Preliminary project report

1. Introduction

1.1. Selected template: CM3060 Natural Language Programming, 12.1 Project Idea 1: Identifying research methodologies that are used in research in the computing disciplines.

The template describes 3 levels of success criteria in increasing difficulty:

1.1.1. Correctly identifying the computing discipline of a paper.

This is the easiest level, as it only requires a rough classification, which can most probably be done using only the abstract and title.

1.1.2. Correctly identifying the field within the discipline.

A slightly more difficult challenge, requiring a more in-depth understanding of the paper itself.

1.1.3. Correctly identifying the research methodology of the paper.

The hardest level is clearly distinct from the previous two. This one requires understanding the actual methodology used in the papers to classify them only by this aspect.

1.2. Description of the problem domain and possible approaches

As we will see in the literature review, this exercise is not a perfectly solved problem. Academic research is ongoing regarding the best approaches.

There are multiple difficulties we face:

- Supervised classification methods require a large enough, properly tagged dataset. It is possible to find such a large tagged dataset (or at least create it by combining multiple datasets), but only for disciplines and fields and not for problem 1.1.3. It would be possible to create the tags for this exercise manually. However, it would not only require a lot of effort to diligently read through hundreds, if not thousands, of research papers, covering all possible classes multiple times, which would already be outside the scope of this project. But also, the author does not possess the required domain knowledge either, making this approach unfeasible.
- Fully unsupervised methods also might fall a bit too short for our purpose (more on that later), but even for trying them, we need to solve the length problem. Without further experimentation, we have no way of knowing if using only the title and abstract of the papers will be sufficient enough for proper classification. There is a

reasonable chance that - especially for the research methodology classification - these blocks do not require enough information for the discipline/field classification algorithm to be effective. This means we might need to train our models on the entire content of the papers. However, that is easier said than done. One often-used and highly effective method for text classification is the use of bidirectional transformer models, such as BERT. Unfortunately, BERT has a limitation of using 512 tokens, which might not be enough to represent longer documents properly. Even if theoretically it is possible to increase the token limit (as some of the derived models like DeBERTa and similar models do), the memory and compute requirements grow quadratically, making this direction unfeasible for us. Instead, we might opt either for models tailored to be used with longer documents without the quadratic compute increase (Longformer, BigBird, LED, Transformer-XL, XLNet, etc), or we might manually build simpler layers on top of each other (hierarchical approaches, like RoBERT [not to be confused with RoBERTa] and ToBERT).

- However, even if we can automatically cluster the research papers, chances are, they will be clustered by their primary content, which will resemble disciples and fields. This could be useful for problems 1.1.1. and 1.1.2., but not so much for 1.1.3., where the research methodologies might stay hidden. (We have to mention that there is a chance in case of a multidimensional/multilabel classification that either some of the dimensions automatically capture the research methodologies or some of the labels will refer to these. But since it is not guaranteed, we will work with the assumption that it might not happen.) To ensure we can also capture the research methodologies, we might need a bit more hands-on approach.
- One possible angle is identifying blocks of text in each paper that describe methodologies and use only those blocks for this particular clustering. This, however, requires identifying the topic of each block. Here, we can use keyword-based search or unsupervised clustering to find the blocks describing the methodologies, or we can use automated topic-extraction like BERTopic. Then, we can extract just the relevant paragraphs and classify or cluster them, which hopefully will give us the research methodology of each paper.

1.3. Motivation

My personal motivation has changed a lot already since starting this project. Originally, I wanted to focus on the more traditional ML methods to highlight their usefulness compared to LLMs. This partially remained the same, as I still want to look for alternatives to prompt-based LLMs. However, based on the reviewed literature, non-LLM Transformer models have piqued my interest; thus, from now on, I plan to focus on these. This is a change in the sense that I will not explore the more traditional ML models (statistical approaches, non-transformer neural networks, etc).

Another revelation during the initial exploration was regarding the structure of the work needed to create a classification system like this project. It's true that after having a refined dataset and ending up with a model, it still requires proper hyperparameterization, and the model still has to be trained. However, my current understanding is that experimenting with the various models and their available options requires a robust infrastructure. Without it,

every experiment would be prohibitively costly (in time and energy), and thus would prevent exploring the available options. Similarly, I realized that the quality of the results of the models will depend significantly on the quality and quantity of the available data.

