OpenAlex: End-to-End Process for Concept Tagging

Abstract

Due to Microsoft shutting down their Microsoft Academic Graph (MAG) service, there is a gap in academic research tools that needed to be filled. OpenAlex is an open-source project that is hoping to fill that gap by providing an interface very similar to MAG so that research into academia can continue unimpeded. One of the great features of MAG that needed to be replicated was the concept tagger that tagged academic papers with the topics/concepts that are present in that paper. Therefore, a multi-class deep learning classifier was trained on the historical MAG data to be able to replicate the tagging process done through MAG. The model performs well with high precision and recall but more importantly, human quality testing shows the model may be better at tagging papers than the MAG in many instances. This model was then deployed in AWS to serve as a REST API and to be used on all new academic papers found in OpenAlex.

1 Introduction

The Microsoft Academic Graph is a heterogeneous graph of scientific publications and relationships between those publications. Microsoft has been collecting academic papers and tagging them with concepts/topics in order to facilitate research on trending topics in academia as described in (Wang et al., 2019) and (Wang et al., 2020). Unfortunately, Microsoft has discontinued MAG and this concept tagging service starting at the end of 2021. In order to fill the gap this would create in academia, a concept tagger needed to be built which would tag new papers with concepts.

MAG allowed a user to explore concept/topic relationships between papers. Concept tags can range anywhere from a basic topic such as “medicine” or “computer science” all the way to a more detailed topic such as “analysis of molecular variance”. Each concept can have parent concepts, children concepts, or both. The following picture shows an example of a concept that has both parents and children:

Figure 1: Example Concept from MAG

The goal of the OpenAlex Concept Tagger was to replicate the MAG concept tagging, not serve as an exact replacement of that process. Therefore, unlike other work which needed to build out the concepts/topics from scratch (see Liu et al., 2013), the OpenAlex concept tagger would be able to use the MAG historical data as a labeled dataset which would make the training of a concept tagger more straightforward and able to be accomplished using a deep learning classifier. This deep learning classifier would need to have high precision because if a paper is tagged with a concept, it is important that the tag is correct.

To understand what exactly needed to be built, an exploration of the data needed to be done. This would allow for a complete understanding of what features might be available when building the model. At the very start of building the concept tagging model, abstracts for each paper were not available to be used as a feature for training. Therefore, the model building process was split up into a "V1" phase and a "V2" phase. During the V1 phase, a model was built using any feature that was available outside of abstracts knowing that this would not be the best model. However, with the decommissioning of MAG at the end of 2021, there was a need to have some type of concept tagging in OpenAlex and it was correctly believed that the newly created concept tagger would be sufficient in the short term. The V2 phase began once the abstracts were available for each paper and this phase was seen as what would be the final version of the concept tagger.

There was an extensive amount of exploration into the historical MAG data before starting to build the model. Not only was there a high-level exploration to determine any features that could be used for the model, there was a more detailed exploration into the candidate features as well. The following features were considered candidates for both the V1 and V2 model because there was at least a small chance some signal could be created:

- Paper title
- Document type (publication method)
- Journal Title
- Author
- Affiliation
- Publication date
- Abstract

Publication date was removed from this list pretty early on because unlike other models, seasonality is not assumed to have any effect on the tagging. Authors were also removed as a candidate feature due to the fact that there are such a large number of authors. Affiliation was explored but unfortunately only 25% of the data has this field filled in so it most likely would not be a good feature for the model. Author and affiliation are also two features that required additional steps for disambiguation and so most likely would not make good features.

Document type might be a helpful feature because certain publication methods might be more prevalent for certain fields of study which would be helpful for the model to know. The following figure shows the breakdown of the different document types:

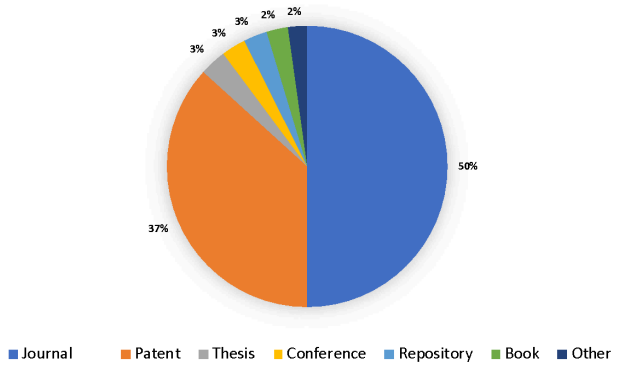


Figure 2: Document Types in the MAG Data

It can be seen that journals and patents make up a large portion of the historical data while all of the other document types make up between 2-3% each. Seeing as journals make up such a large percentage of the data, the name of the journal is kept on the list as a feature to use in the model.

