# Game of Papers - Update

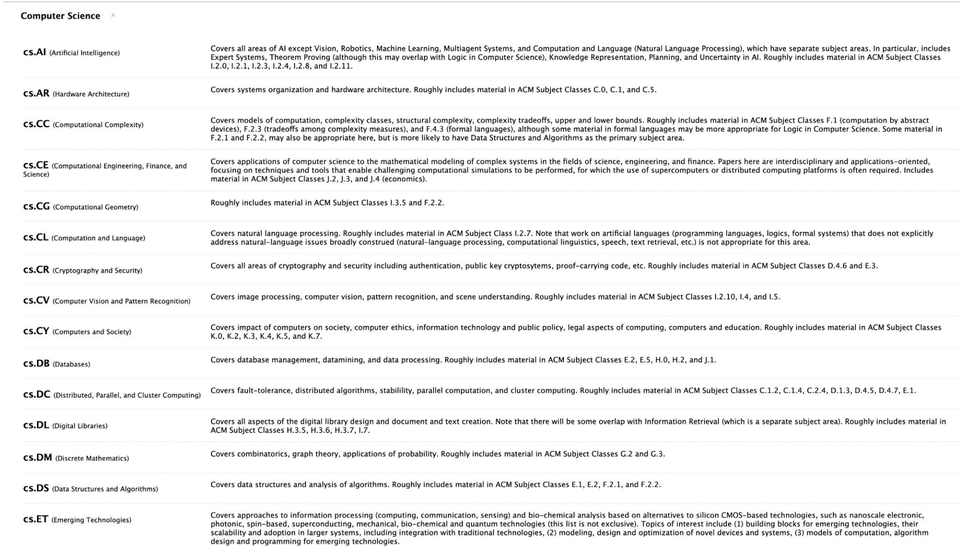
# Scripts for on processing data

**STAGE 1 scripts**: scripts for raw data gathering and parsing that output pickles in an intermediate dictionary-based representation. These will include mostly scripts that download files, parse HTML, etc.

**STAGE 2 scripts**: scripts that enrich the intermediate representations from STAGE 1 in various ways. For example, a script could iterate over publications in database, and if it finds that some entry is missing its pdf contents, integrated with semantic scholar API to populate incomplete data.

# Scripts for on processing data

Script to enrich data with fine grained tags so that we can build validation sets. This will help us calculate metrics to gauge accuracy.



**Computer Science**

| | |
|---|---|
| **cs.AI** (Artificial Intelligence) | Covers all areas of AI except Vision, Robotics, Machine Learning, Multiagent Systems, and Computation and Language (Natural Language Processing), which have separate subject areas. In particular, includes Expert Systems, Theorem Proving (although this may overlap with Logic in Computer Science), Knowledge Representation, Planning, and Uncertainty in AI. Roughly includes material in ACM Subject Classes I.2.0, I.2.1, I.2.3, I.2.4, I.2.8, and I.2.11. |
| **cs.AR** (Hardware Architecture) | Covers systems organization and hardware architecture. Roughly includes material in ACM Subject Classes C.0, C.1, and C.5. |
| **cs.CC** (Computational Complexity) | Covers models of computation, complexity classes, structural complexity, complexity tradeoffs, upper and lower bounds. Roughly includes material in ACM Subject Classes F.1 (computation by abstract devices), F.2.3 (tradeoffs among complexity measures), and F.4.3 (formal languages), although some material in formal languages may be more appropriate for Logic in Computer Science. Some material in F.2.1 and F.2.2, may also be appropriate here, but is more likely to have Data Structures and Algorithms as the primary subject area. |
| **cs.CE** (Computational Engineering, Finance, and Science) | Covers applications of computer science to the mathematical modeling of complex systems in the fields of science, engineering, and finance. Papers here are interdisciplinary and applications-oriented, focusing on techniques and tools that enable challenging computational simulations to be performed, for which the use of supercomputers or distributed computing platforms is often required. Includes material in ACM Subject Classes J.2, J.3, and J.4 (economics). |
| **cs.CG** (Computational Geometry) | Roughly includes material in ACM Subject Classes I.3.5 and F.2.2. |
| **cs.CL** (Computation and Language) | Covers natural language processing. Roughly includes material in ACM Subject Class I.2.7. Note that work on artificial languages (programming languages, logics, formal systems) that does not explicitly address natural-language issues broadly construed (natural-language processing, computational linguistics, speech, text retrieval, etc.) is not appropriate for this area. |
| **cs.CR** (Cryptography and Security) | Covers all areas of cryptography and security including authentication, public key cryptosytems, proof-carrying code, etc. Roughly includes material in ACM Subject Classes D.4.6 and E.3. |
| **cs.CV** (Computer Vision and Pattern Recognition) | Covers image processing, computer vision, pattern recognition, and scene understanding. Roughly includes material in ACM Subject Classes I.2.10, I.4, and I.5. |
| **cs.CY** (Computers and Society) | Covers impact of computers on society, computer ethics, information technology and public policy, legal aspects of computing, computers and education. Roughly includes material in ACM Subject Classes K.0, K.2, K.3, K.4, K.5, and K.7. |
| **cs.DB** (Databases) | Covers database management, datamining, and data processing. Roughly includes material in ACM Subject Classes E.2, E.5, H.0, H.2, and J.1. |
| **cs.DC** (Distributed, Parallel, and Cluster Computing) | Covers fault-tolerance, distributed algorithms, stabilility, parallel computation, and cluster computing. Roughly includes material in ACM Subject Classes C.1.2, C.1.4, C.2.4, D.1.3, D.4.5, D.4.7, E.1. |
| **cs.DL** (Digital Libraries) | Covers all aspects of the digital library design and document and text creation. Note that there will be some overlap with Information Retrieval (which is a separate subject area). Roughly includes material in ACM Subject Classes H.3.5, H.3.6, H.3.7, I.7. |
| **cs.DM** (Discrete Mathematics) | Covers combinatorics, graph theory, applications of probability. Roughly includes material in ACM Subject Classes G.2 and G.3. |
| **cs.DS** (Data Structures and Algorithms) | Covers data structures and analysis of algorithms. Roughly includes material in ACM Subject Classes E.1, E.2, F.2.1, and F.2.2. |
| **cs.ET** (Emerging Technologies) | Covers approaches to information processing (computing, communication, sensing) and bio-chemical analysis based on alternatives to silicon CMOS-based technologies, such as nanoscale electronic, photonic, spin-based, superconducting, mechanical, bio-chemical and quantum technologies (this list is not exclusive). Topics of interest include (1) building blocks for emerging technologies, their scalability and adoption in larger systems, including integration with traditional technologies, (2) modeling, design and optimization of novel devices and systems, (3) models of computation, algorithm design and programming for emerging technologies. |

