause humans to experience a false sense of complete understanding, and can provide a framework for understanding how AI might impact readers. Nguyen's work is focused on two broad case studies: the way epistemic manipulators can use a false sense of clarity to accept conspiracy theories; and the way that institutionalized quantification of complex information can create a false sense of clarity for people working in bureaucratic systems. It is interesting to apply Nguyen's framework to the setting of AI-enhanced digital libraries. In what ways might AI integration create a false sense of clarity for researchers? How might researchers anticipate and respond to such a phenomenon? What responsibility do the developers of digital library technologies have in terms of managing the phenomenon of the seduction of clarity?

Following are three tools that are simultaneously incredibly helpful for researchers and also may carry a risk of creating the illusion of clarity.

- Algorithmic feeds and recommender systems that can be customized to the needs of individual researchers are powerful tools that have great potential to help with the literary discovery process of the research cycle;
- Pages that act as interfaces between users and academic papers, providing bibliometric data such as publication data and references, and content-driven data such as tables, figures, and paper summaries, are useful for deciding whether a given paper should be skimmed or read in depth;
- And quantifications such as the number of citations are useful in understanding the impact or relevance of a given work.

Researchers should use these tools to help them work through large amounts of relevant research, but should also be considerate of whether they are investigating topics thoroughly enough. While users of digital libraries should be responsible with how they use AI-driven tools, digital library developers should also consider the impacts of their work. For example, developers of algorithmic feeds should consider what kinds of bias could be perpetuated by their systems.

A potential downstream effect of automating aspects of digital libraries is how that could effect the way researchers write their papers. Digital libraries may be one of the first places that people's work is discovered, and so the way work is presented matters. The knowledge that automated processes affect how readers interface with papers could influence how writers communicate their work, encouraging them to write in such a way that their paper is easier for an AI to process. Is this necessarily always a bad thing if it encourages writers to communicate in a way that is more clear and standardized? We observed during the annotation phase of our project that, while Shared Task Overview papers typically follow a few formats, there is still a lot of variety in writing style that can make it more difficult to annotate consistently; if scientific communication became more formulaic it would certainly make it easier for machines to structure scientific contributions. Whether researchers should have to make these changes so that their work can be more easily processed and presented in digital libraries is another question, and we should be alert to the potential for digital libraries to become biased against different kinds of researchers whether it is due to their communication style, institution, or research area.

References

Agirre, Eneko, Daniel Cer, et al. (July 2012). “SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity”. In: **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Montréal, Canada: Association for Computational Linguistics, pp. 385–393. URL: https://aclanthology.org/S12-1051 (cit. on p. 71).

Agirre, Eneko and Aitor Soroa Etxabe (2007). “SemEval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems”. In: *Fourth International Workshop on Semantic Evaluations (SemEval-2007)* (cit. on p. 69).

Akkaradamrongrat, Suphamongkol, Pornpimon Kachamas, and Sukree Sinthupinyo (2019). “Text Generation for Imbalanced Text Classification”. In: *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 181–186 (cit. on p. 93).

Ammar, Waleed et al. (2018). “Construction of the Literature Graph in Semantic Scholar”. In: *NAACL* (cit. on pp. 4, 10, 12).

Auer, S. et al. (2020). “Improving Access to Scientific Literature with Knowledge Graphs”. In: *Bibliothek Forschung und Praxis* 44, pp. 516–529 (cit. on p. 4).

Augenstein, Isabelle et al. (Aug. 2017). "SemEval 2017 Task 10: ScienceIE - Extracting Keyphrases and Relations from Scientific Publications". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 546–555. DOI: 10.18653/v1/S17-2091. URL: https://www.aclweb.org/anthology/S17-2091 (cit. on pp. 8, 10).

Bada, Michael et al. (2011). "Concept annotation in the CRAFT corpus". In: *BMC Bioinformatics* 13, pp. 161–161 (cit. on p. 9).

Beltagy, Iz, Kyle Lo, and Arman Cohan (Nov. 2019). "SciBERT: A Pretrained Language Model for Scientific Text". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3615–3620. DOI: 10.18653/v1/D19-1371. URL: https://aclanthology.org/D19-1371 (cit. on p. 16).

Bender, Emily M. and Batya Friedman (2018). "Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science". In: *Transactions of the Association for Computational Linguistics* 6, pp. 587–604. DOI: 10.1162/tacl_a_00041. URL: https://aclanthology.org/Q18-1041 (cit. on p. 17).

Bethard, Steven et al. (June 2015). "SemEval-2015 Task 6: Clinical TempEval". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, pp. 806–814. DOI: 10.18653/v1/S15-2136. URL: https://aclanthology.org/S15-2136 (cit. on p. 72).

Bird, Steven et al. (May 2008a). "The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics". In: *Pro-*

ceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08). Marrakech, Morocco: European Language Resources Association (ELRA). URL: http://www.lrec-conf.org/proceedings/lrec2008/pdf/445_paper.pdf (cit. on p. 9).

Bird, Steven et al. (May 2008b). "The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics". In: *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. Marrakech, Morocco: European Language Resources Association (ELRA) (cit. on p. 11).

Bojanowski, Piotr et al. (2017). "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. DOI: 10.1162/tacl_a_00051. URL: https://aclanthology.org/Q17-1010 (cit. on p. 49).

Buscaldi, D. (2018). "Improving access to scientific literature: a semantic IR perspective". In: *Proceedings of the 5th Spanish Conference on Information Retrieval* (cit. on p. 4).

Cabanac, Guillaume et al. (2016). "Report on the Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL 2016)". In: *ACM SIGIR Forum* 50, pp. 36–43 (cit. on p. 7).

Chandrasekaran, Muthu Kumar, Guy Feigenblat, et al. (2020). "Overview of the First Workshop on Scholarly Document Processing (SDP)". In: *SDP* (cit. on p. 8).

Chandrasekaran, Muthu Kumar, Michihiro Yasunaga, et al. (2019). "Overview and Results: CL-SciSumm Shared Task 2019". In: *Proceedings of the 4th Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL 2019) @ SIGIR 2019*. Paris, France (cit. on pp. 7, 8).

Cohen, Jacob (1960). “A Coefficient of Agreement for Nominal Scales”. In: *Educational and Psychological Measurement* 20, pp. 37–46 (cit. on p. 33).

D’Souza, Jennifer, Sören Auer, and Ted Pedersen (Aug. 2021). “SemEval-2021 Task 11: NLPContributionGraph - Structuring Scholarly NLP Contributions for a Research Knowledge Graph”. In: *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*. Online: Association for Computational Linguistics, pp. 364–376. DOI: 10.18653/v1/2021.semeval-1.44. URL: https://aclanthology.org/2021.semeval-1.44 (cit. on pp. 2, 8, 12, 19).

Devlin, Jacob et al. (2019a). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL* (cit. on pp. 6, 15).

— (June 2019b). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423 (cit. on p. 49).

Dhawan, S. M., B. M. Gupta, and N. K. Singh (2020). “Global Machine-learning Research: a scientometric assessment of global literature during 2009-18”. In: *World Digit. Libr.* 13, pp. 105–120 (cit. on p. 4).

Gábor, Kata, Davide Buscaldi, et al. (June 2018). “SemEval-2018 Task 7: Semantic Relation Extraction and Classification in Scientific Papers”. In: *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 679–688. DOI: 10.18653/v1/S18-1111. URL: https://www.aclweb.org/anthology/S18-1111 (cit. on pp. 8, 10, 11).