Thus, my approach (as described in more detail in section 3) is not to focus solely on a single randomly picked model, and instead:

- Allocate a significant amount of resources to acquire and transform the proper datasets needed for the project.
- Create a professional infrastructure that is capable of handling 100s of gigabytes of datasets, and makes it possible to efficiently train, infer, and evaluate multiple models.
- Run multiple experiments on supervised classification for problems 1.1.1 and 1.1.2.
- Run multiple experiments on methodology-based clustering and classification.

2. Literature review

As mentioned in the introduction, both long document classification and topic modeling are not completely solved problems, there is active research in both areas. However, before diving into the details, we have to start with the foundational papers for Transformers.

2.1. Transformers

"Attention Is All You Need" (Vaswani et al., 2017, collaboration of Google Brain, Google Research, and the University of Toronto) introduced the Transformer architecture. This paper is often cited as the foundation for the emergence of LLMs, but it deserves attention in its own right. The paper questioned the status quo of the time, which relied heavily on recurrent and convolutional layers. Instead, it builds on "self-attention", a mechanism that directly weighs the significance of each word based on all other words. This enables capturing global influences within a text, encoding each word not just by its dictionary meaning, but its meaning within the broader context of the document.

Two years later, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" (Devlin et al., 2019, Google AI Language) built directly on this self-attention-based Transformer logic. This paper marks the diverging point between unidirectional systems like OpenAI's Generative Pre-trained Transformer and Bidirectional Transformers. The authors argue that in unidirectional models (left-to-right, right-to-left, or their shallow concatenation), every token can only attend to the previously available (generated or inputted) tokens, limiting it's awareness of the full context. As a reply, they introduce two novel training logic. Masked Language Modeling (MLM) randomly masks words in sentences and predicts them using the bidirectional context. Next Sentence Prediction (NSP) takes every sentence as A, and 50% of the time, it takes the next sentence as B, while 50% of the time, it takes a random other sentence from the text as B to train. These advances enabled BERT to reach far better scores in the GLUE benchmarks than the competitors of the time (80-82 points on average vs 71-75 points for SOTA, Elmo and GPT).

While we are going to come back to works derived from BERT, there is another angle being explored in "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context" by Dai et al. (2019, Carnegie Mellon University, Google Brain). Publishing a few months before BERT was published, the authors recommended a solution to solve the traditional Transformer models main limitation: fixed-length context. They introduce recurrence into the self-attention network by reusing the hidden states from previous segments as a starting point instead of computing them from scratch. This, in effect, creates a continuous long-term memory, which maintains consistency even through very long sets of tokens (up to thousands).

The same authors also published "XLNet: Generalized Autoregressive Pretraining for Language Understanding" (Yang et al., 2019, Carnegie Mellon University, Google Brain) a month after BERT was published. They argued that BERT suffers not only from its fixed context window (which, in their view, was already superseeded by the previous Transformer-XL model), but also by masking 10-15% of the words, it actually creates discrepancies. They propose a generalized autoregressive pretraining method leveraging permutation language modeling. They argue that XLNet uses permutations effectively to factorize input sequences in all possible orders, capturing bidirectional context without relying on masks. At the time, XLNet overperformed all other existing models, including BERT.

Just a month later, the "RoBERTa: A Robustly Optimized BERT Pretraining Approach" (Liu et al., 2019, Paul G. Allen School of Computer Science & Engineering, Facebook AI) paper takes the original BERT implementation and optimizes it. They remove the Next Sentence Prediction and carefully tweak multiple training parameters. This results in RoBERTa outperforming both BERT and XLNet by a significant margin on the GLUE benchmark.

Two more months after this, in "SciBERT: A Pretrained Language Model for Scientific Text" (Beltagy et al, 2019, Allen Institute for Artificial Intelligence), the authors take a different approach. They pretrain BERT on a semi-random sample of corpus from Semantic Scholar, including papers in the computer science and biomedical domains. The results are mixed: in some benchmark tasks, SciBERT achieves the best results, in other tasks, it closely approaches the then-state-of-the-art results.