## Further Exploring models

Started tinkering with the dataset, since this is a relatively novel problem statement. We need to establish some baselines using current models.

Also, did some experiments with semantic scholar API to extract relevant tags, features for recommendation tasks.

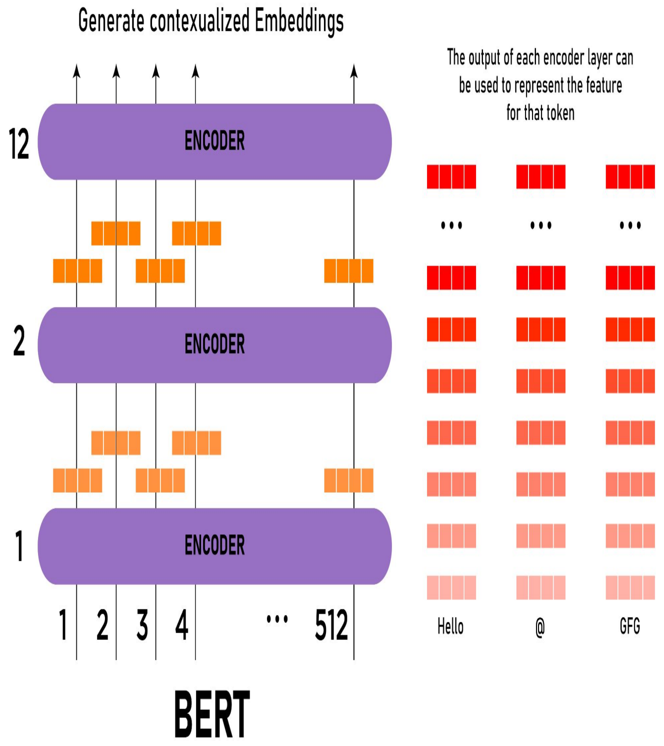
## Trying out recommendations using a few baseline models

1. Bert
2. Jaccard similarity
3. Tf-idf
4. Universal-sentence-encoder
5. SciBert

## Bert

BERT is based on the transfomer architecture. Specifically, BERT is composed of Transformer encoder layers.

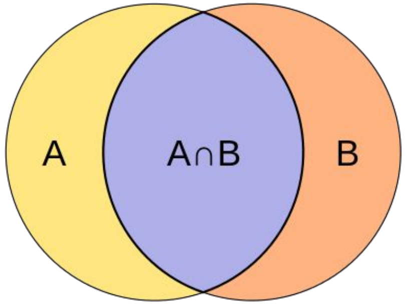
BERT was pre-trained on two tasks: *language modeling* (15% of tokens were masked and the training objective was to predict the original token given its context) and *next sentence prediction* (the training objective was to classify if two spans of text appeared sequentially in the training corpus).

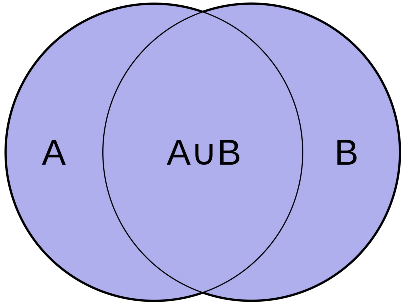

Generate contexualized Embeddings
The output of each encoder layer can be used to represent the feature for that token
12
ENCODER
2
ENCODER
1
ENCODER
1
2
3
4
...
512
Hello
@
GFG
BERT

## Jaccard similarity

The **Jaccard index**, also known as the **Jaccard similarity coefficient**, is a statistic used for gauging the similarity and diversity of sample sets.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$





## Tf-idf

In information retrieval, **tf–idf** (also **TF*IDF**, **TFIDF**, **TF–IDF**, or **Tf–idf**), short for **term frequency–inverse document frequency**, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.[1] It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling.

## Scibert

SciBERT is a BERT model trained on scientific text.

- SciBERT is trained on papers from the corpus of semanticscholar.org. Corpus size is 1.14M papers, 3.1B tokens. We use the full text of the papers in training, not just abstracts.
- SciBERT has its own vocabulary (scivocab) that's built to best match the training corpus. We trained cased and uncased versions. We also include models trained on the original BERT vocabulary (basevocab) for comparison.