Obviously, the paper title will be the most important feature to use for the V1 model when trying to tag concepts because that will give the best indication of what topics the paper might contain. The following figures show the languages of the paper titles:

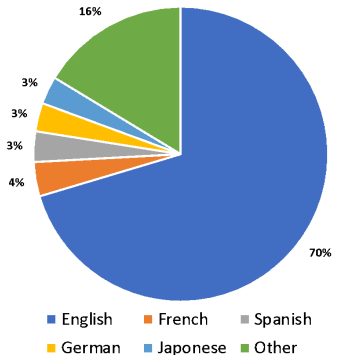


Figure 3: Language of Paper Titles in MAG Data

Papers in English dominate the overall distribution of papers in the MAG historical data and the percentage of new papers that are in English have been trending up in recent years. The following shows the overall statistics for the number of tokens found in the training set paper titles.

Average	12
Standard Deviation	6
Minimum	2
Maximum	125

Table 1: High Level Statistics for Number of Tokens in Paper Title

As a hypothesis to be tested, it was expected that the longer the title of the paper, the more information the model will have to guess the topics. While more information is usually beneficial, there is always the chance that the longest paper titles have too many words and confuse the model. This will be discussed more in the "Model Performance" section later on.

Abstracts are a feature that were only available for the V2 model and it was expected that this would have as significant of a contribution as the paper title, if not more. The following figure shows the statistics for the number of tokens found in the abstracts:

Average	112
Standard Deviation	111
Minimum	5
Maximum	2123

Table 2: High Level Statistics for Number of Tokens in Abstracts

The last thing to explore are the tags themselves. All tags are assigned a "Level" with level 0 being the highest and level 5 being the lowest. In total, there are over 700,000 tags in the historical MAG data. The following table shows how many tags there are for each level:

Tag Level	# of Tags	% of Papers with Tag
0	19	99.4
1	292	99.6
2	159323	95.6
3	357425	80.6
4	124331	43.3
5	77303	34.1

Table 3: Concept Level Distribution

On average, a paper is tagged with 7-8 concepts. The following figure shows a distribution of how many concepts are tagged to each paper:

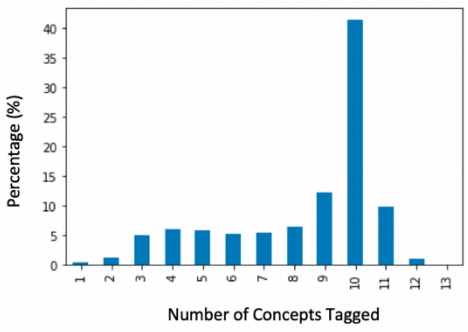


Figure 4: Distribution of Number of Concepts Tagged in MAG Data

In order to reduce the number of possible concepts and more effectively predict each tag, a threshold was chosen so that rarely seen concepts are not an option for prediction. This threshold was based on the number of papers with which the concept is tagged. The following figure shows what the number of available concepts would be for different threshold limits up to 500.

