ARTU / TU Wien and Artificial Researcher@ LongSumm 20

Alaa El-Ebshihy
TU Wien / Vienna, Austria
alaa.el-ebshihy@tuwien.ac.at

Annisa Maulida Ningtyas
TU Wien / Vienna, Austria
annisa.ningtyas@student.tuwien.ac.at

Linda Andersson
Artificial Researcher IT GmbH / Vienna, Austria
linda.andersson@artificialresearcher.com

Florina Piroi
TU Wien / Vienna, Austria
florina.piroi@tuwien.ac.at

Andreas Rauber
TU Wien / Vienna, Austria
andreas.rauber@tuwien.ac.at

Abstract

In this paper, we present our approach to solve the LongSumm 2020 Shared Task, at the 1st Workshop on Scholarly Document Processing. The objective of the long summaries task is to generate long summaries that cover salient information in scientific articles. The task is to generate abstractive and extractive summaries of a given scientific article. In the proposed approach, we are inspired by the concept of Argumentative Zoning (AZ) that defines the main rhetorical structure in scientific articles. We define two aspects that should be covered in scientific paper summary, namely *Claim/Method* and *Conclusion/Result* aspects. We use Solr index to expand the sentences of the paper abstract. We formulate each abstract sentence in a given publication as query to retrieve similar sentences from the text body of the document itself. We utilize a sentence selection algorithm described in previous literature to select sentences for the final summary that covers the two aforementioned aspects.

1 Introduction

Scientific publications differ in structure, format, and style when compared with other text works (e.g. news articles). As a result, summarizing scientific articles is a challenging task since exploiting known summarization techniques, like those employed by the MEAD system (Radev et al., 2004), that work well for general texts, cannot work well when applied to scientific articles.

Summarization of texts, scientific of not, has been of interest to researchers since the 1950s, when Peter Luhn published his paper "The automatic creation of literature abstracts". One of the most notable approaches to the summarization of scientific papers introduces the concept of *Argumentative Zoning* (Teufel et al., 1999) which refers to the examination of the argumentative status of sentences in scientific articles and their assignment to specific argumentative categories (i.e. zones). Building on this work, further research has been done to design automatic techniques for argumentative zoning (Teufel and Moens, 2002; Teufel et al., 2009; Liu, 2017).

A different and more recent approach to summarization makes use of citations to construct a summary of the main concepts and contributions in scientific articles (Qazvinian and Radev, 2008, 2010; Abu-Jbara and Radev, 2011). Starting from this research, since 2014, a series of pilot and shared tasks on summarization have been organized with some regularity. The CL-SciSumm Shared tasks (Jaidka et al., 2014, 2016, 2019; Chandrasekaran et al., 2019) require task participants to map citation sentences from a given scientific publication to reference sentences in the original articles, and generate a summary from those sentences using predefined facets.

This year, 2020, the CL-SciSumm Shared Task has introduced LongSumm as a new challenge (Chandrasekaran et al., 2020). In this challenge, given a scientific paper, it is required to generate the extractive and the abstractive summaries of the paper.

In this paper, we report on our approach to solving the LongSumm challenge of the CL-SciSumm Shared Task. We are proposing a summarization technique that builds on the concept of Argumentative Zoning (Teufel et al., 1999) and uses a Solr

Proceedings of the First Workshop on Scholarly Document Processing, pages 310–317

https://doi.org/10.18653/v1/P17

index to expand the abstract of a scientific article to obtain a summary of it. In this approach we extract the main aspects of the scientific articles into a summary by defining zones of interest in the article: claims, methods, results and conclusions. From our own experience we have observed that these aspects are best to form an informative summary for the researcher.

The rest of the paper is organized as follows: Section 2 discusses the components of the processing pipeline that constitute our approach in more detail. In Section 3 we describe our experimental settings and evaluation results for individual components of our pipeline. Finally, we discuss the conclusion and potential direction of future work in Section 4.