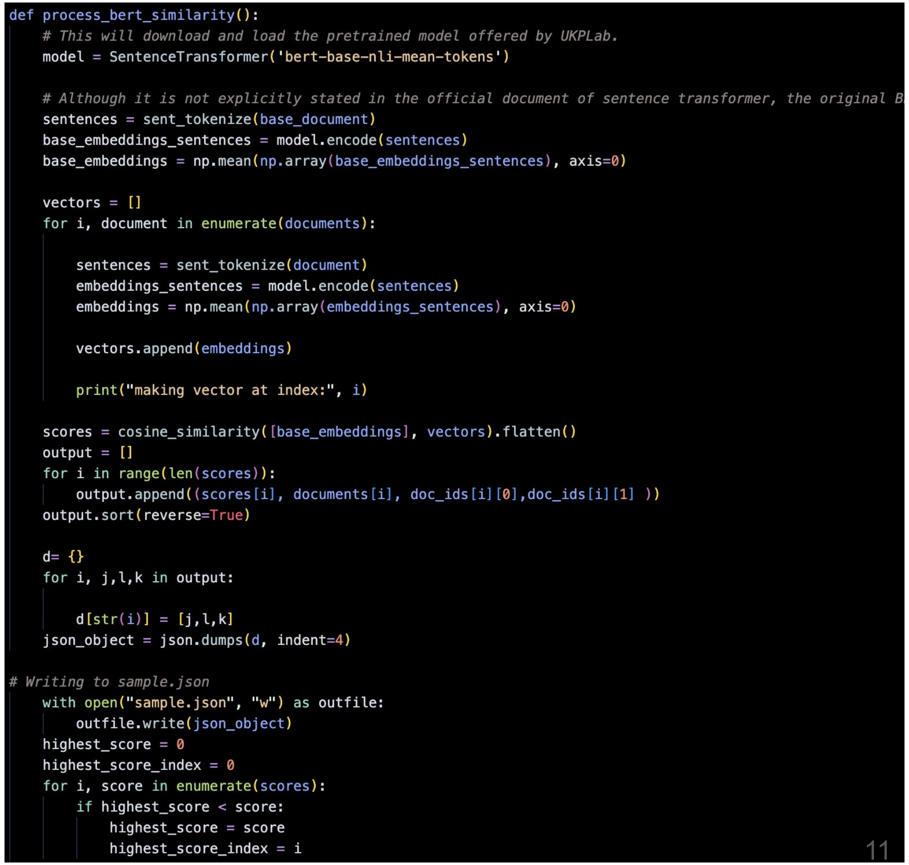
## Universal Sentence Encoder

The Universal Sentence Encoder encodes text into high dimensional vectors that can be used for text classification, semantic similarity, clustering, and other natural language tasks. The pre-trained Universal Sentence Encoder is publicly available in Tensorflow-hub.

It comes with two variations i.e. one trained with **Transformer encoder** and other trained with **Deep Averaging Network (DAN)**