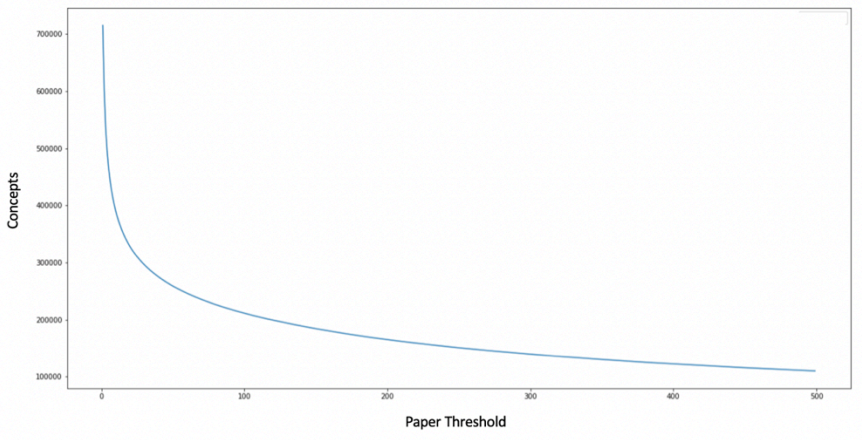


Figure 5: Concepts Available After Applying Threshold

This means that if a paper threshold of 500 is set, any concept that appears in less than 500 papers will not be included in the modeling. Doing this allows the model to focus on more important topics and it gives the model a sufficient number of samples in the training data for each concept. If the threshold is kept too low, the model would receive a number of training examples with rarely-seen concepts and would most likely still not predict those concepts during inference.

3 Model Methodology and Architecture

Modeling Decisions

Out of the data exploration and through discussions with other members on the OpenAlex team, there were many decisions made which ended up impacting the final model.

- Using concepts seen in 500 papers or more and have a wikidata ID
    - To reduce the total number of concepts that the model has to predict, a cutoff of 500 papers was implemented. This means any concepts that did not appear in 500 papers or more were removed from the model entirely, reducing the taggable concepts from a little over 700K to around 65K.

- Only training on papers that are in English
    - In order to take care of the majority of papers, only papers written in English were used to train the model. This accounts for over 70% of the historical papers contained in the MAG data.

- Not training on papers where “Patent” is the document type

- This was a decision made by the OpenAlex team to focus on papers that show up in other document types because the concepts for patent papers have a different distribution than journal papers and leaving patent papers in the training data could lead to the model learning incorrect relationships.

In addition, this model is meant to replicate the MAG model and is trained using MAG historical tagging data. This means that the model will not be able to assign new concepts or create new concepts based on newly trending topics or fields of study. This should not be an issue in the short term but in the long term there might be model degradation due to the fact that there are new topics emerging in academia over time.

Modeling Methodology

One of the first things that needed to be decided is how to turn the paper title and abstract into a numerical/vector representation (embedding) that could be consumed by the model. See (Dieng et al., 2020) and (Levy & Goldberg, 2014) for more information on embeddings and their usefulness.

There are many tokenization methods that could be used to accomplish this such as byte-pair encoding (BPE), word-piece tokenization, or sentence-piece tokenization. For the V1 and V2 model, a basic word tokenizer was used to simplify the preprocessing and also so that the words/embeddings could be explored after model training to confirm the model has learned the correct relationships between words. Since there are many different words that could be found in the title and abstract of an academic paper, some type of cutoff had to be implemented so that the vocab can be reduced to a reasonable size for the model. A cutoff of 300 was chosen so that any word appearing less than 300 times in paper titles and abstracts in the training data were not used as a token in the vocab.

From the data exploration, the following features were chosen for the model:

- Paper title
- Document type (Journal, Book, Conference, etc.)
- Journal Title
- Abstract (V2 only)

A baseline model was created which used only the paper title and a small neural network to predict what the tags would be. The paper title was fed into the neural network as a "Bag-of-Words" (BOW) representation which means the model was only able to distinguish the different tokens/words that were in the paper title, not the order they were in. This prevents the model from learning any context from the order of words. In most cases, this can be a significant drawback to understanding a sentence or paper title. The V1 model used both a BOW representation of the title as well as a sequential representation that was fed into a small multi-head attention network. The V1 model eventually evolved into the network seen in the architecture diagram below.