Gábor, Kata, Haïfa Zargayouna, et al. (2016). “Semantic Annotation of the ACL Anthology Corpus for the Automatic Analysis of Scientific Literature”. In: *LREC* (cit. on pp. 10, 11).

Glavaš, Goran et al. (Dec. 2020). "SemEval-2020 Task 2: Predicting Multilingual and Cross-Lingual (Graded) Lexical Entailment". In: *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Barcelona (online): International Committee for Computational Linguistics, pp. 24–35. DOI: 10.18653/v1/2020.semeval-1.2. URL: https://aclanthology.org/2020.semeval-1.2 (cit. on p. 28).

GROBID (2008–2022). https://github.com/kermitt2/grobid. swh: 1:dir:dab86b296e3c3216e2241968f0d63b68e8209d3c (cit. on p. 43).

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart (2008). "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, pp. 11–15 (cit. on p. 96).

Haneczok, Jacek et al. (Aug. 2021). "Fine-grained Event Classification in News-like Text Snippets - Shared Task 2, CASE 2021". In: *Proceedings of the 4th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2021)*. Online: Association for Computational Linguistics, pp. 179–192. DOI: 10.18653/v1/2021.case-1.23. URL: https://aclanthology.org/2021.case-1.23 (cit. on p. 24).

He, Pengcheng et al. (May 2021). "DeBERTa: Decoding-Enhanced BERT with Disentangled Attention". In: *2021 International Conference on Learning Representations* (cit. on p. 13).

Head, Andrew et al. (2021). "Augmenting Scientific Papers with Just-in-Time, Position-Sensitive Definitions of Terms and Symbols". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (cit. on p. 7).

Hendrickx, Iris and Antal van den Bosch (July 2001). "Dutch Word Sense Disambiguation: Data and Preliminary Results". In: *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*.

Toulouse, France: Association for Computational Linguistics, pp. 13–16. URL: https://aclanthology.org/S01-1003 (cit. on p. 28).

Hou, Yufang et al. (Apr. 2021). “TDMSci: A Specialized Corpus for Scientific Literature Entity Tagging of Tasks Datasets and Metrics”. In: pp. 707–714. DOI: 10.18653/v1/2021.eacl-main.59. URL: https://aclanthology.org/2021.eacl-main.59 (cit. on p. 11).

Jain, Sarthak et al. (2020). “SciREX: A Challenge Dataset for Document-Level Information Extraction”. In: *ACL* (cit. on p. 12).

Jurgens, David and Ioannis Klapaftis (June 2013). “SemEval-2013 Task 13: Word Sense Induction for Graded and Non-Graded Senses”. In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA: Association for Computational Linguistics, pp. 290–299. URL: https://aclanthology.org/S13-2049 (cit. on p. 71).

Kahusk, Neeme, Heili Orav, and Haldur Õim (July 2001). “Sensiting Inflectionality: Estonian Task for SENSEVAL-2”. In: *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*. Toulouse, France: Association for Computational Linguistics, pp. 25–28. URL: https://aclanthology.org/S01-1006 (cit. on p. 27).

Kim, Jin-Dong, Tomoko Ohta, et al. (2003). “GENIA corpus - a semantically annotated corpus for bio-textmining”. In: *Bioinformatics* 19 Suppl 1, pp. i180–2 (cit. on p. 9).

Kim, Jin-Dong, Yue Wang, et al. (2011). “Overview of Genia Event Task in BioNLP Shared Task 2011”. In: *BioNLP@ACL* (cit. on p. 69).

Knoth, Petr, Zdenek Zdráhal, and Andreas Juffinger (2012). “Special Issue on Mining Scientific Publications”. In: *D Lib Mag.* 18 (cit. on p. 7).

Korkontzelos, Ioannis et al. (June 2013). "SemEval-2013 Task 5: Evaluating Phrasal Semantics". In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA: Association for Computational Linguistics, pp. 39–47. URL: https://aclanthology.org/S13-2007 (cit. on p. 72).

Larsen, Peder Olesen and Markus von Ins (2010). "The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index". In: *Scientometrics* 84, pp. 575–603 (cit. on p. 4).

Li, Yang et al. (2020). "A bibliometric analysis on deep learning during 2007–2019". In: *International Journal of Machine Learning and Cybernetics*, pp. 1–20 (cit. on p. 4).

Liu, Haoyang, M. Janina Sarol, and Halil Kilicoglu (Aug. 2021). "UIUC_BioNLP at SemEval-2021 Task 11: A Cascade of Neural Models for Structuring Scholarly NLP Contributions". In: *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*. Online: Association for Computational Linguistics, pp. 377–386. DOI: 10.18653/v1/2021.semeval-1.45. URL: https://aclanthology.org/2021.semeval-1.45 (cit. on pp. 13, 45).

Lo, Kyle et al. (2020). "S2ORC: The Semantic Scholar Open Research Corpus". In: *ACL* (cit. on p. 12).

Luan, Yi et al. (2018). "Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction". In: *EMNLP* (cit. on p. 10).

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). "Introduction to Information Retrieval: Scoring, term weighting, and the vector space model". In: (cit. on p. 48).

Martin, Anna, Jennifer D'Souza, and Ted Pedersen (2021). "Annotating Natural Language Processing Shared Task Descriptions". In: Poster presented at SciNLP 2021: 2nd Workshop on Natural Language Processing for Scientific Text (cit. on pp. 8, 77).

Martin, Anna and Ted Pedersen (Aug. 2021). "Duluth at SemEval-2021 Task 11: Applying DeBERTa to Contributing Sentence Selection and Dependency Parsing for Entity Extraction". In: *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*. Online: Association for Computational Linguistics, pp. 490–501. DOI: 10.18653/v1/2021.semeval-1.60. URL: https://aclanthology.org/2021.semeval-1.60 (cit. on pp. 2, 13, 77).

McHugh, M. L. (2012). "Interrater reliability: the kappa statistic". In: *Biochemia Medica* 22, pp. 276–282 (cit. on p. 34).

Mihalcea, Rada et al. (July 2004). "An evaluation exercise for Romanian Word Sense Disambiguation". In: *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. Barcelona, Spain: Association for Computational Linguistics, pp. 29–32. URL: https://aclanthology.org/W04-0808 (cit. on p. 28).

Mille, Simon et al. (2018). "The First Multilingual Surface Realisation Shared Task (SR'18): Overview and Evaluation Results". In: (cit. on p. 75).

Milosevic, Nikola et al. (2016). "Disentangling the Structure of Tables in Scientific Literature". In: *NLDB* (cit. on p. 7).

Nguyen, C. Thi (2021). "The Seductions of Clarity". In: *Royal Institute of Philosophy Supplement* 89, pp. 227–255 (cit. on p. 79).

Radford, Alec et al. (2019). "Language Models are Unsupervised Multitask Learners". In: (cit. on p. 6).

Rennie, Jason D. M. et al. (2003). "Tackling the Poor Assumptions of Naive Bayes Text Classifiers". In: *ICML* (cit. on p. 62).

Robert-Inacio, F. and L. Yushchenko (2014). "Visual attention model for computer vision". In: *Biologically Inspired Cognitive Architectures* 7, pp. 26–38. ISSN: 2212-683X. DOI: https://doi.org/10.1016/j.bica.2013.11.001. URL: https://www.sciencedirect.com/science/article/pii/S2212683X13000935 (cit. on p. 13).

