Project Report - Auto Citations

Author: C.J. DuHamel and Arian Houshmand

1 Introduction

Citations are essential to academia, facilitating collaboration, aknowledging prior work, and providing a basis for further research. However, managing citations can be very time-consuming, requiring the writer to manually format, organize, and place citations within their work, taking aluable time away from the actual writing and research that goes into the paper. To address this challenge, we created an Auto Citation tool that automates the placing of in-text citations given a list of referenced works. We aim to streamline the citation process, reducing workload on researchers and reducing the spread of plaigerism by ensuring proper attribution.

1.1 Problem Details

The goal of this project is to develop a tool that automatically inserts in-text citations into a document based on a provided bibliography/collection of referenced works. Ideally, the tool should be able to analyze the un-cited document and identify the optimal placements for in-text citations, ensuring that they are contextually relevant to the content being discussed.

We will focus on a subset of the full functionality that would be desired for a complete tool. Specifically, we will target paragraph level citation placement, identifying for each reference the most relevant paragraph in the document to place the citation. To do this, we will use various natural language processing (NLP) techniques, specifically leveraging transformer-based models to utilize modern methods of semantic understanding of text.

2 Data Sources

2.1 CiteWorth

CiteWorth [1] is a dataset of academic papers compiled by researches at the University of Copenhagen. The dataset contains 1.2 million cleaned sentences from academic papers, labeled with their 'cite-worthiness', or whether or not that sentence contains an in-text citation. The dataset is not only labeled with cite-worthiness, but also annotated heavily with details about the cited papers and various representations of the sentences themselves.

Each entry in the dataset represents a paragraph from an academic paper, containing a variety of fields, including:

- **paper_id**: The unique identifier for the paper from which the paragraph was extracted.
- **original_text**: The actual text of the paragraph.
- **section_title**: The title of the section in which the paragraph appears.
- **samples**: A list of sentences within the paragraph, each with its own set of annotations.
  - **text**: The cleaned sentence text without the citations.
  - **label**: A binary label indicating whether the sentence contains a citation (1) or not (0).
  - **original_text**: The original sentence text with citations included.
  - **ref_ids**: A list of reference IDs corresponding to the citations present in the sentence. These are the 'Corpus Ids' used to uniquely identify cited papers in the Semantic Scholar Open Research Corpus (S2ORC) dataset.
  - **citation_text**: List of citation strings as they appear in the original text.

2.2 Semantic Scholar Open Research Corpus

The Semantic Scholar Open Research Corpus (S2ORC) [2] is a large dataset of academic papers (400+ GB) compiled by researchers at the Allen Institute for AI. The dataset contains over 81 million papers, including their full text and metadata for natural language processing tasks. Each paper in the dataset is assigned a unique 'Corpus Id', which is used to identify and reference papers within the dataset.

The dataset is organized in JSONL format, with each line representing a single paper. Each paper entry contains various fields, including:

- **paper_id**: The unique identifier for the paper.
- **metadata**: Metadata about the paper, including title, authors, abstract, and publication venue.
- **body (or content)**: All relevant information from the paper, including the full text, figure and table info/captions, and any included annotations (byte offsets, in-text citation locations, etc.). This section contains a lot of information and is losely structured. It soemtimes contains nested fields, other times the fields are called different names, and sometimes the information is missing altogether. However, the full text of the paper is ussually in a consistent format.
- **bibliography**: A list of references cited in the paper, each with its own set of metadata, including the 'matched_paper_id', which is the id for the referenced paper in the Semantic Scholar Academic Graph (S2AG) [3], a superset of the S2ORC that contains significantly more papers, but not all of them contain text. This also contains a section called 'annotations', which contains the matched paper ids and byte offsets for bibliography entries.

3 System Design Overview

Our system is comprised of several key components that make up the pipeline for auto placing in text citations. We begin with the data sources mentioned above, which will be the basis for our project. The dataset retrieval is a major step in the process. Once the data is retrieved, we create our own local database to facilitate faster lookups during dataset building. Next, we build our training dataset from the combined CiteWorth and S2ORC datasets. This dataset will then be used to train and evaluate our model. Finally, we will utilize the model for inference on new complete documents to place citations automatically.

4 Dataset Creation and Design

Building a high-quality dataset is crucial to model performance. Ideally, we want to take advantage of the large datasets available, since we have to handle them in their entirety anyway. However, we need to ensure that the data we have is both relevant and mostly contained within the S2ORC dataset, since not all referenced papers have full text available (i.e. they are only in the S2AG dataset).