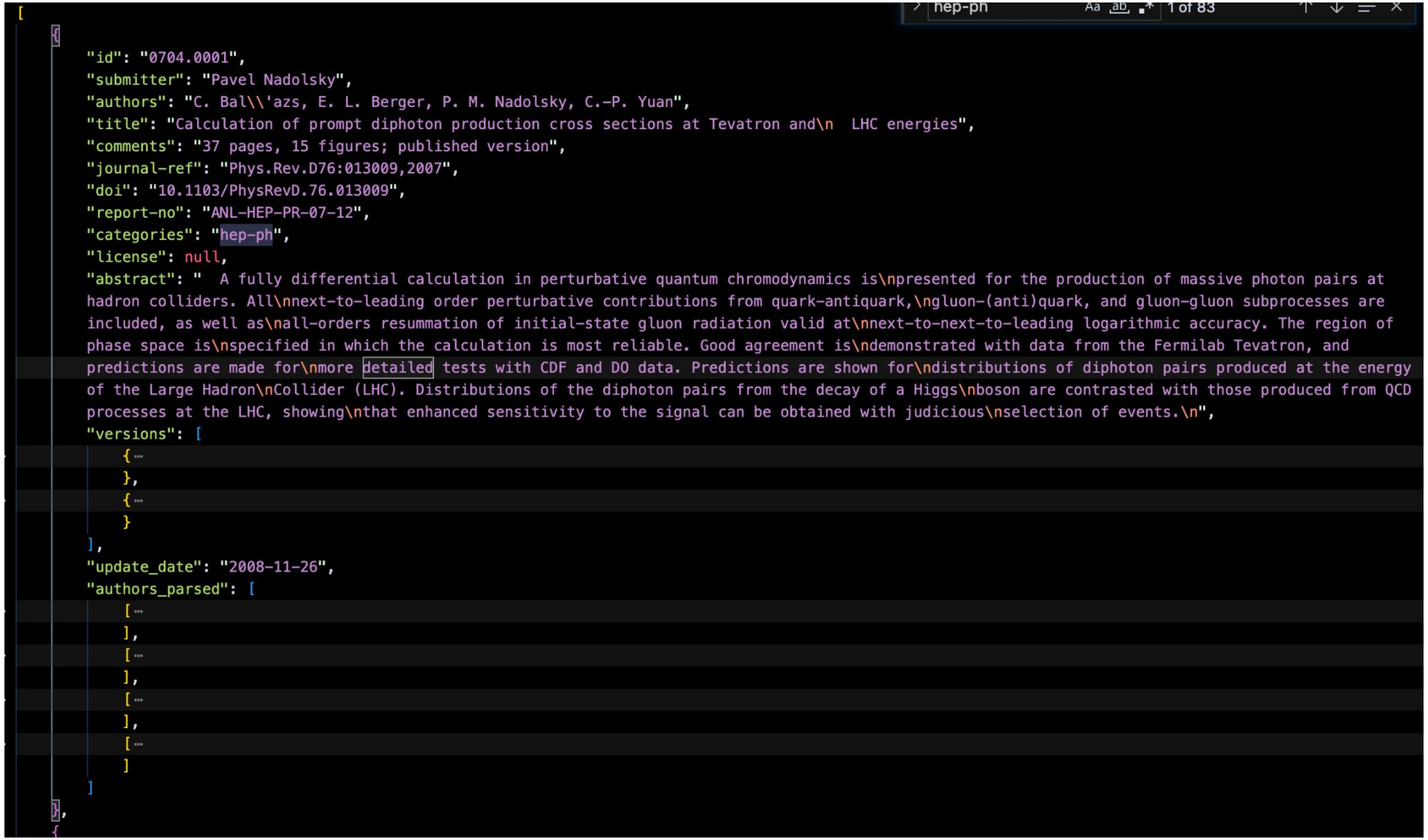
# Running an experiment

1. Used arXiv dataset and metadata of 1.7M+ scholarly papers across STEM
2. Randomly sampled 10,000 papers
3. Loaded the pretrained models, using the abstract and title – calculated the cosine similarity between papers and tried to establish a baseline for our system.



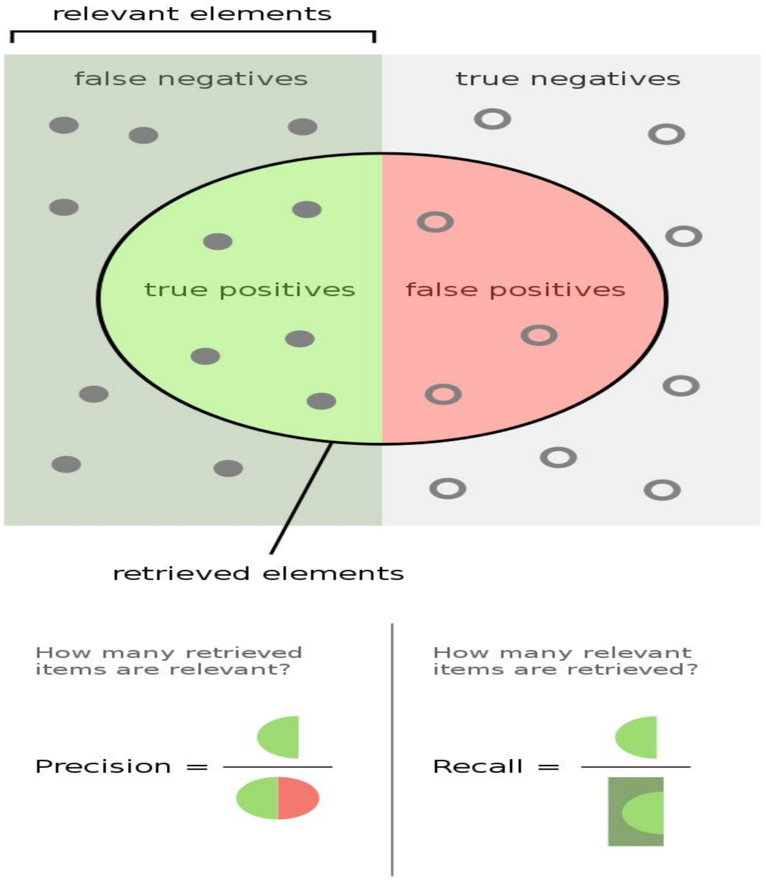
```python
def process_bert_similarity():
    # This will download and load the pretrained model offered by UKPLab.
    model = SentenceTransformer('bert-base-nli-mean-tokens')

    # Although it is not explicitly stated in the official document of sentence transformer, the original B
    sentences = sent_tokenize(base_document)
    base_embeddings_sentences = model.encode(sentences)
    base_embeddings = np.mean(np.array(base_embeddings_sentences), axis=0)

    vectors = []
    for i, document in enumerate(documents):

        sentences = sent_tokenize(document)
        embeddings_sentences = model.encode(sentences)
        embeddings = np.mean(np.array(embeddings_sentences), axis=0)

        vectors.append(embeddings)

        print("making vector at index:", i)

    scores = cosine_similarity([base_embeddings], vectors).flatten()
    output = []
    for i in range(len(scores)):
        output.append((scores[i], documents[i], doc_ids[i][0],doc_ids[i][1] ))
    output.sort(reverse=True)

    d= {}
    for i, j,l,k in output:

        d[str(i)] = [j,l,k]
    json_object = json.dumps(d, indent=4)

# Writing to sample.json
    with open("sample.json", "w") as outfile:
        outfile.write(json_object)
    highest_score = 0
    highest_score_index = 0
    for i, score in enumerate(scores):
        if highest_score < score:
            highest_score = score
            highest_score_index = i
```