Sanderson, Mark and W. Bruce Croft (2012). "The History of Information Retrieval Research". In: *Proceedings of the IEEE* 100, pp. 1444–1451 (cit. on p. 5).

Schumann, Anne-Kathrin and Héctor Martínez Alonso (2018). "Automatic Annotation of Semantic Term Types in the Complete ACL Anthology Reference Corpus". In: *LREC* (cit. on p. 11).

Specia, Lucia, Sujay Kumar Jauhar, and Rada Mihalcea (July 2012). "SemEval-2012 Task 1: English Lexical Simplification". In: **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Montréal, Canada: Association for Computational Linguistics, pp. 347–355. URL: https://aclanthology.org/S12-1046 (cit. on p. 23).

Tjong Kim Sang, Erik F. and Sabine Buchholz (2000). "Introduction to the CoNLL-2000 Shared Task Chunking". In: *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*. URL: https://aclanthology.org/W00-0726 (cit. on p. 23).

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30 (cit. on pp. 14, 15).

Viera, Anthony J, Joanne M Garrett, et al. (2005). "Understanding interobserver agreement: the kappa statistic". In: *Fam med* 37.5, pp. 360–363 (cit. on p. 35).

Vylomova, Ekaterina et al. (2020). "SIGMORPHON 2020 Shared Task 0: Typologically Diverse Morphological Inflection". In: *SIGMORPHON* (cit. on p. 36).

Wang, Alex, Yada Pruksachatkun, et al. (2019). "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems". In: *NeurIPS* (cit. on pp. 6, 13).

Wang, Alex, Amanpreet Singh, et al. (Nov. 2018). "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: pp. 353–355. DOI: 10.18653/v1/W18-5446. URL: https://aclanthology.org/W18-5446 (cit. on pp. 6, 13).

Ware, Mark and Michael Mabe (2015). "The STM report: An overview of scientific and scholarly journal publishing fourth edition". In: (cit. on p. 4).

Weissenbacher, Davy et al. (2019). "Overview of the Fourth Social Media Mining for Health (SMM4H) Shared Tasks at ACL 2019". In: *Proceedings of the Fourth Social Media Mining for Health Applications (#SMM4H) Workshop & Shared Task* (cit. on pp. 70, 71).

Xue, Nianwen et al. (2015). "The CoNLL-2015 Shared Task on Shallow Discourse Parsing". In: *CoNLL* (cit. on p. 74).

Yang, Xiaoyu et al. (2020). "SemEval-2020 Task 5: Counterfactual Recognition". In: *SEMEVAL* (cit. on p. 69).

Yu, L. et al. (2017). "IJCNLP-2017 Task 2: Dimensional Sentiment Analysis for Chinese Phrases". In: *IJCNLP* (cit. on p. 75).

Yuret, Deniz, Aydin Han, and Zehra Turgut (July 2010). "SemEval-2010 Task 12: Parser Evaluation Using Textual Entailments". In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for

Computational Linguistics, pp. 51–56. URL: https://aclanthology.org/S10-1009 (cit. on p. 73).

Zadeh, Behrang Q. and Siegfried Handschuh (2014). “The ACL RD-TEC: A Dataset for Benchmarking Terminology Extraction and Classification in Computational Linguistics”. In: (cit. on p. 9).

Zadeh, Behrang Q. and Anne-Kathrin Schumann (2016). “The ACL RD-TEC 2.0: A Language Resource for Evaluating Term Extraction and Entity Recognition Methods”. In: *LREC* (cit. on pp. 9, 10).

A Dataset Preparation: Alternative Approaches

A.1 Generating Synthetic Data

We generated 130 synthetic samples based on the true samples classified as task descriptions using Markov chains (Akkaradamrongrat, Kachamas, and Sinthupinyo 2019). A Markov chain when applied towards language modeling shows the probabilities of sequences of words given a corpus. As described by Jurafsky and Martin[1] a Markov chain comprises a set of all states (word tokens), a probability matrix that shows the probability of moving from one word to another (the probability that two words will appear consecutively in a sequence), and the probability distribution for start states (the probability that a word will be the starting word in a sentence). Following is an example of an artificial task description generated following the algorithms described in Algorithms 3 and 4: *offensive message containing information extraction task aimed to their confidence is : word-sense disambiguation task required participating systems that message*. It is not totally coherent, but contains keyphrases that would be found in a real task description such as *information extraction task*, *word-sense disambiguation task*, and *required participating systems*.

One issue with this approach is there are only so many artificial samples that can be added. A dataset containing too many artificial samples is insufficiently represen-

[1] https://web.stanford.edu/~jurafsky/slp3/A.pdf

Algorithm 3 Markov Chain

```
procedure createMarkovChain(sentences: array of all task description sentences)
    sentences ← array of all task description sentences
    starts ← empty array
    transition_matrix ← dictionary where keys are unique set of tokens

    for each sentence in sentences do
        starts.append(sentence[0])
        previous ← empty string

        for each word in sentence do
            if previous is not empty then
                transition_matrix[previous].append(word)
            previous ← word
return starts, transition_matrix
```

Algorithm 4 Generate Synthetic Samples Using Markov Chain

```
procedure generateSample(starts, transition_matrix)
    ends ← [., ?, !]
    previous ← random_choice(starts)
    sequence ← [previous]

    while previous not in ends do
        word ← random_choice(transition_matrix[previous])
        sequence.append(word)
        previous ← word
return sequence
```

tative of real world data. Oversampling with artificially generated task descriptions was not enough to close the gap between positive and negative samples.

A.2 Automatic Downsampling

A.2.1 Downsampling Using Cosine Similarity

Downsampling is the removal of samples from a specific class and is a technique that can be used to balance datasets. One could perform downsampling by randomly removing a certain percentage of the samples. However, we wanted to minimize the information lost through downsampling. To do this, we removed samples that were highly similar to other samples. To measure similarity, we calculated the cosine similarity between vectorized sentences. The cosine of the angle between two vectors v_1 and v_2 with length n is the dot product over the product of their magnitudes:

$$cos(v_1, v_2) = \frac{\sum_{i=1}^{n} v_1 v_2}{\sqrt{\sum_{i=1}^{n} v_{1i}^2}\sqrt{\sum_{i=1}^{n} v_{2i}^2}} \tag{A.1}$$

We created a cosine similarity matrix that allows for the similarity between two vectors v_1 and v_2 to be looked up by finding v_2's column in row v_1 (see Algorithm 5). Then we set a threshold of 0.9; any sentence with a similarity greater than 0.9

Algorithm 5 Creating a Cosine Similarity Matrix

```
procedure GENERATESIMILARITYMATRIX(sentences: small subset of negative samples)
    sentences ← get_embeddings(sentences)
    similarity_matrix ← dictionary where keys are sentence embeddings

    for each $v_1$ in sentences do

        for each $v_2$ in sentences do

            if $v_1$ is not $v_2$ then
                cosine_similarity ← cosine($v_1, v_2$)
                similarity_matrix[$v_1$][$v_2$] = cosine_similarity
return similarity_matrix
```

with another sentence was removed (see Algorithm 6). One issue with this approach

Algorithm 6 Choosing Samples To Remove

```
procedure DOWNSAMPLE(threshold: similarity score at which or above $v_2$ should be removed, sentences: sentence embeddings, similarity_matrix)

    for each $v_1$ in sentences do

        for each $v_2$ in sentences do

            if $v_1 == v_2$ then
                continue

            if similarity_matrix[$v_1$][$v_2$] >= threshold then
                sentences.remove($v_2$)
return sentences
```

is that it is computationally expensive to calculate the cosine between every single sentence in a corpus of over 40,000 sentences. In our preliminary experiments, we only measured the cosine similarity between a subset of sentences to improve the compute time. Ultimately we abandoned this method because it was not possible to remove enough samples to balance the dataset.