Another two months pass, and in "Hierarchical Transformers for Long Document Classification" (Pappagari et al., 2019, Johns Hopkins University), two new techniques are introduced. Recurrence over BERT (RoBERT, not to be confused with the previously reviewed RoBERTa) and Transformer over BERT (ToBERT). Both of these techniques aim to overcome the 512-token limit of the traditional Transformer models and, more importantly, the quadratic increase in computing resource for longer token lengths. The goal is to be able to process longer documents without having to truncate them. Recurrence over BERT splits the document into multiple overlapping segments and feeds each of those segments into BERT. RoBERT then takes these outputs and feeds them into a small LSTM layer to create the final document embedding, which feeds into 2 fully connected ReLU layer, and a final softmax. ToBERT works similarly, but here, the output of the BERT models working on the overlapping slices feeds into a small self-attention Transformer model. Performance-wise, RoBERT achieves O(nk) computational complexity, while ToBERT has O(n2/k2) which is theoretically inferior, but still overperforming the original O(n2) BERT model. The techniques

overperformed the traditional BERT model, especially the more long documents the dataset contained.

In April 2020, in "Longformer: The Long-Document Transformer" (Beltagy et al, 2020, Allen Institute for Artificial Intelligence), the authors explore another approach to overcome the quadratically increasing computational requirements of Transformer models for long documents while trying to keep the coherent context for the entire document. Since the traditional self-attention model itself requires the quadratic increase due to every token being related to every other token in the document, they recommend a local windowed attention with a task-motivated global attention. The resulting Longformer (aka LED) model (the vectorized and cuda versions) scale linearly by the document length both for required training time and for memory requirements, while overperforming traditional BERT/RoBERTa models in various benchmarks.

3 month laters, in "Big Bird: Transformers for Longer Sequences" (Zaheer et al, 2020, Google Research), another paper introduces a somewhat similar solution for the same problem. The authors use a novel sparse attention mechanism combining local, global, and random attention patterns. This dramatically reduces the Transformer complexity while maintains strong modeling capacity for long documents. It enables Big Bird to achieve linear complexity in sequence length, and handle up to tens of thousands of tokens efficiently.

Around the same time, the authors of Transformer-XL publish "Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing" (Dai et al., 2020, Carnegie Mellon University, Google Brain). In this, they introduce a "funneling" logic that gradually compresses sequence length for long documents by eliminating redundancy. This allows to model to achieve state-of-the-art effectiveness with reduced computational cost.

Jumping ahead of time, "An Exploration of Hierarchical Attention Transformers for Efficient Long Document Classification" (Chalkidis et al., 2022, University of Copenhagen, CSIRO Data61, Athens University of Economics and Business, Pioneer Centre for AI) investigates the RoBERT/ToBERT-style hierarchical transformers. After testing four different layouts (Ad-Hoc, Interleaved , Early-Contextualization, Late-Contextualization), their findings confirm that properly pretrained hierarchical models can achieve similar results as the LED/BigBird sparse attention models.

In "Length-Aware Multi-Kernel Transformer for Long Document Classification" (Han et al., 2024, University of Memphis, New York University), the authors state that both hierarchical and sparse attention models tend to expect fixed-sized chunks and overfit to specific-sized inputs, performing worse on other sizes. They introduce Length-Aware Multi-Kernel Transformer (LAMKIT). The model uses multiple neural encoders with various kernel sizes to prevent context fragmentation, and it uses a length-aware vectorization (LaV) to promote robustness. The model outperforms all other models in most cases, or lands on second place behind either BigBird, or the health-focused H-BERT.

2.2. Other relevant classification research

These papers are either not using Transformers, or their focus is not on introducing new Transformer models, instead, they are "just" using them for classification purposes. They are

not always relevant to the current project due to our focus on using Transformers, but they might provide useful insights into the challenges of classifying scientific papers.

In "CSO Classifier 3.0: a scalable unsupervised method for classifying documents in terms of research topics" (Salatino et al., 2022), the authors present an unsupervised model for research paper classification, using the predefined taxonomy of Computer Science Ontology (CSO).

In "Unsupervised Multi-Label Document Classification for Large Taxonomies Using Word Embeddings" (Melsbach et al., 2019), the authors use an embedding-based approach to match long documents to a predefined set of hierarchical taxonomy. While this is not a Transformer-based approach, it could serve as a baseline for more advanced models.

