Infinite Regress: A Scalable System for Graphing, Clustering and Multi-Document Summarization of Academic Articles

Tran Duc Khang, Kannan Vishal
NUS High School of Math and Science

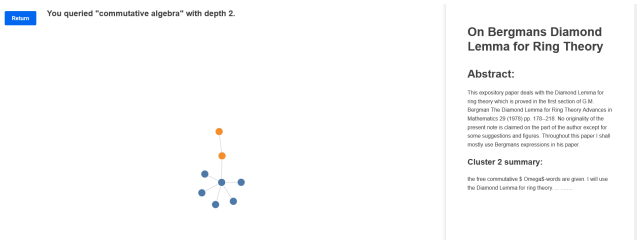

Figure 1: Sample output from https://regress.shirator.net/

Introduction

Our system constructs a citation graph from the user's query, it then performs clustering on the graph data and does multi-document summarization on each cluster to provide the user with a high level summary of each cluster.

How to use

The application can be viewed at https://regress.shirator.net/. Depth 2 is recommended, as depth 1 is quite primitive and depth 3 might fail. Generating the graph might take around 30s-1min. If it fails the first time it may be worth reloading the page. Avoiding reloading the page too much as this may overload the server.

Related Works

Connected Papers is a platform that offers similar citation graph visualization feature as our platform however, the graph generated is constrained to only those supplied by Semantic Scholar's API. Though extensive, this API is not absolute. Realizing this limitation we have an in-built ad-hoc pdf citation extraction module, this allows our software to work with more papers than what Semantic Scholar provides. Furthermore, Connected Papers offer no support for clustering of academic papers nor does it have a multi-document summarization (MDS) engine.

Litmaps is another platform that offer similar functionalities to ours. Though it is not constrained to only one database, it still lacks clustering and MDS functionalities.

Naive Approach

When tasked with the problem of clustering academic papers, our first instinct was to generate a pool of papers based on the user's query, then create a tf-idf matrix to obtain document vectors, which we will use to cluster the papers.

We tried using both hierarchical and kmeans clustering to unremarkable results. The quality of the clusters was so bad that pulling two random sample of two abstracts belonging to the same cluster showed that they were on completely different things, though they notably used a lot of similar words.

```
Sample Article 1:
Multi-task learning is motivated by the observation that humans bring to bear
what they know about related problems when solving new ones. Similarly, deep
neural networks can profit from related tasks by sharing parameters with other
networks. However, humans do not consciously decide to transfer knowledge
between tasks. In Natural Language Processing (NLP), it is hard to predict if
sharing will lead to improvements, particularly if tasks are only loosely
related. To overcome this, we introduce Sluice Networks, a general framework
for multi-task learning where trainable parameters control the amount of
sharing. Our framework generalizes previous proposals in enabling sharing of
all combinations of subspaces, layers, and skip connections. We perform
experiments on three task pairs, and across seven different domains, using data
from OntoNotes 5.0, and achieve up to 15% average error reductions over common
approaches to multi-task learning. We show that a) label entropy is predictive
of gains in sluice networks, confirming findings for hard parameter sharing and
b) while sluice networks easily fit noise, they are robust across domains in
practice.
Topics: ['stat.ML', 'cs.AI', 'cs.CL', 'cs.LG', 'cs.NE']
Sample Article 2:
Learned feature representations and sub-phoneme posteriors from Deep Neural
Networks (DNNs) have been used separately to produce significant performance
gains for speaker and language recognition tasks. In this work we show how
these gains are possible using a single DNN for both speaker and language
recognition. The unified DNN approach is shown to yield substantial performance
improvements on the the 2013 Domain Adaptation Challenge speaker recognition
task (55% reduction in EER for the out-of-domain condition) and on the NIST
2011 Language Recognition Evaluation (48% reduction in EER for the 30s test
condition).
Topics: ['cs.CL', 'cs.CV', 'cs.LG', 'cs.NE', 'stat.ML']
```

Figure 2: Two abstracts from the same cluster

We find that the first abstract is about a new multi-task learning system while the second abstract is about using a unified DNN approach to improve speaker and language recognition tasks

With this we conclude that clustering merely on semantic similarity is not fine enough since our pool of articles are very similar semantically seeing that they were fetched based on the user's query.

Citation Graph

As the naive approach failed, we decided to exploit the citations of paper as a means to gather data as well as cluster papers. [Insert reason why we chose to do this]

Semantic Scholar API

Our first instinct was to cherry pick an API that allowed us to query not only a paper but get its citations as metadata. We eventually stumbled upon the Semantic Scholar API which offered exactly what we needed, however we soon realize that the API imposed a hard call rate limit meaning that we would only be able to fetch 100 papers in 5 minutes.