A.2.2 Downsampling Using TextRank Algorithm

A third method we experimented with for downsampling the dataset was eliminating samples with a low ranking using the TextRank algorithm (see Algorithm 7). TextRank ranks the importance of sentences in a corpus by measuring the similarity between sentences, storing them in a similarity matrix, converting that matrix to a graph where vertices represent sentences and edges represent similarities, then calculating the sentence ranks based on each vertex's edges. We used the Python package NetworkX (Hagberg, Schult, and Swart 2008) to generate the graph and apply the TextRank algorithm to it. The hypothesis was that task description sentences are

Algorithm 7 Filtering With TextRank

```
procedure TEXTRANKFILTER(threshold: rank below which sentence should be
removed, sentences: sentence embeddings, similarity_matrix)
    similarity_graph ← convert_to_graph(similarity_matrix)
    scores ← page_rank(similarity_graph)

    for each score in scores do

        if score<threshold then
            scores.remove(score)
return scores
```

highly important sentences for most documents, so removing all sentences with a rank lower than a given threshold would remove a significantly higher proportion of negative samples than positive samples. However, we wanted to avoid eliminating any positive samples, because we already had so few. We ended up abandoning this method as well, because there was not a threshold large enough to remove a sufficient amount of negative samples while not removing positive samples.

B Software Used

This chapter provides the necessary details for each model we used so that our experiments could be reproduced.

Classifier	Hyperparameters
Basic Linear Models	
Complement Naive Bayes	Default
Logistic Regression	maximum iterations = 1000
SVM	Default
KNN	number of neighbors = 20, weight function = distance
Stochastic Gradient Descent	loss function = modified huber, max iterations = 1000, early stopping
Ensemble Models	
Random Forest	Default
Gradient Boosting	number of boosting stages = 1000, early stopping
XGBoost	number of gradient boosted trees = 1000
Neural Networks	
Neural Network	layers = 2, activation = sigmoid
Deep Neural Network	layers = 3, activation = relu, sigmoid
RNN	layers = 4, activation = sigmoid
RNN with LSTM	layers = 3, activation = sigmoid, dropout = 0.2
RNN with BiLSTM	layers = 3, activation = sigmoid, dropout = 0.2
CNN	layers = 10, activation = relu, sigmoid, dropout = 0.2, pool size = 4
Pretrained Models	
BERT	epochs = 2, 3, 4, batch size = 16, 32, learning rate = 2e-5, 3e-5, 5e-5, optimizer = AdamW
SciBERT	epochs = 2, 3, 4, batch size = 16, 32, learning rate = 2e-5, 3e-5, 5e-5, optimizer = AdamW

Table B.1: This table contains a list of all classifiers used in our training experiments and their hyperparameter settings.

Classifier	Hyperparameters
Basic Linear Models	
Complement Naive Bayes	sklearn.naive_bayes.ComplementNB
Logistic Regression	sklearn.linear_model.LogisticRegression
SVM	sklearn.svm.LinearSVC
KNN	sklearn.neighbors.KNeighborsClassifier
Stochastic Gradient Descent	sklearn.linear_model.SGDClassifier
Ensemble Models	
Random Forest	sklearn.ensemble.RandomForestClassifier
Gradient Boosting	sklearn.ensemble.GradientBoostingClassifier
XGBoost	xgboost.XGBClassifier
Neural Networks	
Neural Network	keras.Sequential, keras.layers.GlobalAveragePooling1D, keras.layers.Dense
Deep Neural Network	keras.Sequential, keras.layers.GlobalAveragePooling1D, keras.layers.Dense
RNN	keras.Sequential, keras.layers.SimpleRNN, keras.layers.Dense
RNN with LSTM	keras.Sequential, keras.layers.LSTM, keras.layers.Dropout, keras.layers.Dense
RNN with BiLSTM	keras.Sequential, keras.layers.Bidirectional, keras.layers.Dropout, keras.layers.Dense
CNN	keras.Sequential, keras.layers.Conv1D, keras.layers.Dropout, keras.layers.MaxPooling1D, keras.layers.GlobalMaxPooling1D, keras.layers.Dense
Pretrained Models	
BERT	transformers.BertForSequenceClassification using bert-base-uncased, bert-base-cased
SciBERT	transformers.BertForSequenceClassification using allenai/scibert_scivocab_uncased, allenai/scibert_scivocab_cased

Table B.2: This table contains a list of all classifiers used in our training experiments and the software used to implement them.

C Full Results

Provided in this chapter is the full set of training results. Table C.1 contains the results from the BERT experiments using the dataset with context and C.2 contains the results for the BERT experiments using the plain dataset. Table C.3 contains the results from the SciBERT experiments using the dataset with context and C.4 contains the results for the SciBERT experiments using the plain dataset. Tables C.5 and C.6 contain the results for the non-neural experiments trained on the dataset with context and the dataset without context, respectively. Table C.7 contains the neural results. All results are the mean of ten runs and the standard deviation is reported.

Confusion matrices are provided for the best model and encoding combination in Tables C.8 through C.21. Losses per epoch are reported in Figures C.1 through C.8 for the best BERT model and hyperparameter combination. Each graph of losses contains ten lines, one for each of the training runs. These lines are printed in different colors to keep them distinguishable.