"Application of BERT Model for Unsupervised Text Classification using Hierarchical Clustering for Automatic Classification of Thesis Manuscript" is, as its title suggests, a BERT-based approach. The authors embed thesis documents using a pre-trained BERT model, then applied hierarchical clustering to group similar documents without prior labels. This approach discovers categories from the data, although, since the nature of unsupervised clusterings, the resulting clusters need to be manually observed and assigned human-understandable labels.

In the very recent "Automated research methodology classification using machine learning" (Kosztyán et el., 2025, University of Pannonia, Institute of Advanced Studies Kőszeg), the authors explore various classical supervised binary classification ML models. This paper is interesting for this project due to its exact focus on categorizing research papers based on their research methodologies, which is rare, compared to topic-based classification. Unfortunately for us, they only classify whether the papers are using "quantitative" or "qualitative" methods. But what's interesting for us is that they compare their model's accuracy using abstracts only to using the full texts, and they report dramatic improvements when using full text.

2.3. Topic-modeling

One of our challenges will be extracting topic information from paragraphs to be able to identify, then classify paragraphs describing research methodologies.

In "Latent Dirichlet Allocation" (Blei et al., 2003, University of California, Stanford University), the authors present LDA, a three-level hierarchical Bayesian model: "The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words."

Further papers requiring more attention:

- Top2Vec: Distributed Representations of Topics (Angelov et al., 2020)
- Anchored Correlation Explanation: Topic Modeling with Minimal Domain Knowledge (Gallagher et al., 2017)
- TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency (Dieng et al., 2017)

- Self-Supervised Neural Topic Modeling (SNTM) (Bahrainian et al., 2021)
- Effective Neural Topic Modeling with Embedding Clustering Regularization (ECRTM) (Wu et al., 2023)
- Self-Supervised Learning for Neural Topic Models with Variance–Invariance–Covariance Regularization (VICNTM) (Xu et al., 2025)

3. System design

As described in Section 1., my approach is to build a reliable and efficient environment first that enables quick and reproducible experimentation.

While, of course, I plan on implementing the actual models, I assume that the primary benefit of my work will be presenting a reproducible environment ideal for similar complex classification workflows.

3.1., Planned infrastructure

3.1.1. AWS as cloud infrastructure

The system will be implemented on Amazon Web Services (AWS). There were multiple considerations behind choosing the cloud as the implementation option:

- The used datasets will be in the hundreds of gigabytes. Downloading, storing, and processing such an amount of data locally is not really feasible.
- Due to a failed CPU fan in the primary laptop and the lack of replacement parts in their country, the author currently only has access to a Windows tablet as a computer, which is woefully inadequate for any kind of Machine Learning work.
- Cloud systems, on the other hand, offer a great deal of flexibility in the type of virtualized hardware they provide, which allows for the use of heavyweight infrastructure for demanding tasks while utilizing low-cost solutions for lightweight processes.
- During the time of this project, the author will travel intercontinentally multiple times, during which time access to the same computer is not guaranteed. By using cloud services, even if a catastrophic failure happens to the local infrastructure (for example, even the current fallback tablet dies or gets stolen), the project will be accessible from any other replacement devices.