Since that is highly undesirable we decided go through the trouble of implementing a citation extraction system that could extract citations accurately from a paper's pdf (which is the format in which most digitized papers are in). Of course this would also mean that our system is no longer limited to any one academic literature database.

Citation Extraction System

We chose to use arxiv's API for fetching our paper pdfs as they offered a generous rate limit of 4 calls per second.

Document Preprocessing After retrieving the pdf file of the paper, we pass it through Apache Tika to obtain a text parse. Then we segment the text parse into Header, Abstract, Body, Reference, and Appendix (if any) based on textual cues (i.e. headings).
The references section are then isolated and split on '\n\n' into individual reference 'chunks' where each chunk supposedly contains a literature reference. Though this method is quite course so some reference chunks might not even be legitimate references, as such our system implements a reference chunk identification routine

Reference Chunk Classification Our goal is to train a model that can accurately distinguish between legitimate and illegitimate reference chunks. We collected the data needed to train our model by running the previous routine on a large number of papers, in the end we got around 7000 reference chunks.

Since we do not want to manually label 7000 examples, we turned towards semi-supervised approaches which allowed us to build robust models off of a largely unlabelled dataset. In total, we were able to label 1382 positive examples and 1783 negative examples. We held out 25% of the labeled examples for testing.

Before training the model we did feature extraction. Through manual examination of legitimate and illegitimate reference chunks we notice a discrepancy in the number of named-entities present, as well as the number of nouns, verbs, punctuation, adjectives, proper nouns, adpositions, and symbols along with the word count of the chunks. As such we vectorized reference chunks using these bins along with another "others" bin for completeness. The named-entity recognition, as well as part-of-speech tagging required for this vectorization was done using spaCy.

Moving on, "No Free Lunch" theorem dictates that we should try out all available models before declaring the best one. As such, we trained and evaluated all of scikit-learn's semi-supervised models which includes the SelfLearningClassifier (using a SVC core), LabelPropagation, and LabelSpreading on the entire dataset as well as a supervised model (i.e. Logistic Regression) on only the labeled examples. The results are in the following table.

	precision	recall	f1-score	support
0	0.90	0.95	0.92	448
1	0.93	0.86	0.89	344
accuracy			0.91	792
macro avg	0.91	0.91	0.91	792
weighted avg	0.91	0.91	0.91	792

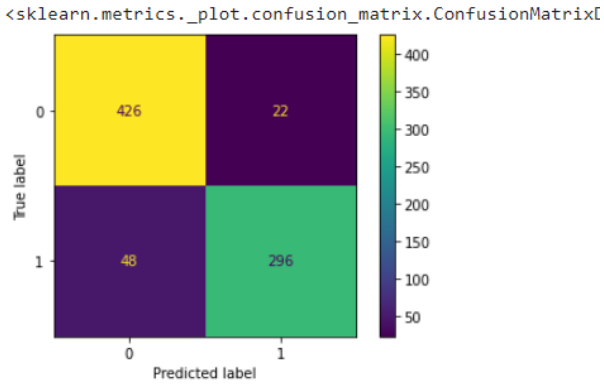

Figure 3: Logistic Regression model for Reference Chunk Classification

	precision	recall	f1-score	support
0	0.96	0.86	0.91	448
1	0.84	0.96	0.89	344
accuracy			0.90	792
macro avg	0.90	0.91	0.90	792
weighted avg	0.91	0.90	0.90	792

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixI

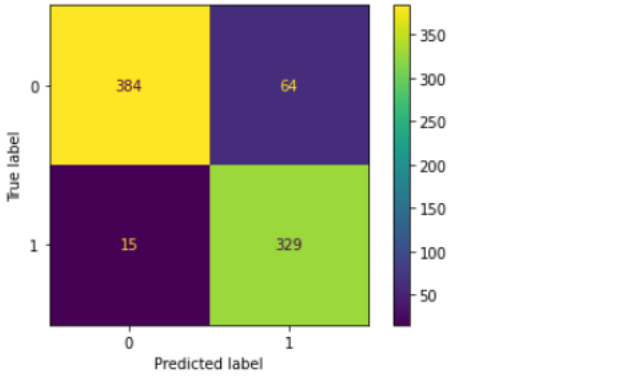

Figure 4: Self Training Classifier model with SVC core for Reference Chunk Classification

	precision	recall	f1-score	support
0	0.94	0.93	0.94	448
1	0.91	0.93	0.92	344
accuracy			0.93	792
macro avg	0.93	0.93	0.93	792
weighted avg	0.93	0.93	0.93	792