2 Approach

In a nutshell, our approach to solve the LongSumm Shared Task is to extract an article summary by expanding its abstract. That is, for each sentence in the article's abstract we extract relevant paragraphs from the same article employing a combination of Language Model-based retrieval and then classification to find similar paragraphs out of which, in a final step, we select certain sentences to be part of the article summary.

Figure 1 shows an overview of our proposed approach. In a first phase, we convert the PDF papers to an XML format using the GROBID[1] PDF parser (the box marked with 1 in figure 1). The output of this step is split into the abstract of the article and the article body. The latter is processed by a rule-based annotation module (2) which will assign specific category labels to the individual paragraphs. The paragraphs and their category annotations are fed into a Solr[2] index (3). The abstract sentences are classified into two categories, *Claim/Method* and *Conclusion/Result* (block $5a$ in Figure 1).

In the next phase, the abstract sentences are sent as queries to the Solr index in order to find paragraphs that are similar to them (4). We note that the abstract sentence annotations are not used in this phase. These annotations will be used later in the last phase of our summarization process. The $\langle sentence, paragraph \rangle$ pairs that are the output of the Solr retrieval step are now passed to a classifier which will decide if the two are similar to each other or not ($5b$). Pairs that are classified as similar are used to extract the final paper summary by applying the sentence selection algorithm mentioned in (Abu-Jbara and Radev, 2011) (6).

The details for each module are discussed in the following subsections.

2.1 PDF Parsing with GROBID

In order to process scientific publications stored as PDF files, we need to convert them to a representation to which we can later apply further text processing methods. We have explored various PDF parsers and we settled to use GROBID[3] as a tool to process scientific articles. GROBID takes as input the PDF file and outputs its extracted content in an XML format. We have chosen GROBID for several reasons: i) it divides the paper to paragraphs; ii) it sorts out special, non-visible characters (e.g. line breaks "`\n`"); and iii) it identifies figures and tables and places information about their occurrence in the PDF file in special XML tags.

The XML output of the GROBID module (box 1 in Figure 1) differentiates between the abstract text and the text in the article's body, which will be treated separately in the next steps of our pipeline (see the next sections).

2.2 Rule-Based Annotation

Taking from the previous step the XML representation of the article body, we want to assign different discourse categories to its paragraphs. We define five categories, or discourse sections: *Introduction*, *Background*, *Method*, *Conclusion* and *Result*. Each paragraph in the article body will be assigned to one of these discourse sections. The category assignment is done by employing a rule-based annotation module (box 2 in Figure 1), where the rules were manually created by the authors of this paper. At the end of this processing step, the paragraphs of the article body will each have a discourse section label attached to it, in addition to the paragraph meta-data delivered by the GROBID tool. Figure 2 shows a sample of the rule-based annotation step output, where the paragraph is categorized as "Method" with the $\langle discourse\_section \rangle$ XML element.

The paragraphs annotated by this module are now fed into a Solr index (box 3), together with the meta-data provided by the GROBID tool.

[1] https://grobid.readthedocs.io
[2] https://lucene.apache.org/solr/
[3] In our implementation, we use GROBID as a tool for PDF parsing to convert PDF articles to an XML format