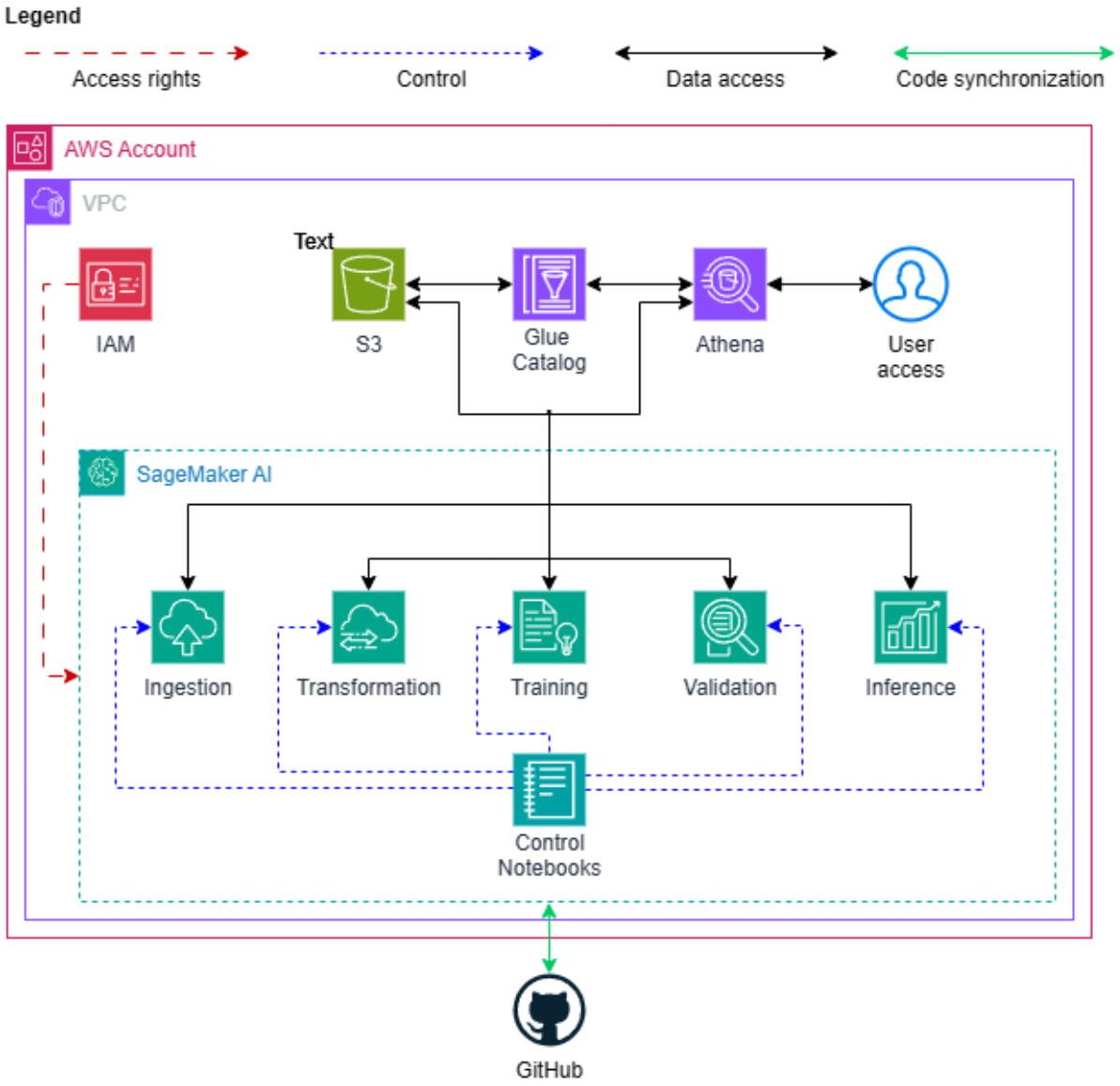
When it comes to selecting a cloud provider, there are multiple options. AWS, Google Cloud Platform, and Microsoft Azure are some famous examples with wide service offerings, but there are a plethora of smaller providers as well.

AWS was picked as the implementation platform for the following reasons:

- The author has some level of previous familiarity with the system.

- It is guaranteed that AWS offers all the services required for the project.
- Some of the core datasets planned to be used are already being shared on AWS S3 buckets, and copying them into another S3 bucket is one of the easiest ways of accessing them.

3.1.2. System architecture



As pictured above, the system architecture is as follows:

- There is a single AWS account with a single VPC containing all elements.
- Access rights among services are managed in IAM.
- API keys and other sensitive information is stored in Secrets Manager (not pictured).
- S3 is being used for storing all data.