# Sample Data



```
[
    {
        "id": "0704.0001",
        "submitter": "Pavel Nadolsky",
        "authors": "C. Bal\\'azs, E. L. Berger, P. M. Nadolsky, C.-P. Yuan",
        "title": "Calculation of prompt diphoton production cross sections at Tevatron and\n  LHC energies",
        "comments": "37 pages, 15 figures; published version",
        "journal-ref": "Phys.Rev.D76:013009,2007",
        "doi": "10.1103/PhysRevD.76.013009",
        "report-no": "ANL-HEP-PR-07-12",
        "categories": "hep-ph",
        "license": null,
        "abstract": "  A fully differential calculation in perturbative quantum chromodynamics is\npresented for the production of massive photon pairs at
        hadron colliders. All\nnext-to-leading order perturbative contributions from quark-antiquark,\ngluon-(anti)quark, and gluon-gluon subprocesses are
        included, as well as\nall-orders resummation of initial-state gluon radiation valid at\nnext-to-next-to-leading logarithmic accuracy. The region of
        phase space is\nspecified in which the calculation is most reliable. Good agreement is\ndemonstrated with data from the Fermilab Tevatron, and
        predictions are made for\nmore detailed tests with CDF and DO data. Predictions are shown for\ndistributions of diphoton pairs produced at the energy
        of the Large Hadron\nCollider (LHC). Distributions of the diphoton pairs from the decay of a Higgs\nboson are contrasted with those produced from QCD
        processes at the LHC, showing\nthat enhanced sensitivity to the signal can be obtained with judicious\nselection of events.\n",
        "versions": [
            {…
            },
            {…
            }
        ],
        "update_date": "2008-11-26",
        "authors_parsed": [
            […
            ],
            […
            ],
            […
            ],
            […
            ]
        ]
    },
    {
```

# Precision and Recall



## Results

| Model | Precision | Recall |
|---|---|---|
| Jaccard | 0.14333333333333334 | 0.5443037974683544 |
| TF-IDF | 0.16 | 0.6075949367088608 |
| Bert | 0.18 | 0.6835443037974683 |
| USE | 0.19666666666666666 | 0.7468354430379747 |
| SciBert | 0.20333333333333334 | 0.7686949802934344 |

All code available on - https://github.com/adit-negi/GameOfPapers

## Semantic Corpus API

**Searching For Papers By Keyword**

The /paper/search endpoint will perform a keyword search for papers using their custom-trained ranker.

http://api.semanticscholar.org/graph/v1/paper/search?query=literature+graph

This will return the top 10 matching papers, with their titles. We can request more results, and additional information with:

http://api.semanticscholar.org/graph/v1/paper/search?query=literature+graph&offset=10&limit=50&fields=title,authors

Which will return the following 50 matching papers, with both titles and authors.

## Semantic Corpus API

S2AG metadata:

- Paper attributes (title, authors, etc.)
- Author attributes (name, paper count, h-index, etc.)
- Citation links (citing/cited paper ids, context, intent)

Paper Content:

- Abstracts
- S2ORC - full text of open-access PDFs

Models:

- TLDRs - Short summaries of papers (from SciTLDR)
- Paper embeddings (from Specter)

Author lookup API:

- Detailed info about papers by the author

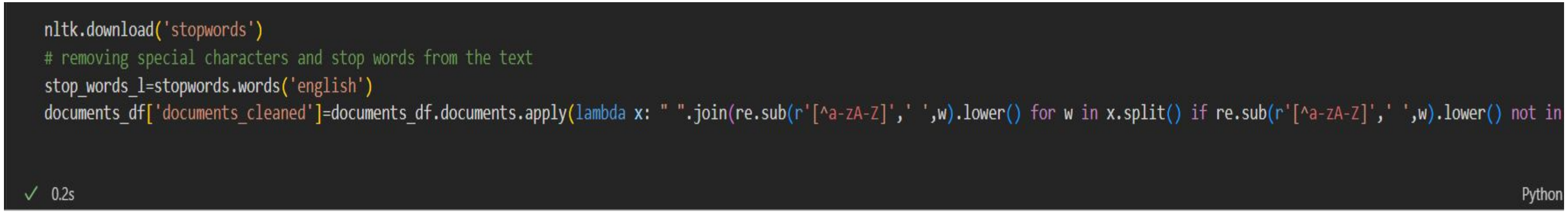
## Next Steps

1. Finetuning bert/Scibert
2. Proposing a new loss function using hyperparameters from the semantic scholar API
3. Get a more robust evaluation framework in place
   a. SciDocs?

## My work this week

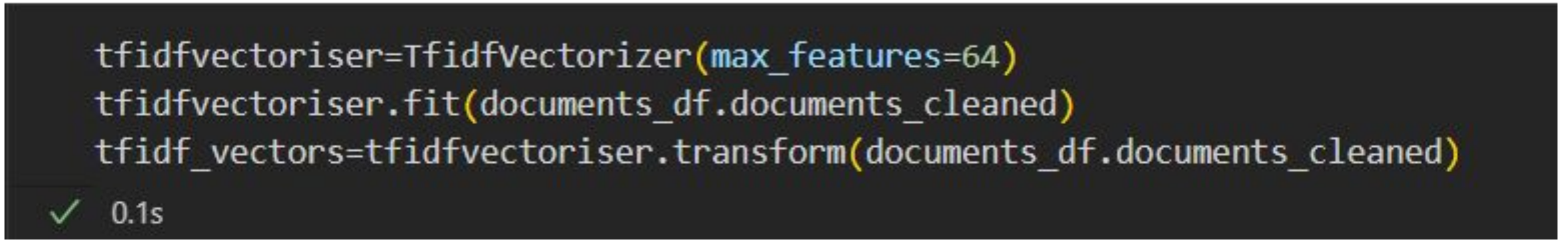
- Determine the similarity between the abstracts of various papers.
- Comparing how various models perform.
- Models used: Doc2vec model, BERT Model
- Similarity Index: Cosine Similarity