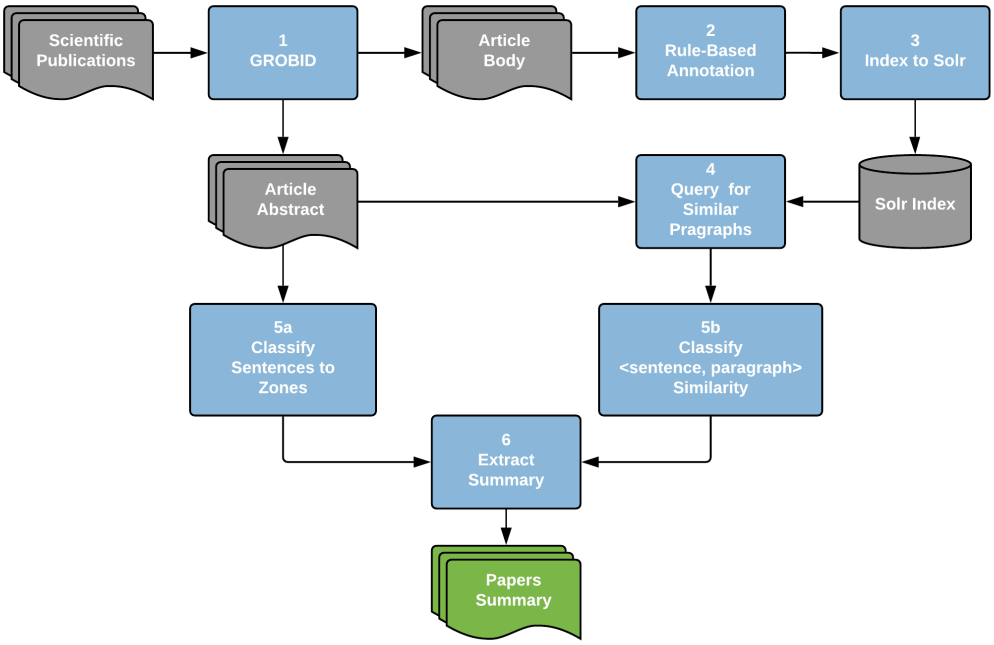

Figure 1: Overview of the ARTU/TU Wien approach to the LongSumm challenge.

```
<field paragraph_id="16">
  <annotation>
    <citations>
      <citation target="#b7"/>
    </citations>
    <discourse_section section_id="METHOD" section_name="Previous Work"/>
    <lexicon entity="['Figure 1']"/>
  </annotation>
  <text>
    Thus, these definitions only allow histories that would occur in a system using long/short read/write item/
    predicate locks. Since locking serializes transactions by preventing certain situations (e.g., two concurrent
    transactions both modifying the same object), we refer to this approach as the preventative approach. Figure 1
    summarizes the isolation levels as defined in [8] and relates them to a lock-based implementation. Thus the READ
    UNCOMMITTED level proscribes P0; READ COM-MITTED proscribes P0 and P1; the REPEATABLE READ level proscribes P0
    -P2; and SERIALIZABLE proscribes P0 -P3.
  </text>
</field>
```

Figure 2: Output sample of the rule-based annotation module.

2.3 Classification of *Abstract Sentence, Paragraph Text* Pairs

One of the key ideas of our approach is to find a summary of an article by expanding on the abstract sentences. To this end, we have to first find the paragraphs in the article body that are related to the sentences in the abstract. Therefore, we use the sentences in the abstract to formulate queries and send them to the Solr index to get paragraphs similar to the abstract sentence (box 4 in Figure 1). To further filter out paragraphs in the article body that may not be useful in extracting the final article summary, we send the obtained pairs of $\langle abstract\_sentence, paragraph \rangle$ to a classifier (box 5 in Figure 1) that will decide whether the two pair components are relevant to each other or not (box 5*b*).

To train the classifier used in this phase we created a training set by sampling 330 pairs of abstract sentences and retrieved paragraphs, and by manually annotating them as *relevant* or *not relevant*. The annotation made use of a simple GUI implemented by us (see Figure 3) where the annotator could compare the abstract sentence to the paragraph text part of its pair and decide whether the paragraph is relevant to the abstract sentence or not. Using this manually annotated set, we train a Random Forest classifier to build the classification model. Important features of this classifier are the similarity score between the abstract sentences and paragraph texts given by Solr, as well as the discourse sections annotations (see Section 2.2).

The evaluation of this classification module is presented later, in Section 3.3.