4.1 Data Retrieval

Initially, we had assumed that the S2ORC dataset could be accessed via API calls to retrieve individual papers and their data. We were, however, mistaken. Thus, we are required to download the full S2ORC dataset in bulk and extract the relevant papers using their 'Corpus Id's. This is significantly more complex than we had originally anticipated, and much time has been spent just designing the downloader.

The general algorithm we wrote for downloading the dataset is as follows:

1. Call the Semantic Scholar (S2) API to retrieve the most recent release of the S2ORC dataset.
2. Call the S2 API again to retrieve the presigned download links for the dataset files.
3. Build a CSV by extracting the "shard_id" for each download link and associating it with the corresponding presigned URL. Add additonal columns to the CSV to track download status and local file paths. This is to facilitate resuming downloads in case of interruptions.

4. Download the dataset files using the presigned links. For each link (i.e. row in the CSV), do the following:
   (a) Check the download status column in the CSV to see if the file has already been downloaded. If it has, skip to the next row.
   (b) If the file has not been downloaded, download it and save it to the specified local file path.
   (c) After a successful download, update the download status column in the CSV to indicate that the file has been downloaded.
   (d) The presigned URLs have a limited validity period, which is much less than the time required to download the whole dataset. If a link has expired, refresh the links by repeating step 2 and updating the CSV with the new links. Then, retry downloading the expired file.

Retrieving the CiteWorth dataset was significantly simpler, as it is available directly via Hugging Face [4]. We simply downloaded the dataset via their API and pandas [5] for easy exploration and manipulation.

4.2 Dataset Building

We begin with a mapping between paper corpus_ids to the file paths in the S2ORC dataset that we created for the last checkpoint. This mapping is stored in a CSV file where each row contains a corpus_id and the corresponding file path (shard) to the S2ORC JSON file. We then read in the CiteWorth dataset and group by the corpus_id to create a list of all the sections in the dataset for each paper. We do the following for each paper in the CiteWorth dataset:

1. If the corpus_id does not exist in the S2ORC mapping, we skip the paper.
2. Fetch the S2ORC JSON object using the file path from the mapping.
3. For each section in the CiteWorth dataset for the paper:
   (a) Identify the bibliography reference IDs that are present in that section
   (b) For each reference ID, locate the corresponding corpus_id in the S2ORC JSON object's bibliography.
   (c) Get each referenced paper's JSON object using the corpus_id to shard mapping.
4. Reiterate through each sample (sentence) in each section:
   (a) If there are no references in the sample, skip it.
   (b) For each reference, create a new entry in the structured dataset that contains:
      - The original paper's corpus_id
      - The sentence text from the original paper
      - The reference paper's corpus_id
      - The reference paper's title
      - The reference paper's authors
      - The first 500 characters of the reference paper's text (either abstract or introduction)
      - A label indicating whether the reference paper should be cited in that sentence (1 for cite, 0 for no cite)
5. Write the new json object to a JSONL file, along with writing the original paper's corpus_id to a .log file to allow for the script to be resumed if interrupted.

The process creates a dataset of sentence - reference pairs, with a label indicating whether the reference should be cited in that sentence. This structured dataset can then be used for training and evaluating our auto citation model.

4.3 Example Entries

Here are some example entries from the structured dataset:


```
{
    "original_paper_id": 386802,
    "sentence": "Clinical antigen-specific studies however, have so far failed to
    ↪  show the desired efficacy despite their good safety profile.",
    "ref_paper_id": 713469,
    "ref_paper_title": "Non-antigenic and antigenic interventions in type 1
    ↪  diabetes",
    "ref_paper_authors": "Anna Ryd\u00e9n, J. Wesley, K. Coppieters, M. V. von
    ↪  Herrath",
    "ref_paper_text": "\nIntroduction\n\nDiabetes mellitus describes the outcome of
    ↪  several metabolic disorders characterized by hyperglycemia, including type 1
    ↪  diabetes (T1D). In the context of T1D, hyperglycemia typically results from
    ↪  an immunologically driven assault on the \u03b2-cells-the insulin-producing
    ↪  cells of the pancreas-leading to insufficient insulin secretion. 4
    ↪  \u03b2-Cell destruction is often rapid in young subjects but more prolonged
    ↪  in adults; this rate, however, is subject to great variability between
    ↪  individua",
    "label": 0
}
{
    "original_paper_id": 386802,
    "sentence": "Clinical antigen-specific studies however, have so far failed to
    ↪  show the desired efficacy despite their good safety profile.",
    "ref_paper_id": 18952695,
    "ref_paper_title": "Immunotherapy for the Prevention and Treatment of Type 1
    ↪  Diabetes",
    "ref_paper_authors": "M. Rewers, P. Gottlieb",
    "ref_paper_text": "\nI\n\nn the past 15 years, multiple clinical trials have
    ↪  attempted to find prevention for type 1 diabetes. The accompanying article by
    ↪  Bresson and von Herrath (1) reviews basic mechanisms underlying
    ↪  immunoprevention and immunotherapy of type 1 diabetes as well as selected
    ↪  human trials in the context of data from animal models. The second part of
    ↪  this minisymposium provides an overview of the recent or ongoing human trials.
    ↪  Immunotherapy for prevention of type 1 diabetes or to ameliorate the course",
    "label": 1
}
```