settings	precision	recall	f1
BERT uncased			
epochs=2, batch size=16, lr=2e-05	0.66±0.15	0.6±0.16	0.61±0.13
epochs=2, batch size=16, lr=3e-05	0.68±0.14	0.61±0.13	0.63±0.11
epochs=2, batch size=16, lr=5e-05	0.61±0.27	0.54±0.26	0.56±0.25
epochs=2, batch size=32, lr=2e-05	0.55±0.34	0.39±0.28	0.41±0.28
epochs=2, batch size=32, lr=3e-05	0.68±0.16	0.64±0.13	0.64±0.08
epochs=2, batch size=32, lr=5e-05	0.66±0.17	0.58±0.13	0.61±0.13
epochs=3, batch size=16, lr=2e-05	0.67±0.16	0.64±0.16	0.64±0.14
epochs=3, batch size=16, lr=3e-05	0.67±0.13	0.66±0.13	0.66±0.1
epochs=3, batch size=16, lr=5e-05	0.67±0.14	0.66±0.19	0.65±0.14
epochs=3, batch size=32, lr=2e-05	0.67±0.16	0.65±0.17	0.65±0.15
epochs=3, batch size=32, lr=3e-05	0.69±0.15	0.66±0.13	0.66±0.1
epochs=3, batch size=32, lr=5e-05	0.66±0.17	0.65±0.14	0.64±0.12
epochs=4, batch size=16, lr=2e-05	0.65±0.14	0.72±0.14	0.68±0.12
epochs=4, batch size=16, lr=3e-05	0.65±0.15	0.71±0.14	0.67±0.12
epochs=4, batch size=16, lr=5e-05	0.64±0.17	0.69±0.14	0.65±0.13
epochs=4, batch size=32, lr=2e-05	0.64±0.16	0.69±0.11	0.66±0.12
epochs=4, batch size=32, lr=3e-05	0.63±0.15	0.69±0.15	0.65±0.12
epochs=4, batch size=32, lr=5e-05	0.63±0.16	0.67±0.15	0.64±0.13
BERT cased			
epochs=2, batch size=16, lr=2e-05	0.55±0.29	0.52±0.27	0.53±0.27
epochs=2, batch size=16, lr=3e-05	0.67±0.09	0.65±0.17	0.65±0.12
epochs=2, batch size=16, lr=5e-05	0.73±0.12	0.57±0.2	0.61±0.14
epochs=2, batch size=32, lr=2e-05	0.55±0.3	0.42±0.23	0.47±0.24
epochs=2, batch size=32, lr=3e-05	0.63±0.23	0.56±0.21	0.58±0.2
epochs=2, batch size=32, lr=5e-05	0.67±0.08	0.63±0.15	0.63±0.1
epochs=3, batch size=16, lr=2e-05	0.69±0.09	0.65±0.14	0.66±0.1
epochs=3, batch size=16, lr=3e-05	0.7±0.09	0.67±0.13	0.68±0.1
epochs=3, batch size=16, lr=5e-05	0.6±0.21	0.59±0.23	0.59±0.21
epochs=3, batch size=32, lr=2e-05	0.75±0.13	0.56±0.17	0.62±0.1
epochs=3, batch size=32, lr=3e-05	0.67±0.12	0.66±0.13	0.66±0.1
epochs=3, batch size=32, lr=5e-05	0.65±0.11	0.66±0.1	0.65±0.08
epochs=4, batch size=16, lr=2e-05	0.69±0.1	0.73±0.1	0.71±0.09
epochs=4, batch size=16, lr=3e-05	0.68±0.08	0.67±0.12	0.67±0.1
epochs=4, batch size=16, lr=5e-05	0.58±0.21	0.61±0.24	0.59±0.21
epochs=4, batch size=32, lr=2e-05	0.67±0.11	0.67±0.11	0.67±0.09
epochs=4, batch size=32, lr=3e-05	0.67±0.1	0.72±0.09	0.69±0.08
epochs=4, batch size=32, lr=5e-05	0.68±0.07	0.68±0.08	0.68±0.06

Table C.1: Training results using BERT trained on data with context.

settings	precision	recall	f1
BERT uncased			
epochs=2, batch size=16, lr=2e-05	0.65±0.12	0.65±0.12	0.64±0.08
epochs=2, batch size=16, lr=3e-05	0.64±0.09	0.66±0.13	0.64±0.07
epochs=2, batch size=16, lr=5e-05	0.63±0.07	0.7±0.13	0.65±0.06
epochs=2, batch size=32, lr=2e-05	0.59±0.23	0.58±0.22	0.57±0.21
epochs=2, batch size=32, lr=3e-05	0.66±0.14	0.69±0.12	0.66±0.09
epochs=2, batch size=32, lr=5e-05	0.66±0.13	0.67±0.11	0.65±0.07
epochs=3, batch size=16, lr=2e-05	0.63±0.11	0.7±0.1	0.65±0.08
epochs=3, batch size=16, lr=3e-05	0.63±0.09	0.7±0.1	0.66±0.07
epochs=3, batch size=16, lr=5e-05	0.62±0.1	0.69±0.13	0.65±0.09
epochs=3, batch size=32, lr=2e-05	0.61±0.1	0.69±0.09	0.64±0.08
epochs=3, batch size=32, lr=3e-05	0.63±0.09	0.7±0.09	0.65±0.06
epochs=3, batch size=32, lr=5e-05	0.63±0.08	0.7±0.1	0.66±0.08
epochs=4, batch size=16, lr=2e-05	0.6±0.06	0.71±0.11	0.65±0.07
epochs=4, batch size=16, lr=3e-05	0.64±0.08	0.68±0.13	0.66±0.09
epochs=4, batch size=16, lr=5e-05	0.65±0.08	0.66±0.12	0.65±0.08
epochs=4, batch size=32, lr=2e-05	0.64±0.1	0.68±0.11	0.66±0.09
epochs=4, batch size=32, lr=3e-05	0.65±0.09	0.68±0.12	0.66±0.09
epochs=4, batch size=32, lr=5e-05	0.61±0.1	0.66±0.15	0.63±0.09
BERT cased			
epochs=2, batch size=16, lr=2e-05	0.69±0.07	0.62±0.07	0.64±0.05
epochs=2, batch size=16, lr=3e-05	0.66±0.08	0.62±0.09	0.64±0.07
epochs=2, batch size=16, lr=5e-05	0.63±0.06	0.58±0.1	0.6±0.06
epochs=2, batch size=32, lr=2e-05	0.54±0.29	0.44±0.24	0.48±0.25
epochs=2, batch size=32, lr=3e-05	0.69±0.1	0.61±0.11	0.64±0.09
epochs=2, batch size=32, lr=5e-05	0.66±0.06	0.64±0.1	0.65±0.06
epochs=3, batch size=16, lr=2e-05	0.67±0.1	0.62±0.11	0.64±0.08
epochs=3, batch size=16, lr=3e-05	0.66±0.06	0.66±0.12	0.65±0.07
epochs=3, batch size=16, lr=5e-05	0.65±0.07	0.61±0.14	0.61±0.07
epochs=3, batch size=32, lr=2e-05	0.59±0.21	0.58±0.23	0.58±0.21
epochs=3, batch size=32, lr=3e-05	0.65±0.05	0.64±0.14	0.64±0.09
epochs=3, batch size=32, lr=5e-05	0.62±0.08	0.63±0.12	0.62±0.07
epochs=4, batch size=16, lr=2e-05	0.65±0.08	0.67±0.11	0.66±0.08
epochs=4, batch size=16, lr=3e-05	0.64±0.1	0.66±0.12	0.65±0.09
epochs=4, batch size=16, lr=5e-05	0.65±0.14	0.6±0.12	0.61±0.11
epochs=4, batch size=32, lr=2e-05	0.64±0.12	0.63±0.08	0.63±0.06
epochs=4, batch size=32, lr=3e-05	0.67±0.09	0.65±0.12	0.65±0.08
epochs=4, batch size=32, lr=5e-05	0.67±0.06	0.64±0.09	0.65±0.06

Table C.2: Training results using BERT trained on text data only.