- The relevant S3 locations are crawled by Glue Catalog to make data available for Athena.
- Athena enables accessing raw json, csv, etc files as if they were SQL tables. This allows for easy manual data exploration and SQL-based data transformations.
- All other workflows are happening within SageMaker AI, using the classic (non-studio) interface.
- A Notebook Instance with Git integration enables us to run Jupyter Lab, on which we can execute multiple Jupyter Notebooks. These notebooks are being used for both exploration and setting up and controlling specific longer-running functions.
- The first of these functions is the ingestion of the required datasets. All the datasets are saved into an S3 bucket. Some of the functionality can be handled (especially during initial exploration) in a notebook or as manual command-line tools, but in an ideal implementation, the control notebooks only orchastrate, while the real process is happening in SageMaker Processing Jobs.
- The second set of functions is everything related to data transformation. Although some of this can happen directly in Athena by creating views and even new tables (which are subsequently stored on S3 as well), we keep this option open for more advanced techniques. Similarly to the previous step, this could take in SageMaker Processing Jobs.
- The third big step is training. Here, SageMaker provides purpose-tailored services called Training Jobs and Hyperparameter Tuning Jobs. Both of these can take data from S3, and can be orchestrates from the control Notebooks.
- The fourth step is validation. This is separated in the plan from the Training step to highlight its importance, but in reality, it will be part of the same Training and Hyperparameter Tuning Jobs.
- The fifth and final step is providing an inference service. Here, AWS offers multiple options, from running real-time API endpoints to one-time batch inference. During the course of this project, we are most probably going to use batch inference due to its lower cost.
- All the SageMaker scripts can access both the files on S3 directly, or process them as SQL tables through Athena. Having access to Athena might be useful for them, as any data modeling happening through Athena avoids having to deal with data transfer and memory constraints, which could create a significant challenge for pandas-like in-memory scripts.

3.2. Datasets

The project will utilize multiple datasets acquired continuously when the need arises. However, even in the planning phase, it is clear that we are going to rely heavily on the following two sources:

- **Semantic Scholar (S2):** S2 is one of the few sources that provide full-text data of papers in their "s2orc" dataset. This dataset is also annotated, with multiple different annotations marking the character boundaries of paragraphs, authors' names, etc. The "papers" dataset enriches this with multiple additional fields, out of which title and abstract will be the most important for us. Both datasets contain a set of external ids, making it possible to connect them to other datasets. Joining is going to be essential for us, since the S2 datasets only contain a 2-level hierarchical categorization. One of these ("cs" for Computer Science) we need to use to limit our scope to the computing sciences, but that leaves us with a single level categorization, which would be inadequate for the 2-level topic categorization required by the project. S2 makes other datasets avaialable as well (authors, citations, etc), which could be useful for similar clustering attempts, but in this project, we are going to focus on pure text-based clustering, and thus, not going to use these.
- **OpenAlex:** The spiritual successor of the discontinued Microsoft Academic Graph contains multiple topic classification systems. Most important for us is the 4-layer hierarchical classification. The first two layers are going to be used by our filters (Physical Sciences and Computer Science), but the remaining two layers can be used for our 2-layer topic-based classification models. There are multiple other taxonomies included in the dataset (like "concepts", "mesh", "keywords", not primary topics, etc.), so further exploration is required to determine the best use case for each model.

Thankfully, both of these datasets contain a shared "mag" (Microsoft Graph API) external ID, which makes it possible to join them.

It must be noted, that the supplementary taxonomies theoretically would allow us to first filter for every publication that has any references to any Computer Science topics, even if its primary topic classification is within another domain (like Biomedical, Social Sciences, etc). This would allow us to capture papers belonging to other domains as well if they utilize computer science tooling (like ML) as they often do. In this case, even the first- and second-level topic categorization would too become variable, thus allowing us to classify accordingly. This approach requires further investigation, but it will most likely be rejected. There is a high likelihood that this would create an unbalanced dataset, with the overwhelming majority of the filtered papers still belonging to Computer Science and only a few papers belonging to every other category. Since that would seriously bias any models towards picking Computer Science, we would need to balance the dataset first, but that would require eliminating most of the most valuable Computer Science slice, dramatically decreasing the training input.

3.3. Disciple and field classification

As mentioned previously, the primary focus of this project is to enable quick and efficient exploration and experimentation of multiple models. Thus, the exact details of the final set of models are not yet available at the time of writing this preliminary report.

However, we can already collect some directions to explore.

- Given that we have an existing classification for some of the texts, we are going to experiment with supervised methods.
- One aspect of the experiments will be comparing the accuracy of classification based on title only, title+abstract, and full text.
- As models, we are primarily going to use Transformer models. Given the arbitrary length of the full texts, our focus will be on those being able to handle longer token lengths: Longformer, BigBird, XLNet, RoBERT, ToBERT, and other hierarchical tranformers. BERT, CSO Classifier 3.0, and word embeddings based classification could serve as baselines for comparison.

3.4. Research methodology classification

Compared to the topic-classification, the methodology-classification will be a more complex process, with various methods that differ from each other on a structural level.