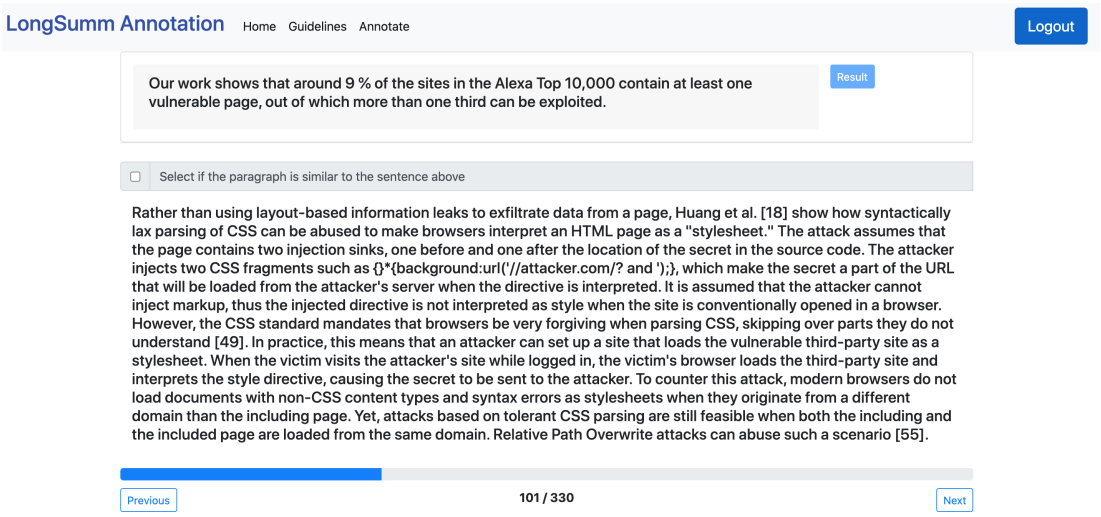

Figure 3: Paragraph to sentence relevance annotation sample.

2.4 Abstract sentences Classification

To make sure that a summary of the article equally covers the discourse sections defined in Section 2.2, we have to attach the same categories to the abstract sentences.

For this phase we decided, again, to use a classification model that would annotate the abstract sentences with one of the discourse sections chosen by us (box $5a$ in Figure 1). The training data to build this classification model has been collected by running an annotation task where we sampled 50 abstracts from the LongSumm data set and split them by sentences. The annotators had to decide which discourse category each abstract sentence falls into. For this task, the annotators had to select one of four categories: *Claim*, *Method*, *Conclusion* and *Result*. The *Introduction* category was left out as we consider it not to be relevant when constructing a summary of a scientific article using our approach. We used the BRAT tool (Stenetorp et al., 2012) to collect the annotations (see Figure 4). Because the inter-annotator agreement score turned out to be unsatisfactory, we have analyzed in more detail the labels assigned by the annotators to find a reason for the low score. We have found that there were disagreements between the *Claim* and the *Method* categories and between the *Conclusion* and the *Result* categories as the annotators could not clearly differentiate, for example, whether a sentence should be considered part of the *Conclusion* discourse or of the *Result* discourse. Therefore we took the decision to combine the *Claim* and the *Method* categories into one category, *Claim/Method*, and the *Conclusion* and the *Result* categories into the *Conclusion/Result* category.

From the sentences manually classified into one of these latter two categories, we extract key phrases and build a lexicon, where each phrase in the lexicon has two weights attached to it, one for each of the two categories. These weights are based on the key phrase occurrence frequency in the two categories.

The classifier trained with the manually annotated data just described will assign categories to abstract sentences by summing the weights of the lexicon key phrases that are part of the sentence, and choosing the category with the highest sum. In case of a tie, the sentence is classified as a *Method/Claim* sentence.

2.5 Summary Extractor

Having completed the parsing, indexing, and classifying paragraphs and sentences, we are now in the position of creating an article sumary (box 6 in Figure 1). The implementation of this module follows the methodology described in (Abu-Jbara and Radev, 2011) where, in our implementation, we consider the abstract sentences to be the target paper sentences, and the sentences similar to the abstract sentences (as per Section 2.3) to be the implicit citation sentences.