settings	precision	recall	f1
SciBERT uncased			
epochs=2, batch size=16, lr=2e-05	0.69±0.07	0.72±0.09	0.7±0.05
epochs=2, batch size=16, lr=3e-05	0.7±0.07	0.7±0.07	0.69±0.04
epochs=2, batch size=16, lr=5e-05	0.69±0.06	0.65±0.1	0.67±0.08
epochs=2, batch size=32, lr=2e-05	0.66±0.09	0.7±0.09	0.67±0.06
epochs=2, batch size=32, lr=3e-05	0.67±0.08	0.68±0.13	0.66±0.07
epochs=2, batch size=32, lr=5e-05	0.71±0.05	0.65±0.13	0.67±0.06
epochs=3, batch size=16, lr=2e-05	0.69±0.04	0.7±0.11	0.69±0.05
epochs=3, batch size=16, lr=3e-05	0.69±0.03	0.73±0.12	0.71±0.06
epochs=3, batch size=16, lr=5e-05	0.68±0.04	0.68±0.12	0.68±0.06
epochs=3, batch size=32, lr=2e-05	0.68±0.06	0.73±0.12	0.7±0.06
epochs=3, batch size=32, lr=3e-05	0.69±0.04	0.71±0.13	0.69±0.07
epochs=3, batch size=32, lr=5e-05	0.71±0.06	0.68±0.13	0.69±0.08
epochs=4, batch size=16, lr=2e-05	0.68±0.05	0.72±0.1	0.7±0.05
epochs=4, batch size=16, lr=3e-05	0.67±0.05	0.69±0.12	0.67±0.06
epochs=4, batch size=16, lr=5e-05	0.68±0.05	0.67±0.14	0.67±0.07
epochs=4, batch size=32, lr=2e-05	0.69±0.06	0.72±0.11	0.7±0.06
epochs=4, batch size=32, lr=3e-05	0.7±0.06	0.7±0.1	0.7±0.06
epochs=4, batch size=32, lr=5e-05	0.69±0.04	0.68±0.09	0.68±0.04
SciBERT cased			
epochs=2, batch size=16, lr=2e-05	0.66±0.1	0.65±0.13	0.65±0.1
epochs=2, batch size=16, lr=3e-05	0.68±0.09	0.68±0.12	0.68±0.08
epochs=2, batch size=16, lr=5e-05	0.63±0.11	0.67±0.11	0.64±0.1
epochs=2, batch size=32, lr=2e-05	0.68±0.12	0.71±0.15	0.69±0.11
epochs=2, batch size=32, lr=3e-05	0.7±0.11	0.67±0.14	0.68±0.1
epochs=2, batch size=32, lr=5e-05	0.7±0.08	0.7±0.11	0.69±0.08
epochs=3, batch size=16, lr=2e-05	0.68±0.11	0.68±0.12	0.68±0.1
epochs=3, batch size=16, lr=3e-05	0.65±0.08	0.68±0.09	0.66±0.07
epochs=3, batch size=16, lr=5e-05	0.65±0.1	0.68±0.09	0.66±0.08
epochs=3, batch size=32, lr=2e-05	0.63±0.11	0.68±0.14	0.65±0.12
epochs=3, batch size=32, lr=3e-05	0.65±0.12	0.7±0.13	0.67±0.12
epochs=3, batch size=32, lr=5e-05	0.66±0.11	0.68±0.11	0.67±0.09
epochs=4, batch size=16, lr=2e-05	0.65±0.11	0.67±0.14	0.66±0.11
epochs=4, batch size=16, lr=3e-05	0.68±0.11	0.69±0.14	0.68±0.11
epochs=4, batch size=16, lr=5e-05	0.71±0.09	0.66±0.11	0.68±0.09
epochs=4, batch size=32, lr=2e-05	0.67±0.14	0.67±0.13	0.67±0.12
epochs=4, batch size=32, lr=3e-05	0.66±0.12	0.69±0.1	0.67±0.09
epochs=4, batch size=32, lr=5e-05	0.66±0.09	0.67±0.1	0.66±0.08

Table C.3: Training results using SciBERT trained on data with context.

settings	precision	recall	f1
SciBERT uncased			
epochs=2, batch size=16, lr=2e-05	0.65±0.09	0.65±0.09	0.65±0.07
epochs=2, batch size=16, lr=3e-05	0.69±0.09	0.66±0.08	0.67±0.06
epochs=2, batch size=16, lr=5e-05	0.68±0.1	0.63±0.09	0.64±0.06
epochs=2, batch size=32, lr=2e-05	0.68±0.11	0.67±0.09	0.67±0.08
epochs=2, batch size=32, lr=3e-05	0.67±0.11	0.66±0.09	0.66±0.06
epochs=2, batch size=32, lr=5e-05	0.68±0.1	0.67±0.11	0.67±0.08
epochs=3, batch size=16, lr=2e-05	0.68±0.1	0.66±0.1	0.67±0.09
epochs=3, batch size=16, lr=3e-05	0.7±0.12	0.67±0.08	0.67±0.08
epochs=3, batch size=16, lr=5e-05	0.66±0.11	0.64±0.07	0.64±0.07
epochs=3, batch size=32, lr=2e-05	0.67±0.11	0.68±0.1	0.67±0.08
epochs=3, batch size=32, lr=3e-05	0.67±0.09	0.68±0.08	0.67±0.06
epochs=3, batch size=32, lr=5e-05	0.68±0.11	0.66±0.11	0.67±0.09
epochs=4, batch size=16, lr=2e-05	0.65±0.11	0.66±0.08	0.65±0.08
epochs=4, batch size=16, lr=3e-05	0.68±0.1	0.64±0.07	0.66±0.06
epochs=4, batch size=16, lr=5e-05	0.66±0.11	0.6±0.06	0.63±0.06
epochs=4, batch size=32, lr=2e-05	0.65±0.1	0.67±0.07	0.66±0.06
epochs=4, batch size=32, lr=3e-05	0.65±0.11	0.66±0.1	0.65±0.08
epochs=4, batch size=32, lr=5e-05	0.67±0.09	0.63±0.1	0.64±0.07
SciBERT cased			
epochs=2, batch size=16, lr=2e-05	0.74±0.14	0.68±0.1	0.71±0.11
epochs=2, batch size=16, lr=3e-05	0.74±0.13	0.67±0.08	0.7±0.08
epochs=2, batch size=16, lr=5e-05	0.73±0.14	0.67±0.07	0.69±0.09
epochs=2, batch size=32, lr=2e-05	0.78±0.14	0.61±0.19	0.64±0.17
epochs=2, batch size=32, lr=3e-05	0.74±0.14	0.68±0.08	0.7±0.08
epochs=2, batch size=32, lr=5e-05	0.73±0.13	0.66±0.1	0.69±0.09
epochs=3, batch size=16, lr=2e-05	0.72±0.12	0.71±0.1	0.71±0.09
epochs=3, batch size=16, lr=3e-05	0.71±0.1	0.7±0.08	0.7±0.07
epochs=3, batch size=16, lr=5e-05	0.66±0.14	0.64±0.11	0.64±0.09
epochs=3, batch size=32, lr=2e-05	0.7±0.12	0.69±0.07	0.69±0.08
epochs=3, batch size=32, lr=3e-05	0.67±0.09	0.66±0.05	0.66±0.06
epochs=3, batch size=32, lr=5e-05	0.71±0.11	0.68±0.08	0.69±0.08
epochs=4, batch size=16, lr=2e-05	0.73±0.12	0.68±0.07	0.7±0.07
epochs=4, batch size=16, lr=3e-05	0.72±0.13	0.67±0.09	0.68±0.08
epochs=4, batch size=16, lr=5e-05	0.68±0.13	0.65±0.07	0.66±0.08
epochs=4, batch size=32, lr=2e-05	0.72±0.1	0.72±0.08	0.72±0.08
epochs=4, batch size=32, lr=3e-05	0.69±0.12	0.72±0.07	0.7±0.07
epochs=4, batch size=32, lr=5e-05	0.73±0.11	0.71±0.07	0.72±0.07

Table C.4: Training results using SciBERT trained on text data only.