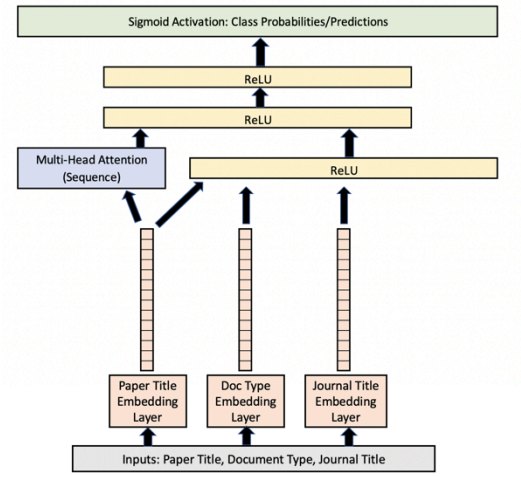


Figure 6: V1 Model Architecture

Once the abstract data was added to the model, the final V2 architecture settled on the following:

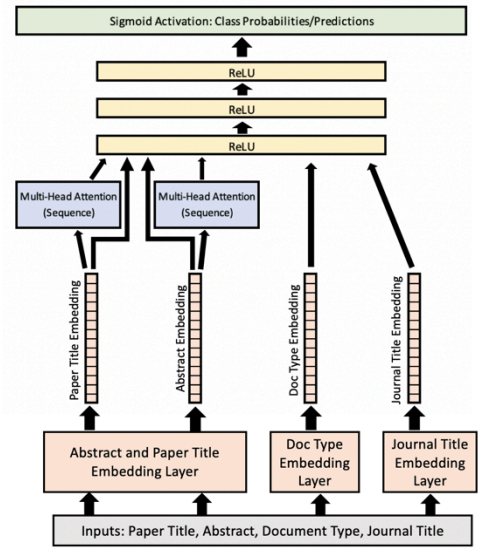


Figure 7: V2 Model Architecture

The document type and journal (if available) are both fed into an embedding layer and sent through the fully-connected ReLU layers of the network. The paper title and abstract are both fed into an embedding layer but is then sent down two different paths: a fully-connected ReLU layer as well as a Multi-Head Attention Layer. This helps to generalize the model beyond what either the BOW or sequential representation part of this neural network would be able to do.

The Multi-Head Attention output and other features were concatenated and sent through three fully-connected ReLU layers before being sent through a final sigmoid activation layer where the probabilities/predictions could be found. Although this model architecture was decided upon over many iterations, it took additional time and testing to come up with the exact configuration

through hyperparameter tuning and training. Both of these topics will be discussed in more detail in the next section.

4 Model Hyperparameters and Training

After removing patents from the data, there were around 100 million papers left with which to work. The first step was to split up the data based on dates. For training and validation, only data up to May 2021 was used while everything after this month was used in the test set. This was done to make sure that there was no data leakage into the test set when trying to determine how the model would perform on brand new data. The data from before June 2021 was then split into 99.5-0.45-0.05 size datasets that could be used for training, validation, and testing purposes, respectively. The final dataset sizes are as follows:

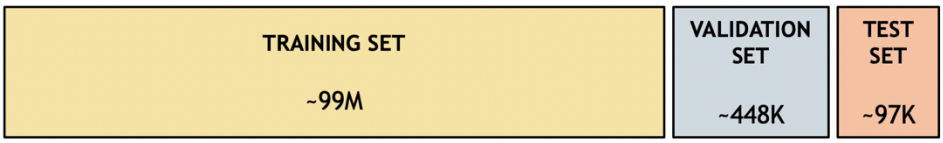


Figure 7: Train/Val/Test Dataset Sizes

To clarify, the test dataset that is used for calculating accuracy/metrics is a combination of data from the training period (up to June 2021) as well as all data starting in June 2021. In the Model Performance section, the differences in model performance between these 2 time periods will be observed to make sure there is little to no degradation.

Due to the unbalanced nature of the classes (concepts) the model was trained using Focal Loss which tries to give less weight to classes that show up more frequently and are easier to predict, seen in (Lin et al., 2020).

With Focal Loss being used for this model, there were now 7 hyperparameters that need to be tuned (search spaces shown):

- Number of heads (in multi-head attention): 2 - 12
- Number of layers (in multi-head attention): 2 - 6
- Number of nodes in each ReLU layer: 256 - 2048
- Learning rate (for ADAM optimizer): 0.01 – 0.0001
- Alpha (focal loss): 0.25, 0.75
- Gamma (focal loss): 1.5 – 3.5
- Batch size: 32 - 1024

Experiments were done with various configurations while tracking loss and other key metrics (precision, F1 score) in order to determine which configuration worked the best. Some hyperparameters were dependent on the compute resource while others could be changed without

worrying about Out-of-Memory (OOM) issues. After the experiments were done, the final model configuration used for modeling was as follows:

- First dense layer size: **2048**
- Second dense layer size: **1024**
- Third dense layer size: **1024**
- Number of heads (multi-head attention): **6**
- Number of layers (multi-head attention): **4**
- Initial Learning Rate: **0.006**
- Alpha: **0.25**
- Gamma: **2.8**
- Batch size: **1024**

While the initial learning rate is set, a learning rate schedule is also created to decrease the learning rate after six epochs so that the loss continued to steadily decrease throughout training. The following figure shows an example of learning rate decay:

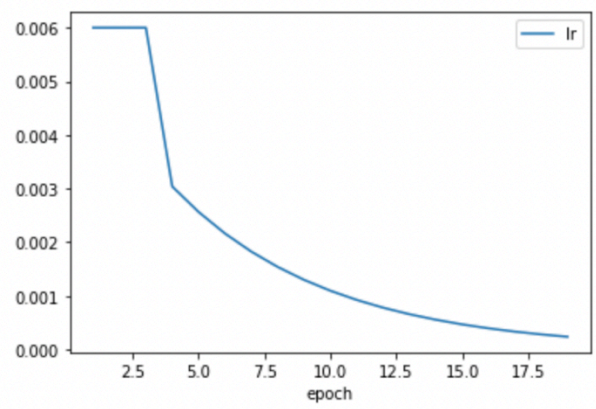


Figure 8: Learning Rate Decay Example

With the configuration set, the model was trained for 10 epochs using an AWS GPU EC2 Machine (p3dn.24xlarge). The training took multiple days to finish.

5 Model Performance

In order to determine how well the model was performing across many different iterations and architectures, it was important to keep track of key metrics. As mentioned previously, precision is favored over recall because there needs to be confidence that when a paper is tagged with a concept, the concept makes sense. Therefore, precision was prioritized in the modeling process. In addition to precision, recall and F1 score was also tracked in order to get a complete picture of how the model was performing.

To score each model, the same test set was used throughout the entire process ensuring that metrics were directly comparable. A benchmark model was also used as a baseline to make sure that the V1 and V2 model were outperforming the most basic classifier that could be created. The benchmark model used only the word tokens from the paper title passed through a couple feed-forward layers in a neural network.

The following sections go through the different subsets of test data that were looked at as well as the overall performance for the final model compared to the baseline model. All metrics reported (precision, recall, F1) are unweighted.

Overall Performance (Baseline vs V1 vs V2)

After taking all of the test data and running the metrics for both the baseline, V1, and V2 models, the following performance was observed:

Metric	Baseline Model	V1 Model	V2 Model
Precision	0.335	0.425	0.597
Recall	0.275	0.462	0.683
F1 Score	0.302	0.443	0.637

Table 4: Overall Model Performance

What is seen here is the V2 model clearly outperformed the baseline and V1 model, as expected. While these metrics are used to compare models against each other and determine the best model, the true test comes from sampling the data and comparing the predicted concept tags with the MAG tags.

Concept/Tag Levels (V2 Model)

In order to get a better idea of how the model is performing on a concept level, the data was split up into levels and the metrics were calculated using only the concepts in that level.

- % of papers with tag - percentage of papers that MAG assigned a tag for that level
- % of papers with prediction - percentage of papers that the model assigned a tag for that level
- Precision, Recall, F1 Score

The metrics can be seen here:

Level	% of Papers with Tag	% of Papers with Prediction	Precision	Recall	F1 Score

0	99.5	99.1	0.659	0.873	0.751
1	99.5	95.5	0.468	0.714	0.565
2	93.2	93.1	0.614	0.662	0.637
3	74.9	73.6	0.579	0.602	0.590
4	43.3	41.3	0.510	0.531	0.521
5	34.1	36.5	0.529	0.562	0.545

Table 5: Model Performance at Each Level

As expected, the model mostly gets less accurate as it tries to predict concepts from the bottom of the tree (level 3-5) compared to the top level. This is not surprising because those levels also contain the highest number of tags as opposed to level 0 and level 1 which have 19 and 292 concepts, respectively.

Document Type (V2 Model)

Another view that was looked at was the different documents types to make sure that the model could handle the different publication methods that were found in the historical MAG data.