In our final processing step, we first split the paragraphs similar to the abstract sentences into single sentences. Then, we define a cluster for each abstract sentence which collects all single sentences split from all paragraphs similar to the abstract sentence. Lastly, for each sentence in the cluster, we calculate its LexRank score (Erkan and

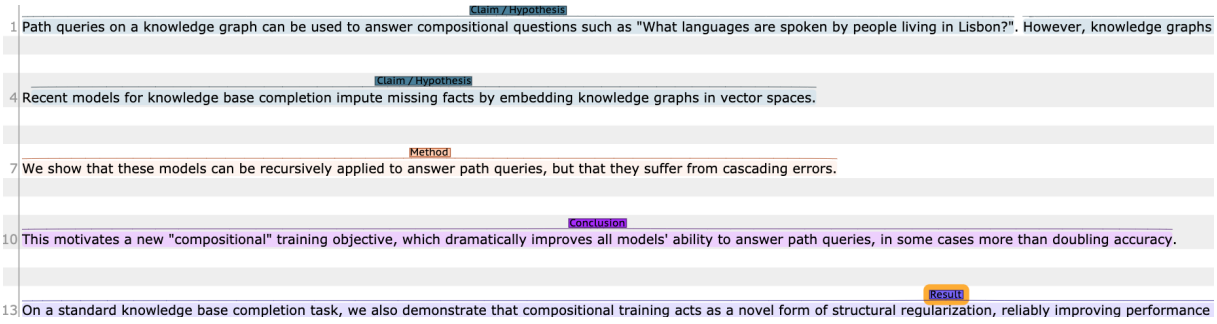

Figure 4: Abstract sentences annotation sample.

Radev, 2004).

To construct the final summary, similarly to (Abu-Jbara and Radev, 2011), sentences are added to the summary in the order of their category, the size of their clusters, then their LexRank values. The first category in our order is *Claim/Method*. Clusters within the same category are ordered by their size (i.e. the number of sentences in each cluster) and finally the sentences in each cluster are ordered by their LexRank values in descending order.

According to the task description, the maximum summary size allowed is 600 words. We extract the sentences from the clusters in a round-robin sequence. In other words, we extract the first ranked sentence in the largest cluster in the *Claim/Method* category then the first ranked sentence in the largest cluster in the *Result/Conclusion* category then the first ranked sentence in the second largest cluster in the *Claim/Method* category and so on. This process continues until either the maximum size of the summary is reached or we run out of sentences in the clusters.

3 Experiments and Result

In this section, we discuss the experimental setup and the evaluation results for the annotation tasks and the classification modules described in the previous section. We close the section with an evaluation of the final summarization step in our approach.

3.1 Abstract Sentence Annotation

Two annotators have provided the category labels for the abstract sentence annotation task discussed earlier in section 2.4. To measure the effectiveness of the annotation task, we computed Cohen's kappa coefficient of agreement between annotators, (κ) (Cohen, 1960). For this specific annotation task, the coefficient (κ) was **0.613**. We considered this agreement value to be acceptable for this task.

3.2 Abstract Sentence Classification

As described in section 2.4, we use a lexicon based approach to assign one category to each sentence in the abstract. To evaluate this module, we have measured the Precision, Recall, F-measure and the Accuracy of our approach on the annotated data set from previous step. Table 1, shows the evaluation results of Precision, Recall and F-measure for each category. The Accuracy of the this approach is **81.5%**.

	Claim /Method	**Conclusion /Result**	**Average**
Precision	0.875	0.613	0.744
Recall	0.883	0.597	0.739
F-measure	0.879	0.605	0.742

Table 1: Abstract sentences classification evaluation results.

3.3 *Abstract Sentence, Paragraph Text* Pair Annotation

Two annotators have participated in the annotation task described in section 2.3. The value of the inter annotation agreement, using the Cohen's kappa coefficient (κ) (Cohen, 1960), is **0.361**. We notice that the inter annotation agreement is rather low. Therefore, as a future work, we plan to analyze the reasons for low agreement and design a second annotation round after clarifying the task.