model	one-hot	tf-idf	word ngrams	char ngrams	embeddings
naive bayes	0.41 ± 0.02	0.0 ± 0.0	0.34 ± 0.01	0.0 ± 0.0	N/A
	0.68 ± 0.01	0.0 ± 0.0	0.45 ± 0.0	0.0 ± 0.0	N/A
	0.51 ± 0.01	0.0 ± 0.0	0.39 ± 0.0	0.0 ± 0.0	N/A
logistic regression	0.73 ± 0.01	1.0 ± 0.0	0.0 ± 0.0	1.0 ± 0.0	N/A
	0.43 ± 0.02	0.03 ± 0.0	0.0 ± 0.0	0.03 ± 0.0	N/A
	0.54 ± 0.02	0.06 ± 0.0	0.0 ± 0.0	0.06 ± 0.0	N/A
svm	0.68 ± 0.0	0.87 ± 0.0	0.64 ± 0.02	0.69 ± 0.01	N/A
	0.55 ± 0.01	0.46 ± 0.01	0.22 ± 0.01	0.42 ± 0.01	N/A
	0.61 ± 0.01	0.6 ± 0.01	0.33 ± 0.01	0.52 ± 0.01	N/A
knn	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	1.0 ± 0.0	N/A
	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.1 ± 0.01	N/A
	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.18 ± 0.01	N/A
random forest	1.0 ± 0.0	1.0 ± 0.0	0.78 ± 0.08	1.0 ± 0.0	N/A
	0.09 ± 0.01	0.07 ± 0.01	0.23 ± 0.02	0.06 ± 0.02	N/A
	0.17 ± 0.03	0.12 ± 0.03	0.36 ± 0.03	0.1 ± 0.04	N/A
sgd	0.67 ± 0.09	0.76 ± 0.07	0.67 ± 0.09	0.64 ± 0.05	0.13 ± 0.07
	0.44 ± 0.08	0.5 ± 0.05	0.33 ± 0.04	0.48 ± 0.08	0.11 ± 0.11
	0.52 ± 0.05	0.6 ± 0.04	0.44 ± 0.04	0.55 ± 0.06	0.1 ± 0.05
gradient boost	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.0 ± 0.0
	0.13 ± 0.01	0.07 ± 0.0	0.19 ± 0.01	0.07 ± 0.03	0.0 ± 0.0
	0.23 ± 0.01	0.12 ± 0.01	0.32 ± 0.02	0.14 ± 0.04	0.0 ± 0.0
xgb	0.93 ± 0.01	0.73 ± 0.05	0.58 ± 0.01	0.71 ± 0.0	1.0 ± 0.0
	0.45 ± 0.01	0.17 ± 0.03	0.26 ± 0.01	0.16 ± 0.01	0.07 ± 0.0
	0.61 ± 0.01	0.28 ± 0.04	0.36 ± 0.02	0.27 ± 0.01	0.12 ± 0.01

Table C.5: Training results for dataset with context. For each model, row 1 of results is the precision, row 2 of results is the recall, and row 3 of results is the f1 score.

model	one-hot	tf-idf	word ngrams	char ngrams	embeddings
naive bayes	0.2 ± 0.01	0.05 ± 0.0	0.15 ± 0.0	0.0 ± 0.0	N/A
	0.37 ± 0.01	0.04 ± 0.0	0.66 ± 0.01	0.0 ± 0.0	N/A
	0.26 ± 0.01	0.04 ± 0.0	0.24 ± 0.0	0.0 ± 0.0	N/A
logistic regression	0.64 ± 0.0	1.0 ± 0.0	0.0 ± 0.0	1.0 ± 0.0	N/A
	0.27 ± 0.01	0.04 ± 0.0	0.0 ± 0.0	0.08 ± 0.0	N/A
	0.38 ± 0.01	0.08 ± 0.0	0.0 ± 0.0	0.14 ± 0.0	N/A
svm	0.49 ± 0.01	0.56 ± 0.0	0.5 ± 0.0	0.55 ± 0.0	N/A
	0.31 ± 0.01	0.2 ± 0.0	0.16 ± 0.0	0.22 ± 0.02	N/A
	0.38 ± 0.0	0.29 ± 0.0	0.24 ± 0.0	0.31 ± 0.01	N/A
knn	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N/A
	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N/A
	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N/A
random forest	0.7 ± 0.46	0.4 ± 0.49	0.63 ± 0.21	0.1 ± 0.3	N/A
	0.03 ± 0.02	0.02 ± 0.02	0.06 ± 0.02	0.0 ± 0.01	N/A
	0.05 ± 0.03	0.03 ± 0.04	0.11 ± 0.03	0.01 ± 0.02	N/A
sgd	0.46 ± 0.15	0.52 ± 0.04	0.54 ± 0.11	0.44 ± 0.06	0.03 ± 0.04
	0.41 ± 0.09	0.32 ± 0.05	0.21 ± 0.04	0.28 ± 0.08	0.03 ± 0.04
	0.41 ± 0.05	0.4 ± 0.05	0.3 ± 0.06	0.33 ± 0.07	0.03 ± 0.04
gradient boost	1.0 ± 0.0	0.8 ± 0.16	1.0 ± 0.0	0.85 ± 0.12	0.0 ± 0.0
	0.1 ± 0.02	0.09 ± 0.02	0.04 ± 0.0	0.1 ± 0.02	0.0 ± 0.0
	0.18 ± 0.03	0.17 ± 0.04	0.08 ± 0.0	0.18 ± 0.03	0.0 ± 0.0
xgb	0.74 ± 0.03	1.0 ± 0.0	0.24 ± 0.04	0.72 ± 0.1	0.4 ± 0.12
	0.39 ± 0.01	0.31 ± 0.01	0.12 ± 0.0	0.14 ± 0.02	0.04 ± 0.0
	0.51 ± 0.01	0.48 ± 0.01	0.16 ± 0.01	0.23 ± 0.03	0.07 ± 0.0

Table C.6: Training results for the plain dataset. For each model, row 1 of results is the precision, row 2 of results is the recall, and row 3 of results is the f1 score.

model	embeddings (trained on dataset with context)	embeddings (trained on dataset without context)
nn	0.0 ± 0.0	0.0 ± 0.0
	0.0 ± 0.0	0.0 ± 0.0
	0.0 ± 0.0	0.0 ± 0.0
dnn	0.0 ± 0.0	0.0 ± 0.0
	0.0 ± 0.0	0.0 ± 0.0
	0.0 ± 0.0	0.0 ± 0.0
rnn	0.34 ± 0.2	0.45 ± 0.18
	0.13 ± 0.1	0.21 ± 0.13
	0.17 ± 0.11	0.26 ± 0.11
cnn	0.67 ± 0.07	0.61 ± 0.36
	0.34 ± 0.12	0.1 ± 0.09
	0.43 ± 0.1	0.16 ± 0.13
lstm	0.74 ± 0.28	0.59 ± 0.24
	0.26 ± 0.25	0.22 ± 0.13
	0.33 ± 0.25	0.29 ± 0.14
bilstm	0.89 ± 0.12	0.54 ± 0.27
	0.24 ± 0.12	0.26 ± 0.16
	0.35 ± 0.12	0.34 ± 0.19

Table C.7: Training results for neural networks. For each model, row 1 of results is the precision, row 2 of results is the recall, and row 3 of results is the f1 score.

	Predicted		
True	10	5	Positive
	14	230	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and one hot encoding.

	Predicted		
True	5	9	Positive
	20	225	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and one hot encoding.

Table C.8: Confusion matrices for naive bayes

	Predicted		
True	9	12	Positive
	3	235	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and one hot encoding.

	Predicted		
True	14	38	Positive
	8	199	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and one hot encoding.

Table C.9: Confusion matrices for logistic regression

	Predicted		
True	17	14	Positive
	8	220	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and one hot encoding.

	Predicted		
True	14	31	Positive
	15	199	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and one hot encoding.

Table C.10: Confusion matrices for svm

	Predicted		
True	1	9	Positive
	0	249	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and tfidf char ngrams encoding.