## Preprocessing:



```
nltk.download('stopwords')
# removing special characters and stop words from the text
stop_words_l=stopwords.words('english')
documents_df['documents_cleaned']=documents_df.documents.apply(lambda x: " ".join(re.sub(r'[^a-zA-Z]',' ',w).lower() for w in x.split() if re.sub(r'[^a-zA-Z]',' ',w).lower() not in
```

✓ 0.2s Python



```
tfidfvectoriser=TfidfVectorizer(max_features=64)
tfidfvectoriser.fit(documents_df.documents_cleaned)
tfidf_vectors=tfidfvectoriser.transform(documents_df.documents_cleaned)
```

✓ 0.1s

## Results

- I played around with a lot of data, first I gave the two models random sentences. In all of the cases BERT performed slightly better.
- Next I gave the two models the abstract of two papers that had similar applications i.e. Delay prediction in Public Transport Systems BERT model performed better with Cosine Similarity 0.67 compared to 0.55 using Doc2vec model.
- Finally, I played with some papers that were completely different and the model that performed the best was 0.45 on Doc2vec compared to 0.35 on BERT.

## Conclusion

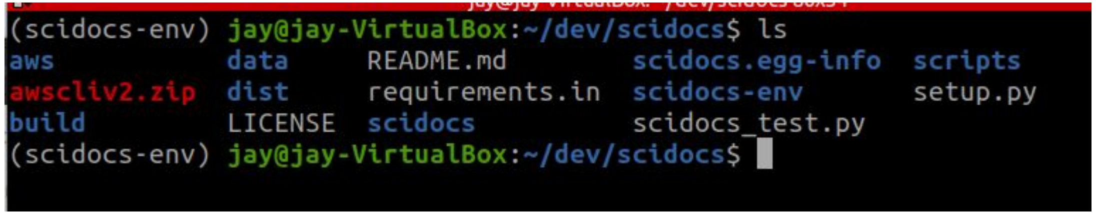
- We conclude that BERT model has shown promising results.
- BERT model in finding the similarity index of our entire corpus of data and compare with that of the input paper from the user and this can be a major feature for our final algorithm that we plan on using.

## Evaluation Suite

- Allenai - **SciDocs.**
- Reason to checkout ? Applications on scientific documents, such as classification and recommendation.
- Used for - Evaluation of Embedding Models
- Why Embedding Models ?
  - BERT - powerful textual representations, but focus more towards token, sentence based representations.
  - We need document level representations, and evaluation suites for the same
  - SciDocs - evaluation benchmark consisting of seven document-level tasks ranging from citation prediction, to document classification and recommendation.

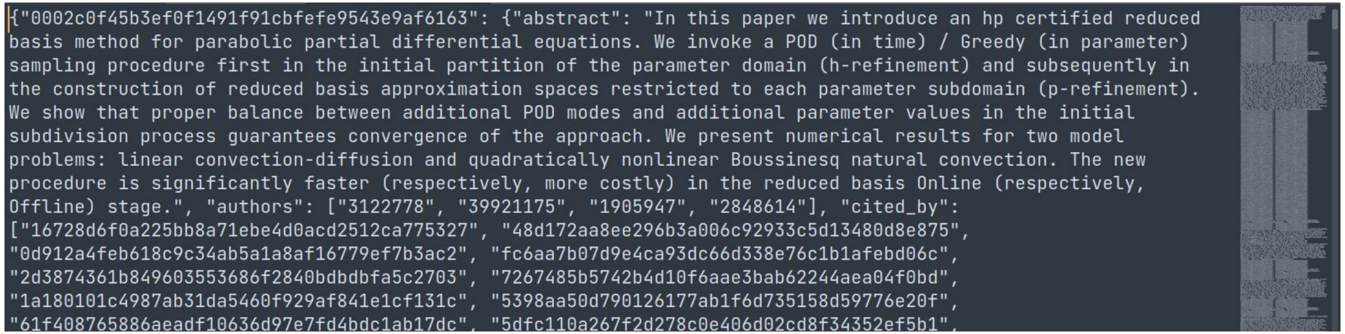
## Attempt on a test run

- Installing



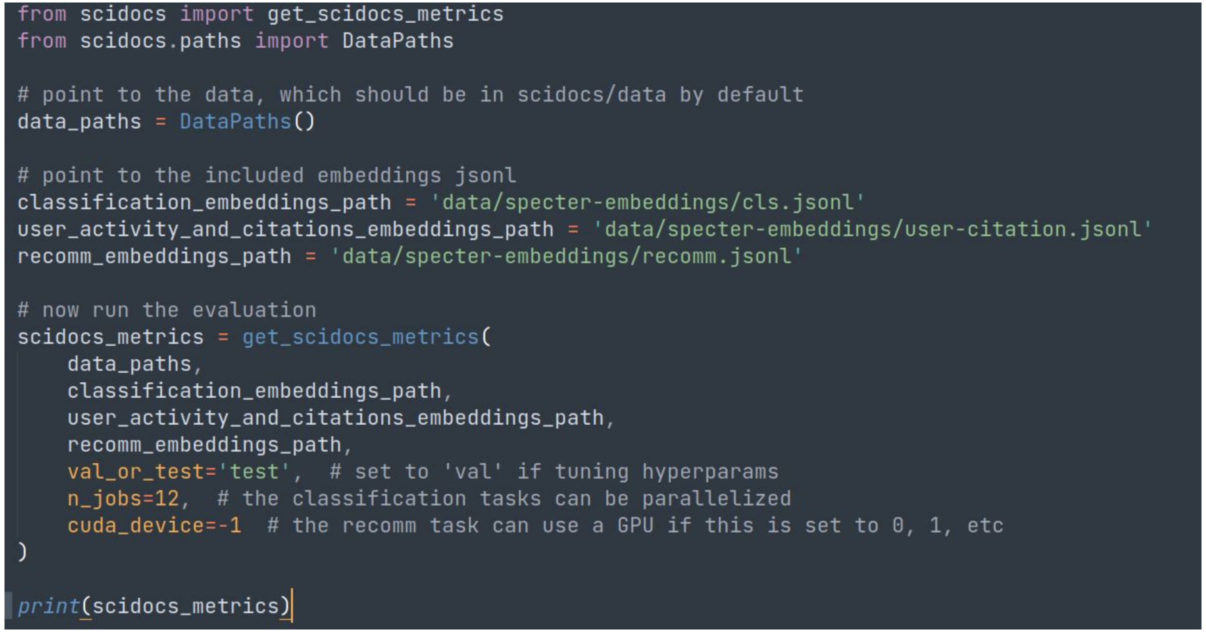
```
(scidocs-env) jay@jay-VirtualBox:~/dev/scidocs$ ls
aws             data      README.md         scidocs.egg-info  scripts
awscliv2.zip    dist      requirements.in   scidocs-env       setup.py
build           LICENSE   scidocs           scidocs_test.py
(scidocs-env) jay@jay-VirtualBox:~/dev/scidocs$
```