3.4 *Abstract Sentence, Paragraph Text* Pair Classification

We use the data generated from the annotation task, as discussed earlier in section 2.3, to build a classification model that determines whether, for an abstract sentence the paired paragraph text is relevant or not. We split the training data into 70%

training and 30% validation sets to determine the best classification model and its parameters.

We have tested different classification models including: linear SVM, Logistic Regression and Random Forest algorithms. We decided to use the model given by the Random Forest algorithm since it gives the best evaluation results (see Table 2).

	Similar	Not Similar	Average
Precision	0.755	0.474	0.624
Recall	0.861	0.333	0.597
F-measure	0.815	0.391	0.604

Table 2: Abstract sentence paragraph random forest classification evaluation results.

3.5 Final Summary Evaluation

Our main goal was to develop an extractive summarizer algorithm that starts from abstract sentences pre-labeled as *Method/Claims* and *Result/Conclusions*, and thereafter extracts relevant sentences from the remainder of a given publication. In order to quickly give the user the essence of the paper, a secondary interest of our work was to see if this method is applicable to abstractive summarization as well. We compared the summaries obtained by our approach with both the extractive and the abstractive summaries available in the task repository[4]. We used the provided evaluation script (Chandrasekaran et al., 2020) to evaluate the summaries given by our approach. As seen in Table 3 the method is more adapted to be extractive summarization.

In Table 4, we compare our method with the other submitted runs in the 2020 Longsumm task. We see that our team (ARTU) holds the place 7-9 depending on the descending order of different rouge measurements.

4 Conclusion and Future work

We have presented an approach to solve the LongSumm'20 Shared Task. Our aim was to extract the essence of a scientific paper in order to give the reader a quick overview of its central aspects: the authors' hypothesis (the claim), the method used test the hypothesis and the outcome of the experiment.

In (Saggion and Lapalme, 2002), the authors propose to create Indicative-Informative summaries by observing how professional abstractors generated scientific summaries, and thereafter manually align the professional abstract with the source data. An Indicative Abstract is composed of the topic of the document, i.e. the essence of the paper. Since we did not have access to professional abstractors, we have decided to use different approach to derive the essence of a paper. We assumed that the existing abstract is a very short summary focusing on the *why*, the *how*, and on the general outcome of a scientific work, which we organize into *Claims*, *Methods*, *Results* and *Conclusions* using the Argumentative Zoning schema.

In a first run we asked students to label abstract sentences with these four categories. In order to align the labeled abstract sentences with the body text of an article, we have used a Language Model where each abstract sentence is converted into a query and the system retrieves similar sentences from the body text of the same article. In a second annotation task we have asked students to assess if the retrieved sentence is relevant to the abstract, and if it could be considered to be assigned to same category as the abstract sentence.

Finally, we utilize the sentence selection method described by (Abu-Jbara and Radev, 2011) to form the final summary.

We have not yet explored the full potential of the algorithm due to two main factors:

- Limited training data.
- Ambiguous annotation due to the low inter annotator agreement score in the second annotation task.

As future work, we plan to address these these issues by clarifying the annotation guidelines and test the anotators' performance accross different scientific fields. Other future work is to investigate how changes in the round-robin sentence selection algorithm may impact the quality of the extracted summaries. More specifically, we may look at recomputing the scores of the sentences in the clusters after each sentence selection, where scores could reflect not only relevance to the abstract sentence, but also novelty compared to the other sentences in the same cluster, as well as the clusters of other abstract sentences of the same category.

[4]https://github.com/guyfe/LongSumm#training-data Maximum summary size for an article was of 600 words.

	rouge1_f	rouge1_r	rouge2_f	rouge2_r	rougeL_f	rougeL_r
Abstractive summaries	0.411	0.484	0.108	0.125	0.159	0.198
Extractive summaries	0.548	0.529	0.275	0.265	0.230	0.221

Table 3: The evaluation results of the proposed approach using rouge metrics against the abstractive and extractive data sets.