5 Model

Our goal is to decide, for a given sentence from a paper and a candidate reference (represented by a short "reference block" containing its title, authors, and a snippet of text), whether that reference should be cited in that sentence. We treat this as a binary classification problem over sentence–reference pairs:

$$(\text{sentence}, \text{ref\_block}) \mapsto y \in \{0, 1\},$$

where $y = 1$ indicates that the reference should be cited in that sentence.

On a high level, we fine-tune a SciBERT [6] cross-encoder on a set of labeled sentence–reference pairs, perform threshold calibration and a small hyperparameter grid search, and then reuse the trained model for paragraph-level citation inference on full papers.

5.1 methods

Input representation. Each training instance consists of:

- **sentence**: the in-paper sentence from the original article.
- **ref_block**: a concatenation of the candidate reference's title, author list, and the first 500-2000 characters of the reference paper
- **label** $\in \{0, 1\}$: 1 if the sentence actually cites this reference, 0 otherwise.

We format each pair as a standard Transformer sequence pair:

`[CLS]` sentence `[SEP]` ref_block `[SEP]`.

We use the SciBERT tokenizer with

- `max_length = 512`,
- `truncation = "only_second"` (if we need to truncate the input to match the 512 tokens, we only truncate from the ref block and not the original sentence),
- padding to `max_length` during training for efficient batching.

Class imbalance and loss. The dataset is moderately imbalanced: roughly 32% of sentence-reference pairs are positive (CITATION) and 68% are negative (NOT_CITATION). To counter this, we compute class weights on the training split,

$$w_c = \frac{N}{2N_c}, \quad c \in \{0, 1\},$$

where N is the total number of training examples and N_c is the number of examples of class c. These weights are passed to a weighted cross-entropy loss:

$$\mathcal{L} = -\sum_i w_{y_i} \log p_\theta(y_i \mid x_i),$$

so that mistakes on positive (citation) examples are penalized more heavily.

Training procedure. We split the structured dataset into train/validation/test by original paper ID so that no paper appears in more than one split. This prevents information leakage between splits (e.g., seeing different sentences from the same paper in both train and test).

Our initial baseline configuration uses:

- learning rate 2×10^{-5},
- batch size 8 (train and eval),
- 6 epochs,

The trainer keeps the checkpoint with the lowest validation loss, which helps avoid overfitting to the last epoch.

5.2 Threhsold tuning and Grid search

Sentence-level threshold calibration. Because we optimize log-loss on an imbalanced dataset, the default decision rule $p(y = 1) \geq 0.5$ gave low recall and therefore poor F1, even though the probability scores themselves were sensible. We therefore swept the decision threshold on the validation set and selected the value that maximized F1 for the CITATION class. This moved the best operating point from $t = 0.5$ to $t \approx 0.3$, increasing test F1 from about 0.33 to 0.55 while keeping accuracy roughly the same. All reported SciBERT results in this report use this tuned threshold.

Hyperparameter grid search. After establishing a good threshold, we ran a small grid search over two key hyperparameters:

- learning rate $\in \{1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}\}$,
- number of epochs $\in \{3, 4\}$.

For each configuration, we:

1. Initialize a fresh SciBERT classification model.
2. Train using the weighted loss
3. Evaluate on the validation set using the fixed tuned threshold $t^\star$.
4. Record validation accuracy, precision, recall, and F1.

The best configuration by validation F1 at the tuned threshold was:

- learning rate 1×10^{-5},
- 4 epochs.

This setting slightly improves validation F1 over the baseline configuration and reduces overfitting. We retrain our final model with this setting and use it for the rest of this project.

