# Project Name: Paperank

**Needle in a Data Haystack Final Project**

Team Members:

Roy Zohar – [roy.zohar.40@gmail.com](mailto:roy.zohar.40@gmail.com) – 209896174, roy\_zohar

Yoav Tamir – [yoavtamir00@gmail.com](mailto:yoavtamir00@gmail.com) – 322291519, yoav.tamir

**Overview:**

Our Project’s goal was to develop a tool that helps researchers find relevant and influential papers. On the practical side, we wanted to take a huge amount of academic papers, and be able to search and categorize them, in order to find important data.

**Data:**

We used the ***Semantic Scholar*** database, which contains a whopping 40 million academic papers. Each academic paper contains lots of raw data regarding the paper. The total size of the database is 87GBs. This database is open-source, and we downloaded it off the semantic scholar website: <https://www.semanticscholar.org/>

**Our solution:**

The main stages of our project were:

1. We processed the entire database, and organized it in databases that are convenient to use.
2. We ran a PageRank algorithm on the papers, where instead of edges we used citations.
3. We clustered the most important papers according to keywords from the abstract of each paper.
4. We implemented a heuristic search through the database, similar to Google Search, that takes into account both importance and matchness.
5. Content (Researcher) Based recommender system.

**1. Processing the Database:**

The raw data we used was organized in 40 huge JSON files. Obviously, our data cannot be kept all at once in the RAM. Therefore, we first off built 2 SQL databases, that act as dictionaries, that map each paper to a unique index identifier and vice versa.

We noticed that the SQL databases weren’t running in reasonable enough time. After trying a few tools and running some tests, we found that numpy’s memmap tool was the most efficient option. We created a few memmap arrays, that represent the edge matrix of citations. Since (40\*10^6)^2 was more memory than we could allow ourselves, we couldn’t save a 2-dimensional array in memory as is. Our solution here was to use the sparsity of the matrix, and store the edges in a sort of array of linked lists.

**2. Running Page Rank**

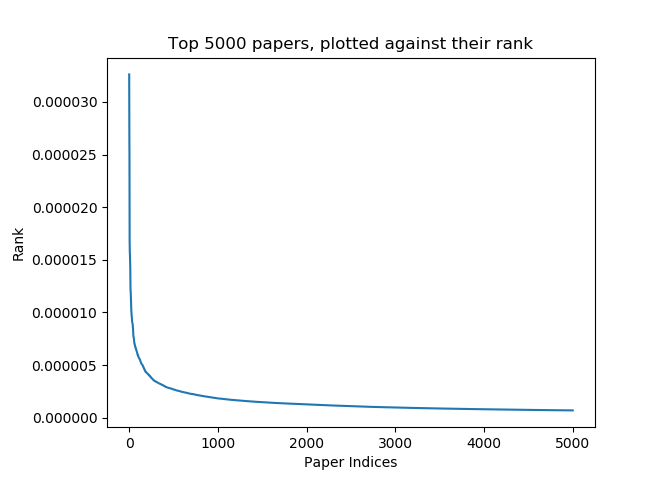
Next, we ran the PageRank algorithm on all the academic papers. The idea behind using PageRank, is that a paper is more important in our opinion if has been heavily cited, especially by other important papers. The algorithm is identical to the one we saw in class.

As a sanity check, we looked at the most important paper to see if it seems like an important article. The algorithm outputted:



Which makes perfect sense, since it has been tagged on Semantic Scholar as “highly influential”, and has been cited a whopping 41475 times.

We also plotted the top 5000 papers’ ranks, and got the following figure:



What can we learn from this visualization? There is a small group of papers that are extremely significant by a few orders of magnitude than most of the papers. This fits our understanding of the academic world – there are tons of papers but only a select few are truly groundbreaking.

**4. Academic Paper Search**

We wanted to implement