Participant Team	rouge1_f	rouge1_r	rouge2_f	rouge2_r	rougeL_f	rougeL_r
Summaformers	0.4938	0.439	0.1686	0.1498	0.2138	0.1898
GUIR	0.5311	0.546	0.1677	0.1728	0.2034	0.209
wing	0.5058	0.5116	0.1662	0.1675	0.205	0.2066
IIITBH-IITP	0.4903	0.4984	0.1574	0.16	0.2046	0.208
Auth-Team	0.5011	0.4693	0.1537	0.1423	0.1959	0.1818
CIST_BUPT	0.4899	0.4974	0.1506	0.1522	0.2013	0.2039
ARTU	**0.4803**	**0.4678**	**0.1476**	**0.1428**	**0.1804**	**0.1743**
IITP-AI-NLP-ML	0.4646	0.4743	0.1461	0.1486	0.1958	0.1995
Monash-Summ	0.4916	0.4935	0.128	0.1276	0.1831	0.1833
mummert	0.1232	0.0683	0.063	0.0349	0.0989	0.0549

Table 4: Submission results of all participating teams from the Longsumm 2020 leaderboard. Our team is **ARTU**.

References

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 500–509.

Muthu Kumar Chandrasekaran, Guy Feigenblat, Eduard Hovy, Abhilasha Ravichander, Michal Shmueli-Scheuer, and Anita De Waard. 2020. Overview and insights from scientific document summarization shared tasks 2020: Cl-scisumm, laysumm and longsumm. In *Proceedings of the First Workshop on Scholarly Document Processing (SDP 2020)*.

Muthu Kumar Chandrasekaran, Michihiro Yasunaga, Dragomir Radev, Dayne Freitag, and Min-Yen Kan. 2019. Overview and results: Cl-scisumm shared task 2019. *arXiv preprint arXiv:1907.09854*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Beatriz Fisas Elizalde, Rahul Jha, Christopher Jones, Min-Yen Kan, Ankur Khanna, Diego Molla-Aliod, Dragomir R Radev, Francesco Ronzano, et al. 2014. The computational linguistics summarization pilot task. In *Proceedings of Text Ananlysis Conference, Gaithersburg, USA*.

Kokil Jaidka, Muthu Kumar Chandrasekaran, Sajal Rustagi, and Min-Yen Kan. 2016. Overview of the cl-scisumm 2016 shared task. In *Proceedings of the joint workshop on bibliometric-enhanced information retrieval and natural language processing for digital libraries (BIRNDL)*, pages 93–102.

Kokil Jaidka, Michihiro Yasunaga, Muthu Kumar Chandrasekaran, Dragomir Radev, and Min-Yen Kan. 2019. The cl-scisumm shared task 2018: Results and key insights. *arXiv preprint arXiv:1909.00764*.

Haixia Liu. 2017. Automatic argumentative-zoning using word2vec. *CoRR*, abs/1703.10152.

Vahed Qazvinian and Dragomir Radev. 2010. Identifying non-explicit citing sentences for citation-based summarization. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 555–564.

Vahed Qazvinian and Dragomir R Radev. 2008. Scientific paper summarization using citation summary networks. *arXiv preprint arXiv:0807.1560*.

Dragomir R. Radev, T. Allison, Sasha Blair-Goldensohn, John Blitzer, A. elebi, S. Dimitrov, E. Drábek, A. Hakim, W. Lam, D. Liu, Jahna Otterbacher, H. Qi, Horacio Saggion, S. Teufel, M. Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD - A Platform for Multidocument Multilingual Text Summarization. In *LREC*.

Horacio Saggion and Guy Lapalme. 2002. Generating indicative-informative summaries with SumUM. *Computational Linguistics*, 28(4):497–526.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Junichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted

text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational linguistics*, 28(4):409–445.

Simone Teufel, Advaith Siddharthan, and Colin Batchelor. 2009. Towards domain-independent argumentative zoning: Evidence from chemistry and computational linguistics. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 1493–1502.

Simone Teufel et al. 1999. *Argumentative zoning: Information extraction from scientific text*. Ph.D. thesis, Citeseer.