Document Type	Precision	Recall	F1 Score
Journal	0.602	0.689	0.643
Conference	0.573	0.650	0.609
Repository	0.593	0.700	0.642
Book	0.537	0.569	0.553
Book Chapter	0.530	0.597	0.561
Thesis	0.543	0.608	0.574

Table 6: Model Performance for Each Publication Type

One of the things that jump out here is the fact that Journal and Repository both have higher scores than the remaining document types. It is unclear what is leading to this increase but there is something about those document types that leads to higher metrics. One theory is that those publication methods have paper titles that are more well-defined and easier to understand compared to the others. Also, if a paper is released in a repository, it most likely means that code is involved which might make the paper topics easier to guess.

Different Top-Level Concepts (V2 Model)

Another area to look into to determine model performance was the top-level (level 0) concepts. There are 19 level 0 topics. The following table gives a rough indicator of how each concept performed using the F1 score as the metric:

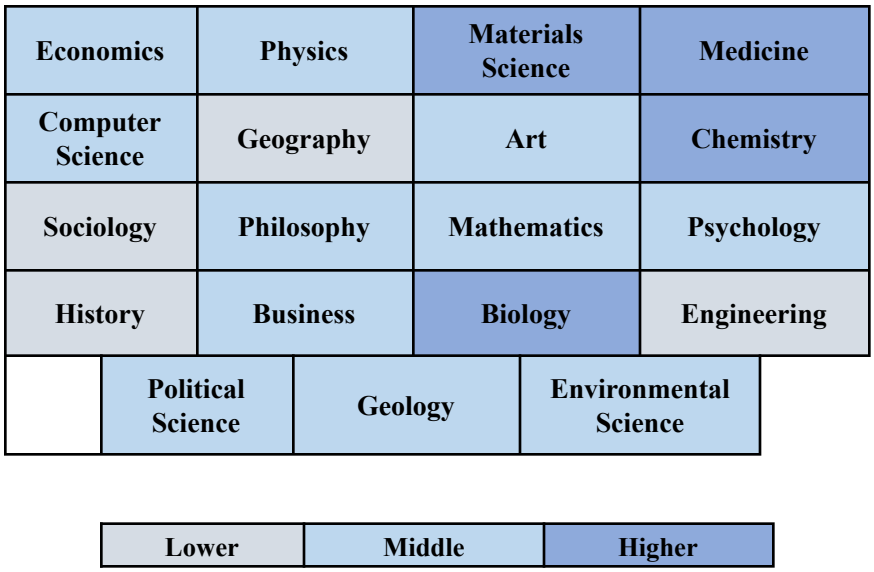


Figure 10: Heat Map of Level 0 Performance (F1 Score)

Paper Title Length (V2 Model)

Paper title length was previously mentioned in the data exploration section so it was something that deserved an analysis to see how different paper title lengths performed:

Paper Title Length (Words)	% of Data	Precision	Recall	F1 Score
Less than 10	27.3	0.581	0.653	0.615
10 - 20	61.8	0.601	0.692	0.643
20-40	10.6	0.611	0.699	0.652

Table 7: Model Performance for Different Paper Title Lengths

There seems to be a trend that as paper length increases, the precision increases slightly and the recall shoots up, which raises the F1 score. This means that with longer paper titles (more information) the model is getting better at predicting concepts, which is expected happen.

Time (V2 Model)

Another thing to look at was the effect of time on the predictions and seeing if there is a drastic drop in the performance of the model.

Time Period	Precision	Recall	F1 Score
Before 2003	0.605	0.669	0.635
2003 - 2008	0.605	0.681	0.641
2009-2014	0.606	0.676	0.639

2015-2020	0.597	0.686	0.639
2021 (Up to June)	0.560	0.688	0.617
2021 (June and Beyond)	0.582	0.704	0.637

Table 8: Model Performance for Different Time Periods

Looking at the metrics over time shows that the model performs about the same at all dates in the past. The model was only trained on data up until June 2021 so not having any model degradation for papers published after that date is a huge accomplishment. While the model performs well for this data, it is expected that over time it will degrade due to newer topics emerging that the model was not trained on.