5.3 Metrics

We evaluate the sentence-level classifier using standard binary classification metrics, always focusing on the CITATION class as the positive label.

Table 1 summarizes the key sentence-level metrics for the SciBERT model under the naive and tuned thresholds.

Threshold	Split	Accuracy	Precision	Recall	F1
0.50	Validation	0.52	0.31	0.42	0.36
0.50	Test	0.50	0.29	0.39	0.33
0.30	Validation	0.62	0.48	0.56	0.52
0.30	Test	0.64	0.46	0.63	0.53

Table 1: Sentence-level SciBERT performance at different decision thresholds.

5.4 Inference

Once the SciBERT cross-encoder is trained, we use it as a scoring engine for paragraph-level citation inference on full papers. The inference pipeline is implemented as a standalone script (`cite.py`) that takes as input:

- a JSON file describing a single paper (paragraphs, references, and optional gold paragraph indices for each reference), and
- a directory `saved_model/` containing the fine-tuned SciBERT model, tokenizer, and an `inference_config.json` with the tuned threshold and other metadata.

Paper representation. For a given paper we assume:

- a list of paragraphs $(P_0, P_1, \ldots, P_{N-1})$,
- a list of references, each with a title, authors, and text (used to reconstruct `ref_block`),
- optionally, for evaluation, a list `referenced_in_paragraphs` giving the gold (truth) paragraph indices where each reference is actually cited.

Sentence splitting and scoring. For each paragraph P_j, we split the text into sentences using a simple regex-based splitter. For each reference r we build the same `ref_block` format used during training. We then:

1. For every sentence s in paragraph P_j, run the SciBERT model on the pair $(s, \text{ref\_block}_r)$ to obtain a citation probability
$$p(s, r) = P(\text{CITATION} \mid s, \text{ref\_block}_r).$$

2. Aggregate sentence-level probabilities into a single paragraph-level score for (P_j, r) using the *mean*:
$$\text{score}(P_j, r) = \frac{1}{m} \sum_{i=1}^{m} p(s_i, r),$$
where m is the number of sentences in P_j. We also experimented with $\max_i p(s_i, r)$, but the mean produced the most stable rankings across paragraphs with different lengths.

5.5 inference results

We evaluate the paragraph-level inference pipeline on an annotated scientific paper that was not part of the training, validation, or test splits. For each reference with at least one annotated citation paragraph, we compute:

- **Top-1 accuracy**: the fraction of references for which the single highest-scoring paragraph matches at least one gold citation paragraph.
- **Top-2 accuracy**: the fraction of references for which at least one of the top-2 predicted paragraphs matches a gold citation paragraph.

On this held-out paper we obtain:

- Top-1 paragraph accuracy: $8/13 \approx 0.615$,
- Top-2 paragraph accuracy: $10/13 \approx 0.765$.

In other words, for roughly two-thirds of the references, the true citation paragraph appears somewhere in the model's top-2 ranked paragraphs.

These results are consistent with the sentence-level metrics: the model's probabilities are well-calibrated enough to rank plausible citation locations highly, but not perfect. Qualitative inspection of the per-reference reports shows that many of the "false positives" are paragraphs that discuss the same topic or experiment as the gold paragraph and would not be unreasonable places to cite the reference, which partly explains why precision at the sentence level is not higher despite the model's useful ranking behavior.

6 Evaluation

6.1 Testing

Data splits and protocol. All quantitative evaluation is performed under a strict train/validation/test split at the *paper* level. Each original paper ID appears in exactly one of these splits, so the model never sees other sentences from the same paper in training and evaluation. The validation split is used for:

- tuning the sentence-level threshold $t^\star$,
- comparing hyperparameter configurations (learning rate and number of epochs),

The test split is held out until final model selection and is evaluated only once for the reported metrics.

Baselines and sanity checks. We use a TF-IDF + logistic regression model as a strong lexical baseline on the same sentence-reference pairs. Its performance (accuracy $\approx$ 0.50-0.52, F1 $\approx$ 0.33-0.36) is substantially worse than the tuned SciBERT cross-encoder, which reaches F1 $\approx$ 0.53 on the test set. This confirms that the transformer is capturing useful contextual and cross-sentence information beyond surface lexical overlap.