	Predicted		
True	0	0	Positive
	0	259	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and tfidf char ngrams encoding.

Table C.11: Confusion matrices for knn

	Predicted		
True	8	27	Positive
	2	222	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and tfidf word ngrams encoding.

	Predicted		
True	5	78	Positive
	3	173	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and tfidf word ngrams encoding.

Table C.12: Confusion matrices for random forest

	Predicted		
True	16	16	Positive
	5	222	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and tfidf encoding.

	Predicted		
True	15	22	Positive
	18	205	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and one hot encoding.

Table C.13: Confusion matrices for sgd

	Predicted		
True	4	17	Positive
	0	238	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and tfidf word ngrams encoding.

	Predicted		
True	4	36	Positive
	0	219	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and tfidf char ngrams encoding.

Table C.14: Confusion matrices for gradient boost

	Predicted		
True	17	21	Positive
	1	220	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and one hot encoding.

	Predicted		
True	15	23	Positive
	5	215	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and one hot encoding.

Table C.15: Confusion matrices for XGBoost

True	Predicted		
	0	0	Positive
	0	259	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and embeddings encoding.

True	Predicted		
	0	0	Positive
	0	259	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and embeddings encoding.

Table C.16: Confusion matrices for nn

True	Predicted		
	0	0	Positive
	0	259	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and embeddings encoding.

True	Predicted		
	0	0	Positive
	0	259	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and embeddings encoding.

Table C.17: Confusion matrices for dnn

True	Predicted		
	3	20	Positive
	6	230	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and embeddings encoding.

True	Predicted		
	1	4	Positive
	1	253	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and embeddings encoding.

Table C.18: Confusion matrices for rnn

True	Predicted		
	9	17	Positive
	4	228	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and embeddings encoding.

True	Predicted		
	1	8	Positive
	1	249	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and embeddings encoding.

Table C.19: Confusion matrices for cnn

	Predicted		
True	6	17	Positive
	2	234	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and embeddings encoding.

	Predicted		
True	10	35	Positive
	7	207	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and embeddings encoding.

Table C.20: Confusion matrices for lstm

	Predicted		
True	6	19	Positive
	1	233	Negative
	Positive	Negative	

(a) Trained on dataset with contextual features and embeddings encoding.

	Predicted		
True	11	31	Positive
	9	207	Negative
	Positive	Negative	

(b) Trained on dataset without contextual features and embeddings encoding.

Table C.21: Confusion matrices for bilstm

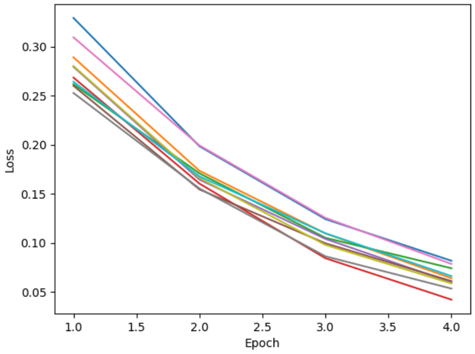


Figure C.1: Losses for uncased BERT trained on dataset with context

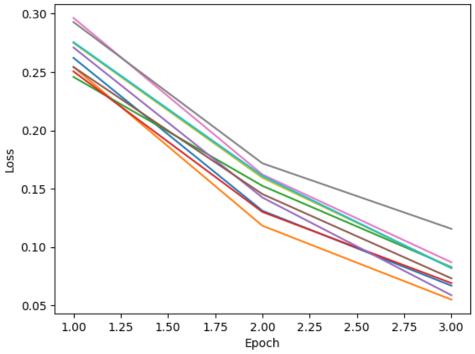


Figure C.2: Losses for uncased BERT trained on dataset without context

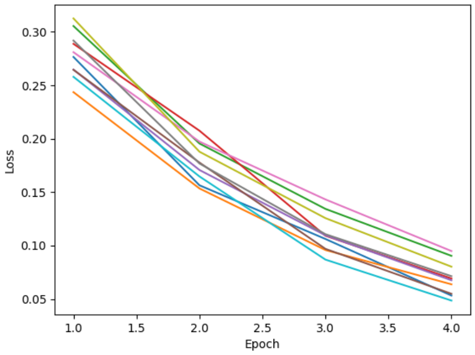


Figure C.3: Losses for cased BERT trained on dataset with context

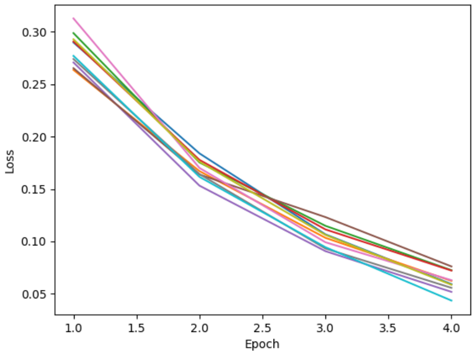


Figure C.4: Losses for cased BERT trained on dataset without context

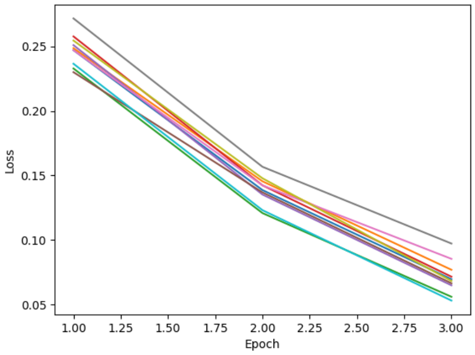


Figure C.5: Losses for uncased SciBERT trained on dataset with context

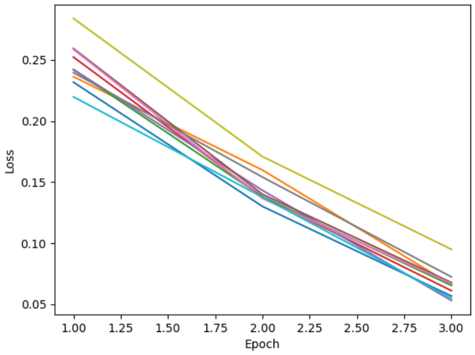


Figure C.6: Losses for uncased SciBERT trained on dataset without context

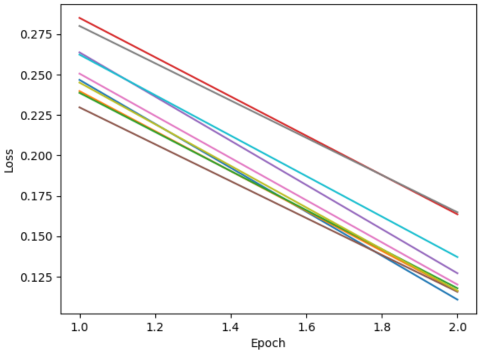


Figure C.7: Losses for cased SciBERT trained on dataset with context

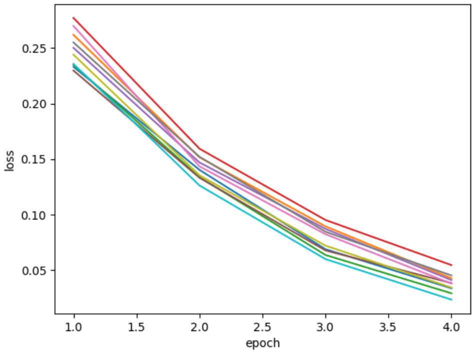


Figure C.8: Losses for cased SciBERT trained on dataset without context