Abstracts (V2 Model)

In order to determine how important of a role the abstracts play in tagging concepts, performance of papers with an abstract were compared to papers without an abstract:

Abstract?	Precision	Recall	F1 Score
Yes	0.605	0.696	0.647
No	0.561	0.621	0.589

Table 9: Model Performance for Papers with Abstract Data vs No Abstract Data

It is expected that the model would perform better at tagging a paper if there is abstract data available so the metrics seen in this table are not surprising.

Metrics are important but at the end of the day the only thing that matters is how the model performs at tagging concepts compared to the MAG tagging. The context of the test data must be remembered because the source of truth in this situation are the tags assigned by MAG. In reality, these tags may be incorrect or incomplete which would lead to lower metrics/scores being reported. In the next section, we will go over specific examples and how in some cases the V2 model's concept tagging performed better than the MAG tagging.

6 Model Examples

In order to show how exactly the model is performing compared to the MAG tagging, some examples are shown below. All of the tags were split up into 3 groups for each paper:

- MAG and Model Concepts Match: Concepts that were tagged by both MAG and the model (True Positives)
- Missed MAG Concepts: Concepts that were tagged by MAG but the model did not end up predicting (False Negatives)
- Extra Model Concepts: Concepts that were predicted by the model but were not tagged by MAG (False Positives)

Examples

Paper Title:	*Design of a two-Degree-of-Freedom Controller for a Magnetic Levitation System Based on LQG Technique*		
MAG and Model Concepts Match:	magnetic levitation	nonlinear system	control theory
	control engineering	controller	engineering
	magnetic bearing	Matlab	linear quadratic gaussian control
Missed MAG Concepts:	Kalman filter	optimal projection equations	
Extra Model Concepts:	computer science	tracking	

Table 10: Example 1

The example in Table 10 shows a largely successful tagging by the V2 model. While the two missed concepts might help further explain what is in the paper, the more important fact is that the V2 tagger did not incorrectly predict a tag. Both "computer science" and "tracking" are correct, relevant topics for this paper so it could be said that the model achieved near perfect precision even though the concepts don't line up exactly with the MAG tagging.

Paper Title:	*Enhanced Indexation via Chance Constraints*		
MAG and Model Concepts Match:	investment strategy	benchmark	computer science
	probabilistic logistic	portfolio	set
	computational intelligence	mathematical optimization	
Missed MAG Concepts:	indexation	expected value	
Extra Model Concepts:	portfolio optimization	expected shortfall	tracking error
	ranking	econometrics	

Table 11: Example 2

In the Table 11 example, the model does a great job of predicting additional tags to go with the paper that MAG did not assign. This would lead to a much lower precision score when calculating the metrics but it could be argued that the "true" precision for this example is much higher.

Paper Title:	*Role of Silicon Content and Final Annealing Temperature on Microtexture and Microstructure Development in Non-Oriented Silicon Iron*		
MAG and Model Concepts Match:	materials science	silicon	microstructure
	electrical steel	metallurgy	annealing
	grain size	grain growth	
Missed MAG Concepts:	hot rolled		
Extra Model Concepts:	recrystallization		