- Data - AWS S3, Freely available
  - Embedding Representation looks something like - {"paper_id": "0dfb47e206c762d2f4caeb99fd9019ade78c2c98", "embedding": [-3, -6, 0, ..., 2]}

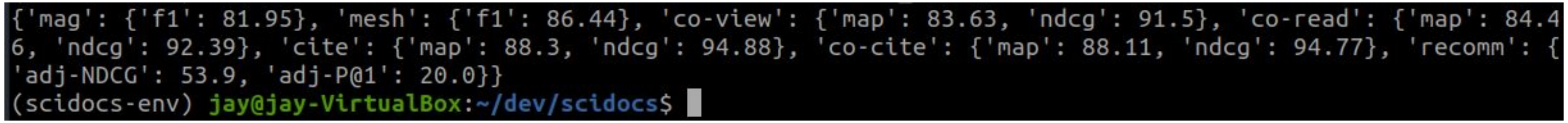


```
{"0002c0f45b3ef0f1491f91cbfefe9543e9af6163": {"abstract": "In this paper we introduce an hp certified reduced
basis method for parabolic partial differential equations. We invoke a POD (in time) / Greedy (in parameter)
sampling procedure first in the initial partition of the parameter domain (h-refinement) and subsequently in
the construction of reduced basis approximation spaces restricted to each parameter subdomain (p-refinement).
We show that proper balance between additional POD modes and additional parameter values in the initial
subdivision process guarantees convergence of the approach. We present numerical results for two model
problems: linear convection-diffusion and quadratically nonlinear Boussinesq natural convection. The new
procedure is significantly faster (respectively, more costly) in the reduced basis Online (respectively,
Offline) stage.", "authors": ["3122778", "39921175", "1905947", "2848614"], "cited_by":
["16728d6f0a225bb8a71ebe4d0acd2512ca775327", "48d172aa8ee296b3a006c92933c5d13480d8e875",
"0d912a4feb618c9c34ab5a1a8af16779ef7b3ac2", "fc6aa7b07d9e4ca93dc66d338e76c1b1afebd06c",
"2d3874361b849603553686f2840bdbdbfa5c2703", "7267485b5742b4d10f6aae3bab62244aea04f0bd",
"1a180101c4987ab31da5460f929af841e1cf131c", "5398aa50d790126177ab1f6d735158d59776e20f",
"61f408765886aeadf10636d97e7fd4bdc1ab17dc", "5dfc110a267f2d278c0e406d02cd8f34352ef5b1",
```