Option 1: Unsupervised clustering on the full texts, with manual observation of the resulting locations. Some dimensions might represent research methodologies, only requiring reduction and manual labeling.

1) Option 2, step 1: Identifying paragraphs describing research methodologies first.
    a) Option 2, step 1, method a: Unsupervised clustering on the paragraphs and manually identifying clusters referencing research methodologies.
    b) Option 2, step 1, method b: Unsupervised topic extraction from each paragraphs, with manually identifying topics that belong to research methodologies.
    c) Option 2, step 1, method c: Non-ML classification of paragraphs based on full-text search for a list of predefined keywords.
2) Option 2, step 2: Given the already identified text-chunks describing the research methologies, we again, have multiple options on how to actually classify the documents.
    a) Option 2, step 2, method a: Unsupervised clustering, with manual identification and labeling of the resulting clusters.
    b) Option 2, step 2, method b: Unsupervised topic extraction.
    c) Option 2, step 2, method c: Weakly guided approach of providing a predefined set of keywords as methodology labels together with the methodology chunks and clustering them together to label each paper based on their nearest label.

As is evident, this part of the project will require further research and planning before having any chance for a real implementation.

Feature prototype

Given the nature of the project, I decided against prototyping a single machine-learning model just for the sake of having a prototype for the following reasons:

- Just writing a Python script that would train and evaluate a given model (which we would expect from a prototype in this template) is not very useful without having the proper dataset and space for fine-tuning the model and comparing its results to baseline results.
- Creating a single-use environment for a single model, would require a lot of time: all the datasets need to be acquired, most probably downloaded locally, transformed temporarily, etc. Only to be used once for the prototype, and then be discarded so that a proper environment can be implemented for the final project.

So instead, I decided to follow the development plan that was outlined in the previous parts: implement the required cloud environment, acquire the core datasets, and transform the data to be ready for being used in models.

This is my progress so far:

- A new AWS account was created.
- IAM Roles and users were created to enable interacting with the services for both other services and for manually executed scripts.
- S3 Buckets were set up. One, called "bsc-final-sagemaker-data-bucket" is meant to contain all the raw datasets. Another one called "bsc-final-athena-data-bucket" is intended to be used by Athena to store both temporary query results and purposefully created materialized tables.
- An ingestion script was executed in a notebook format and ingested 328GB of data in 361 files from the S2 datasets API. This script still needs to be finalized and extracted to a separate ingestion script as described above for reproducible results. One of the challenges is that the files are only available on signed, temporary URLs after querying the datasets API to get these URLs. However, since the files are large, during download, they might expire, which requires getting another set of temporary URLs and continuing from where we left off. For the first time, we overcame this by manually running chunks of scripts to be able to verify the usefulness of the dataset as soon as possible, but this needs to be productionalized.
- For the "papers" dataset, all the downloaded .gz files were first uploaded to S3 for long-term storage, before a script downloaded all of them one-by-one to extract them and upload them to S3 again. This was based on not knowing that Athena can actually access compressed files without having to extract them first. This step requires further exploration to confirm weather all other use cases will be able to directly handle the compressed format, or would any of them require the raw formats. Either way, currently, the "papers" Athena table is based on the extracted files, whiles the "s2orc" full text Athena table is based on the compressed files. This needs to be standardized in one of the two ways.

- For the OpenAlex dataset, the entire exposed S3 bucket was synchronized into an own S3 bucket using a console command. This ingested 417GB of data in 2571 files, spanning across 12 different datasets. The ingestion worked well, but this process should also be transformed into a dedicated ingestion script for reproducibility. Within that restructuring, the restructuring of the S3 directory structure should be included, as currently it is inconsistent between the datasets coming from different source systems.
- Glue Catalog crawlers have been set up to parse the raw data coming from the S2 source and make them available as databases. Triggering the crawlers after ingestion currently happens manually. This works fine, but this also should be part of the respective ingestion scripts. Additional crawlers need to be set up to parse the relevant OpenAlex datasets.
- Exploratory data-transformation was conducted on the S2 "papers" and "s2orc" datasets to evaluate the feasibility of using them. The datasets were proven useful in general (especially the title and abstract from "papers", and the text and paragraph annotations from "s2orc"), but as mentioned previously, the shallow categorization was insufficient. This led to acquiring the OpenAlex dataset, which still waits to be explored.