/usr/local/lib/python3.7/dist-packages/sklearn/semi_supe
probabilities /= normalizer
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixD

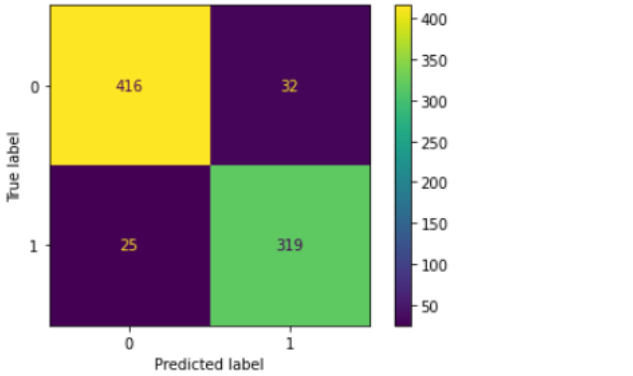

Figure 5: Label Propagation model for Reference Chunk Classification

	precision	recall	f1-score	support
0	0.93	0.93	0.93	448
1	0.91	0.92	0.91	344
accuracy			0.92	792
macro avg	0.92	0.92	0.92	792
weighted avg	0.92	0.92	0.92	792

/usr/local/lib/python3.7/dist-packages/sklearn/semi_sup
probabilities /= normalizer
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixI

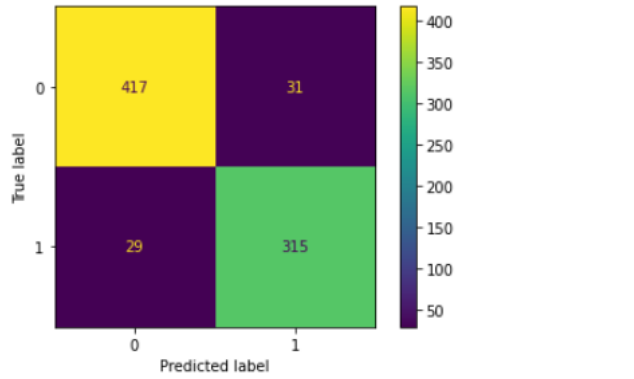

Figure 6: Label Spreading model for Reference Chunk Classification

Although we care more about minimizing false positives than minimizing false negatives for this particular application, we went with the LabelPropagation model over the Logistic Regression model because the discrepancy in the number of false positives between the two models was slight with respect to the total number of examples used for testing while the true positives of label propagation was much higher. Furthermore, we believe that our paper fetching routine is robust enough to handle this slight margin of error.

Paper Name Extraction

After having done all that, we should be left with mostly legitimate reference chunks. The problem now is that we need a way to extract the name of the paper out of the reference chunk which also includes the authors' names as well as other supporting information such as year of publication, publication link, etc...

Since not all reference chunks are formatted the same way, some might have the year before the title, some might have the authors' names after the title, we decided that this is another task for machine learning.

We note the fact that all reference chunks have their information separated by '', thus we split each chunk by '' and the problem is reduced to identifying which "sentence" is a valid paper name. We note that the discrepancy in the frequency of named-entity, proper nouns, and the aforementioned bins is even more stark in this case as anything that is not a paper name either has a bunch of named-entities

and punctuations in it (i.e. author list) or filled with symbols and numbers (i.e. link or year). Thus we should expect the paper name extraction model to perform even better than the reference chunk classification model.

And that is indeed the case. Similar to reference chunk classification, we labelled a small subset of our example and utilized a semi-supervised model and tested it on a hold-out set. However, unlike the previous task, for this one we only need to evaluate one model (in this case the SelfTraining-Classifier using an SVC core) as it gave more than perfect performance, scoring 100% for precision and 95% for recall.

	precision	recall	f1-score	support
0	0.95	1.00	0.98	100
1	1.00	0.95	0.97	100
accuracy			0.97	200
macro avg	0.98	0.97	0.97	200
weighted avg	0.98	0.97	0.97	200

<sklearn.metrics._plot.confusion_matrix.ConfusionMatri:

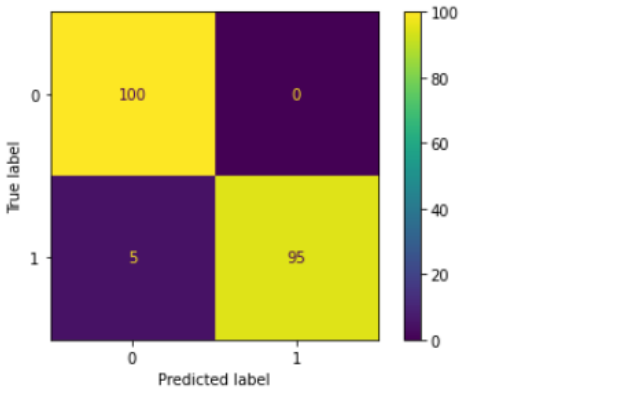

Figure 7: Self Training Classifier model with SVC core for Paper Nam Extraction

As you can see for the above confusion matrix, the model classified perfectly except for 5 miss-classified false negatives, or so we thought. On further inspection of the 5 so-called miss-classified examples, we find that 4 out of those 5 examples were indeed negatives and it was our manual labelling that was erroneous. A possible explanation for why it was able to do this might lie in the fact that the SelfTrainingClassifier had an SVC core, which was resistant to outliers.

```
62              https://doi.org/10.1145/1297846.1297897
105               https://scan.coverity.com/ [2] [n.d.]
157    GLUCOSE: GeneraLized and COntextualized story ...
180                    https://www.sonarqube.org/ [3] N
194                 http://arxiv .org/ abs/1910.03771 .
Name: 0, dtype: object
```

Figure 8: False negatives classified by model

Since this model performed exceptionally well, we did not find a need to experiment further with other models. "No Free Lunch" theorem does not apply as we have reached the upper bound of performance

Note: We went back and check to see if we have made any silly errors such as using the training set for testing and we couldn't find any.

Graph Generation and Document Clustering

For graph generation, we first take the user query and pass it to the API to get the paper that is most relevant to that query. Then we use multi-threaded breath-first search to grow the graph, in an effort to speed up computation time, we gave the user the option to limit the depth of the graph as well as the maximum edges per node.

For clustering we used Infomap as according to [ŠEW16][2] it (along side OSLOM, Metimeter and Louvain) is able to give high quality clusters that matched those done by actual experts in the field.

Entity Linking

We tried a number of methods such as using Chunking with NLTK which identifies parts of english grammar in a corpus allowing us to filter out possibly useful noun phrases. But this did not narrow the number of keywords enough. We thus tried transformers, namely the BERT model, which is able to embed the corpus based on semantics and relevance to the rest of the text, but it often picked out proper nouns which were too specific to be compared with other articles. In the end, we used a bag of words TFIDF model, which did decent in picking out the words. These keywords can later be associated with each document and cosine similarity of these keywords was used to determine the similarity of documents.

We also considered using the similarity of documents to weigh the edges between documents, i.e. make the graph links longer or shorter, but the similarity score is often close to one so we did not include it in the final model for optimization purposes.

Multi-Document Summarization

From literature review, we attempted to implement Mean-Sum model[CL18], which is an LSTM-RNN model which features two modules, one autoencoding module, to obtain a representation of the research article abstracts, and one summarization module, which produces a representation of the research articles abstracts which is as close to all the abstracts semantically. However, this model, while works as expected with numeric sequences, fails with word embeddings, possibly due to the complexity of the model. We thus

opted for using transfer learning where we used a pretrained transformer model by HuggingFace which was able to summarize articles to a reasonable degree.

Front-end

The logic code was compiled into a main python file (See backend.zip/citation-graph/citation-graph.py) and hosted on a DigitalOcean server. Using web backend frameworks, we created a REST API which will take a query and generate the graph. We also made a frontend frontend which will provide an interface where the user can make queries to the server and visualize the literature graph.

Challenges

Optimization

The AI models are quite slow as large corpuses of text have to be fetched from public APIs when generating the graphs. Many sites throttle public requests which results in this being a large bottleneck. We tried to circumvent this with multithreading, but as traversing the graph of citations is done through BFS, finding neighbours requires processing the references section at every node.

References

[ŠEW16] Lovro Šubelj, Nees Jan van Eck, and Ludo Waltman. "Clustering Scientific Publications Based on Citation Relations: A Systematic Comparison of Different Methods". In: *PLOS ONE* 11.4 (Apr. 2016), pp. 1–23. DOI: 10.1371/journal.pone.0154404. URL: https://doi.org/10.1371/journal.pone.0154404.

[CL18] Eric Chu and Peter J. Liu. "Unsupervised Neural Multi-document Abstractive Summarization". In: *CoRR* abs/1810.05739 (2018). arXiv: 1810.05739. URL: http://arxiv.org/abs/1810.05739.