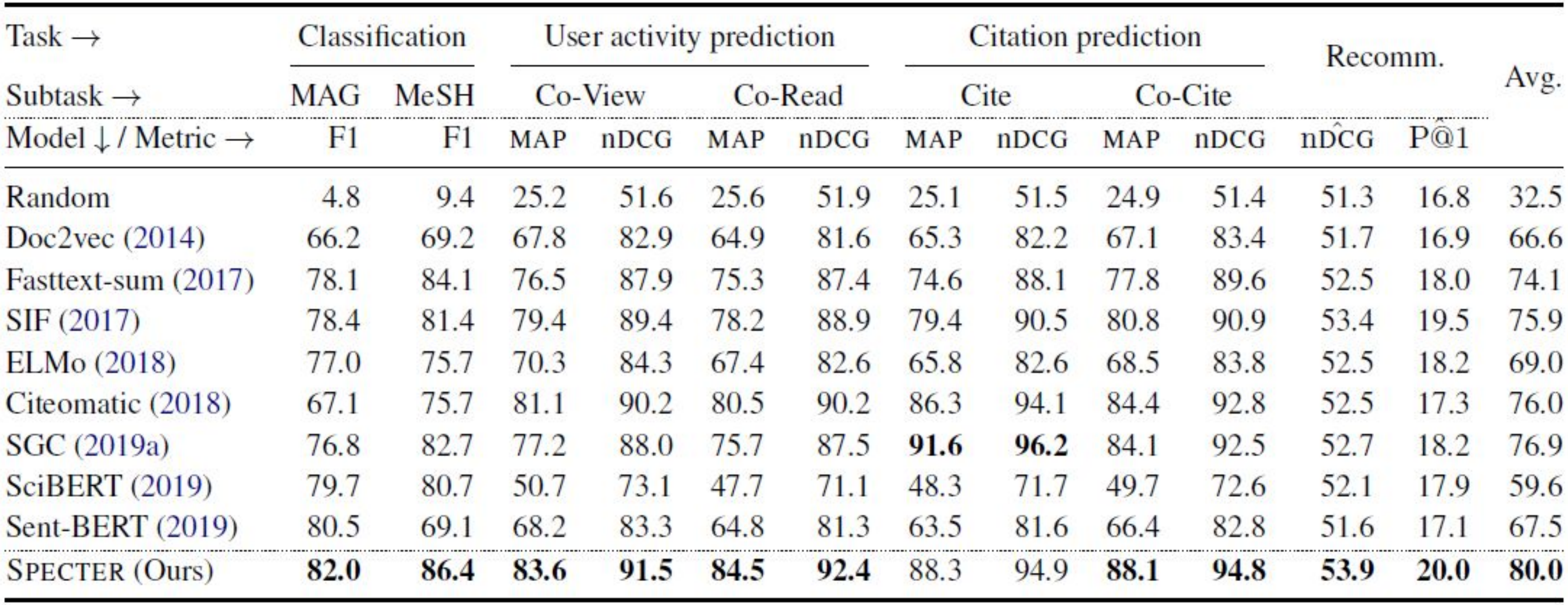
## Test script



```python
from scidocs import get_scidocs_metrics
from scidocs.paths import DataPaths

# point to the data, which should be in scidocs/data by default
data_paths = DataPaths()

# point to the included embeddings jsonl
classification_embeddings_path = 'data/specter-embeddings/cls.jsonl'
user_activity_and_citations_embeddings_path = 'data/specter-embeddings/user-citation.jsonl'
recomm_embeddings_path = 'data/specter-embeddings/recomm.jsonl'

# now run the evaluation
scidocs_metrics = get_scidocs_metrics(
    data_paths,
    classification_embeddings_path,
    user_activity_and_citations_embeddings_path,
    recomm_embeddings_path,
    val_or_test='test',  # set to 'val' if tuning hyperparams
    n_jobs=12,  # the classification tasks can be parallelized
    cuda_device=-1  # the recomm task can use a GPU if this is set to 0, 1, etc
)

print(scidocs_metrics)
```



```
{'mag': {'f1': 81.95}, 'mesh': {'f1': 86.44}, 'co-view': {'map': 83.63, 'ndcg': 91.5}, 'co-read': {'map': 84.4
6, 'ndcg': 92.39}, 'cite': {'map': 88.3, 'ndcg': 94.88}, 'co-cite': {'map': 88.11, 'ndcg': 94.77}, 'recomm': {
'adj-NDCG': 53.9, 'adj-P@1': 20.0}}
(scidocs-env) jay@jay-VirtualBox:~/dev/scidocs$ 
```



| Task → | Classification | | User activity prediction | | | | Citation prediction | | | | Recomm. | | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Subtask → | MAG | MeSH | Co-View | | Co-Read | | Cite | | Co-Cite | | | | |
| Model ↓ / Metric → | F1 | F1 | MAP | nDCG | MAP | nDCG | MAP | nDCG | MAP | nDCG | $\widehat{\text{nDCG}}$ | $\hat{\text{P}}$@1 | |
| Random | 4.8 | 9.4 | 25.2 | 51.6 | 25.6 | 51.9 | 25.1 | 51.5 | 24.9 | 51.4 | 51.3 | 16.8 | 32.5 |
| Doc2vec (2014) | 66.2 | 69.2 | 67.8 | 82.9 | 64.9 | 81.6 | 65.3 | 82.2 | 67.1 | 83.4 | 51.7 | 16.9 | 66.6 |
| Fasttext-sum (2017) | 78.1 | 84.1 | 76.5 | 87.9 | 75.3 | 87.4 | 74.6 | 88.1 | 77.8 | 89.6 | 52.5 | 18.0 | 74.1 |
| SIF (2017) | 78.4 | 81.4 | 79.4 | 89.4 | 78.2 | 88.9 | 79.4 | 90.5 | 80.8 | 90.9 | 53.4 | 19.5 | 75.9 |
| ELMo (2018) | 77.0 | 75.7 | 70.3 | 84.3 | 67.4 | 82.6 | 65.8 | 82.6 | 68.5 | 83.8 | 52.5 | 18.2 | 69.0 |
| Citeomatic (2018) | 67.1 | 75.7 | 81.1 | 90.2 | 80.5 | 90.2 | 86.3 | 94.1 | 84.4 | 92.8 | 52.5 | 17.3 | 76.0 |
| SGC (2019a) | 76.8 | 82.7 | 77.2 | 88.0 | 75.7 | 87.5 | **91.6** | **96.2** | 84.1 | 92.5 | 52.7 | 18.2 | 76.9 |
| SciBERT (2019) | 79.7 | 80.7 | 50.7 | 73.1 | 47.7 | 71.1 | 48.3 | 71.7 | 49.7 | 72.6 | 52.1 | 17.9 | 59.6 |
| Sent-BERT (2019) | 80.5 | 69.1 | 68.2 | 83.3 | 64.8 | 81.3 | 63.5 | 81.6 | 66.4 | 82.8 | 51.6 | 17.1 | 67.5 |
| SPECTER (Ours) | **82.0** | **86.4** | **83.6** | **91.5** | **84.5** | **92.4** | 88.3 | 94.9 | **88.1** | **94.8** | **53.9** | **20.0** | **80.0** |

Table 1: Results on the SCIDOCS evaluation suite consisting of 7 tasks.

## Thank you.

## Any Questions?