Robustness and qualitative testing. Beyond aggregate metrics, we perform manual inspection of model outputs:

- On the sentence-level test set, we examine high-confidence false positives and false negatives. Many false positives are semantically very close to the referenced paper and often occur in neighboring sentences or paragraphs to the gold (truth) citation; many false negatives occur in very short or generic sentences where even humans might disagree about whether a citation is strictly required.
- On the held-out full paper used for inference experiments, we inspect the per-reference reports generated by `cite.py`. For each reference, we compare the scores of the gold paragraphs to those of the predicted top-1 and top-2 paragraphs. In most cases where the model misses the exact gold paragraph, it still assigns relatively high probability to it, indicating reasonable ranking behavior even when top-1 is incorrect.

Overall, the testing strategy combines:

- quantitative evaluation on a clean held-out test split for sentence-level performance,
- paragraph-level top-k evaluation on a separate annotated paper,
- and targeted qualitative analysis of the mistaken predicctions.

This gives us a reasonably complete picture of how the system behaves, where it succeeds, and where it fails.

7 Limitations and Problems

While we were working on our project, we encountered a variety of limitations that required workarounds, patience, or simply accepting defeat. The major limitations we faced were:

7.1 S2ORC Dataset Size

The S2ORC dataset is extremely large, at over 500 GB in size. As outlined in the Dataset Retrieval section, downloading the dataset itself was a significant portion of our time spent on the project. Additionally, there was no cloud storage solution available for free to store 500 GB of data, so we had to store it locally on CJ's personal computer. This severely limited our ability to collaborate effectively, as only one of us could access the full S2ORC dataset. We were, however, able to work around this by splitting up the work in a different way.

The size was not just a factor in storage and download time, but also in processing time. Searching through the dataset, even after building our local mapping, was still quite slow, despite our best efforts to parallelize the work. Not only did this slow down our dataset building process, requiring about 2 weeks to build the full dataset, but it also restricted CJ's ability to utilize his personal computer for other tasks while the dataset building was running. This resulted in trading off between data aquisition and the usage of his PC, which is used as a remote machine for all of his projects. On average, it took about 5-7 minutes to identify a usable entry from the CiteWorth dataset and extract all relevant information from the S2ORC dataset, Thus resulting in over 100 hours of continuous processing time to build the full dataset.

7.2 Parsing Issues with S2ORC

The S2ORC dataset is not perfectly structured, and there are many inconsistencies in the way the data is parsed and stored. For example, some papers have their main text stored within a field called "body" that contains a variety of inconsistent nested subfields. Other papers have their main text stored in a field called "content", which is structured differently. Additionally, while many papers contain annotations for the locations of in-text citations, bibliography entries, and other useful information, many papers are missing this information or have limited annotations, so this method was not a reliable way of extracting the necessary data.

In addition, while all papers that have paragraph entries in the CiteWorth dataset have corresponding entries in the S2ORC dataset, not all referenced papers in the CiteWorth dataset have full text available in the S2ORC dataset, thus was not in the S2ORC dataset at all. This made it difficult to build a complete dataset,

as we had to skip many entries that still required a portion of the processing time to determine that they were unusable. This may, however, have a silver lining, as it forced our dataset to contain a larger bredth of citation examples, rather than many examples from fewer papers. This may help our model generalize better to unseen papers during inference, as it was trained on a wider variety of examples.

7.3 Low Precision

A major modeling limitation is that our final SciBERT classifier operates at a relatively low precision. At the tuned sentence-level threshold ($t \approx 0.3$), we achieve good recall on the test set (around 0.63) and a reasonable F1 score (around 0.53), but precision is only about 0.44. In practice, this means that a non-trivial fraction of suggested citations are false positives, which would still require manual filtering by an author.

There are several reasons for this behavior. First, we deliberately chose a low threshold to favor recall over precision: missing citations is worse than suggesting extra ones, but this inherently produces more false positives. Second, many "false positives" are semantically related references that would not be unreasonable to cite, so from the model's perspective they are hard negatives. Overall, our current system is tuned for high recall and decent F1, at the cost of precision. A more production-oriented system would likely adopt a stricter threshold to reduce false positives while preserving as much recall as possible.

7.4 Corruption...

During the building of the test dataset, CJ's computer crashed. Upon restart we received the error shown in figure 1. This error seems to indicate that the filesystem was corrupted. This drive is now approximately 8 years old, so it may be that it has reached the end of its lifespan. Unfortunately, this meant that all the data on the drive was lost, including the entirety of the S2ORC dataset, the built index, the partially built test dataset, and all of the additional logic that was written to build the test dataset (not to mention over a TB of personal data). Despite numerous attempts at repairing the drive, including restarts, opening in other OSs, and running drive repair utilities, we were unable to regain access to the data on the drive. Fortunately, all code prior to building the test set was in GitHub, so not all was lost.