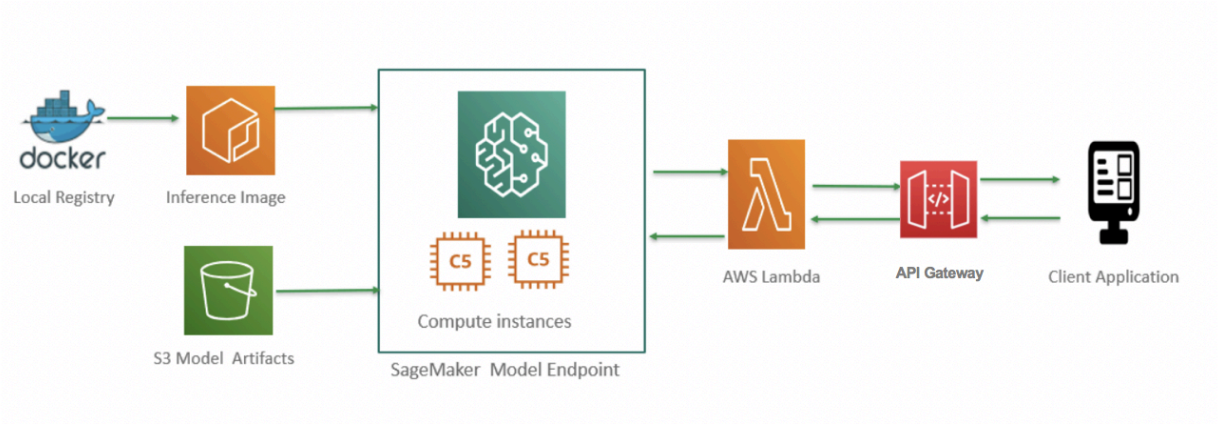
Table 12: Example 3

In this final example, the V2 model achieves a near perfect score compared to the MAG and the additional tag is extremely relevant to the paper. Therefore, it only missed one term which could be considered a huge success.

These few examples show that the calculated precision might be lower than the "true" precision when looking at the examples on a case by case basis. While the MAG concept tagger is powerful since it is able to produce new concepts, the OpenAlex V2 concept tagger does a great job at replicating the MAG tagging and also providing additional, relevant tags for academic papers.

7 Model Deployment and Throughput Testing

In order to effectively use the model in production, AWS was chosen as a service provider to host the model and API endpoint. The following is an architecture diagram to show which services were used and how everything is connected.



The steps taken to deploy the model are summarized as follows:

1. Used Docker to create a container which contains the inference code as well as the serving code used by AWS SageMaker
2. Uploaded the newly created image to AWS ECR (Elastic Container Registry)
3. Uploaded the model artifacts to AWS S3
4. Created an AWS SageMaker endpoint using the image and model artifacts
5. AWS Chalice was used to quickly create a REST API using AWS API Gateway and AWS Lambda

Once these steps were completed, requests could be made to the API to tag papers using the model. Different configurations can be set up in AWS SageMaker to scale appropriately with the number of requests being made to the API.

Throughput Testing

In order to make sure model inference time would be acceptable, tests were run to check the throughput and latency of the model when it is served as a REST API through AWS. Since the model is served using AWS Sagemaker, the throughput can be increased by using larger instance sizes and increasing the number of instances. However, it is still important to know the latency for a single call to the model so estimates can be made later on for the time it will take to score a batch of data.

Using the Python requests library, the API was tested and on average a call to the model took 155ms to respond. The model is able to accept both single and batch input so in order to maximize throughput for a given resource, it is important to send batch calls to the model. In performance testing, batch size was increased to the maximum size without getting an Out-of-Memory (OOM) error while also using the Python multiprocessing library to make concurrent calls to the model.

8 Conclusion

This document describes the end-to-end process for developing and deploying a model to imitate MAG tagging. While this model does not provide as much tag coverage as MAG, the model

achieves good accuracy at assigning tags to papers and can be confidently used for academic purposes. For any questions, feel free to reach out to jason@ourresearch.org. All of the code used to create this model can be found on GitHub at: https://github.com/ourresearch/openalex-concept-tagging.

References

Dieng A. B., Ruiz F. J. R., Blei D. M.; Topic Modeling in Embedding Spaces in Transactions of the Association for Computational Linguistics 2020; 8 439–453.

Levy, O., & Goldberg, Y.; "Dependency-based word embeddings" in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics 2014 Volume 2: Short Papers pp. 302-308.

Lin, T., Goyal P., Girshick R., He K. and Dollar P., "Focal Loss for Dense Object Detection" in IEEE Transactions on Pattern Analysis & Machine Intelligence, 2020 vol. 42, no. 02, pp. 318-327.

Liu, T., Zhang, N. and Chen, P.; "Hierarchical latent tree analysis for topic detection" in European Conference on Machine Learning and Knowledge Discovery in Databases, 2014 pp. 256–272

Wang, K., Shen, Z., Huang, Wu, C.-H., Dong, Y, Kanakia, A.; "Microsoft Academic Graph: When experts are not enough" in Quantitative Science Studies 2020; 1 (1): 396–413.

Wang, K., Shen, Z., Huang, C.-Y., Wu, C.-H., Eide, D., Dong, Y., … Rogahn, R.; "A Review of Microsoft Academic Services for Science of Science Studies" in Frontiers in Big Data, 2019 2, 45.