We unfortunately did not have time or storage capacity to re-download the S2ORC dataset and re-build the test set, so we worked around this by hand parsing a single paper that was not contained in any of our current datasets (it is actually a paper that is closely related to CJ's thesis research). While this is not ideal and is very time consuming, it allows us to at least demonstrate the inference capabilities of our model.

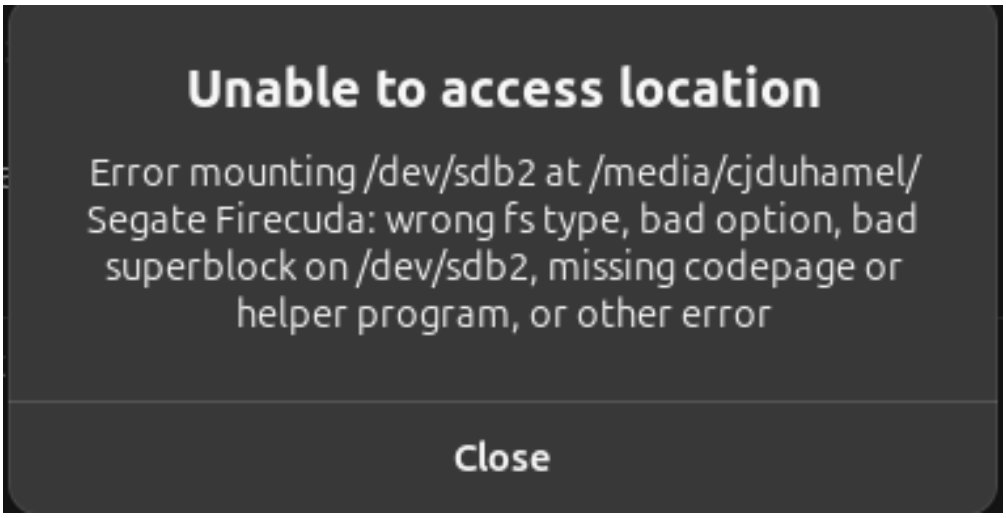

Figure 1: Error message after CJ's computer crashed during test set building.

8 Conclusion and Future Work

In conclusion, our Auto Citation tool demonstrates the potential effectiveness of Transformer architectures in creating citation tools for academia. While we faced many challenges and constraints during the project, we were able to build a functioning prototype that can place citations in a document based on a provided bibliography and reference list.

Future work on this project coudl involve improving the descision mehchansim for placing citations, such as considering more contextual information from the document and references, or incorporating user feedback to refine the citation placements. Additionally, as for most machine learning tasks, gathering a curating more high quality data would likely improve model performance. Finally, integrating the tool into an easy to use user interface, such as a VSCode extension or web app, would make it more accessible, testable, and usable.

References

[1] D. Wright and I. Augenstein. "CiteWorth: Cite-Worthiness Detection for Improved Scientific Document Understanding." arXiv: 2105.10912 [cs], Accessed: Nov. 13, 2025. [Online]. Available: http://arxiv.org/abs/2105.10912, pre-published.

[2] K. Lo, L. L. Wang, M. Neumann, R. Kinney, and D. Weld, "S2ORC: The Semantic Scholar Open Research Corpus," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 4969–4983. DOI: 10.18653/v1/2020.acl-main.447. Accessed: Nov. 13, 2025. [Online]. Available: https://aclanthology.org/2020.acl-main.447/.

[3] A. D. Wade, "The Semantic Scholar Academic Graph (S2AG)," in *Companion Proceedings of the Web Conference 2022*, ser. WWW '22, New York, NY, USA: Association for Computing Machinery, Aug. 16, 2022, p. 739, ISBN: 978-1-4503-9130-6. DOI: 10.1145/3487553.3527147. Accessed: Dec. 10, 2025. [Online]. Available: https://dl.acm.org/doi/10.1145/3487553.3527147.

[4] "Hugging Face – The AI community building the future.," Accessed: Dec. 11, 2025. [Online]. Available: https://huggingface.co/.

[5] W. Mckinney, "Pandas: A Foundational Python Library for Data Analysis and Statistics," *Python High Performance Science Computer*, Jan. 1, 2011.

[6] I. Beltagy, K. Lo, and A. Cohan, "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)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3615–3620. DOI: 10.18653/v1/D19-1371. Accessed: Dec. 11, 2025. [Online]. Available: https://aclanthology.org/D19-1371/.