


# Academus

## An all-in-one compact search engine solution

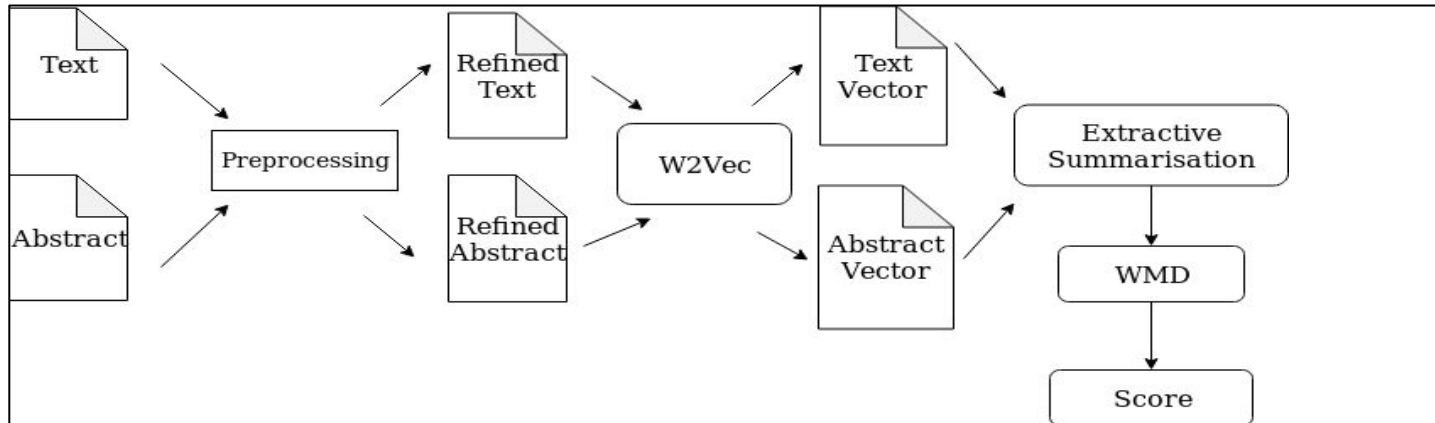
By Arjun, Chaitanya, Pranav & Sai  
(VictoryShadows)



# The Workings

## Notable Black Boxes

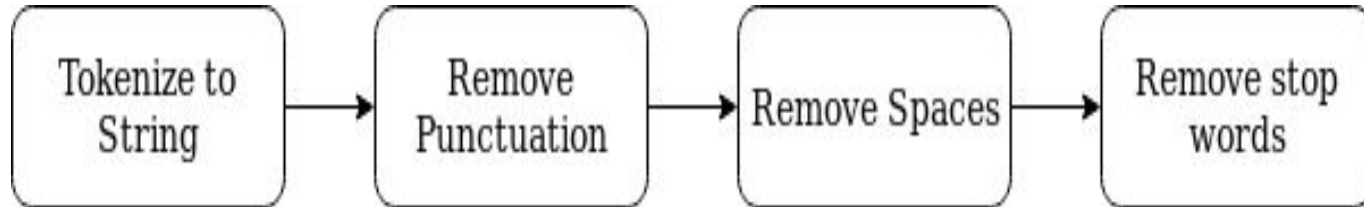
- Preprocessing
- Word2Vec Conversion
- Extraction Summarizing
- Word Mover Distance (WMD) Calculation



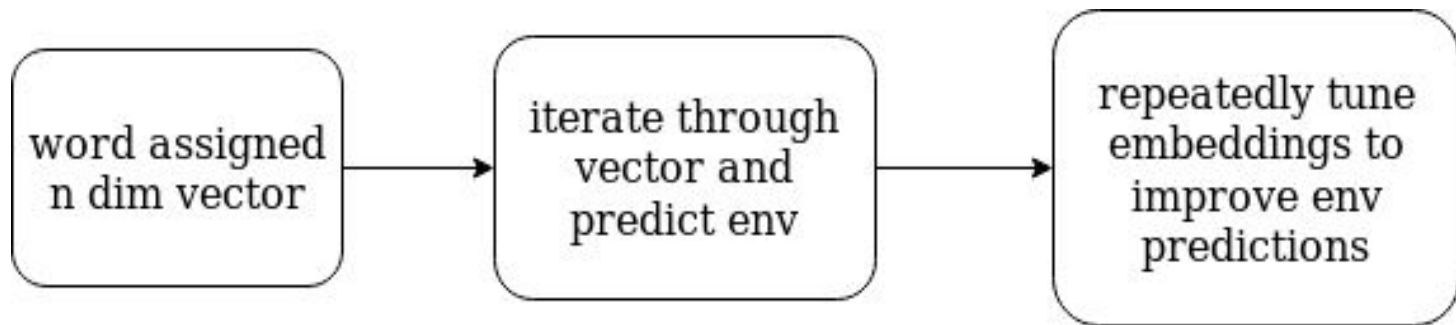
# Preprocessing

Conducts rudimentary  
preprocessing for NLP libraries

Uses regex to tokenize the text  
into sentences



# Word2Vec Conversion



## Standard Method

### String Matching

- Manually manoeuvres each document and checks for matches

## Example Issue

The strings "fruit " & "orange" are of similar topics.

String matching would not return a match for these two strings.

## Our Approach

### Word2Vector Model

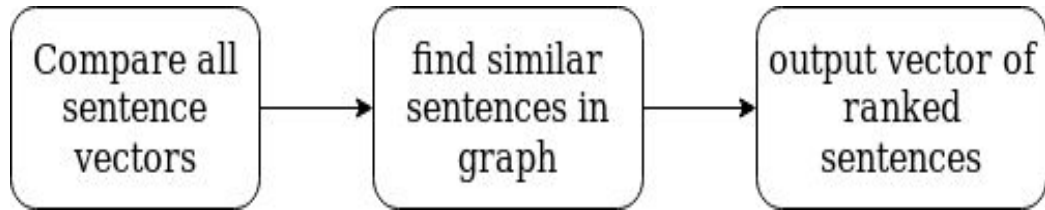
- Uses word environment to train the embeddings
- Similar words have a similar environment, leads to similar vector creation

## Our Resolution

The strings "fruit" & "orange" are normalized into similar vectors (due to similar environmental embeddings).

Our solution would return a match for these two strings.

# Extraction Summarization



Every sentence has its own vector

We compare & group similar sentences. Then we output a descending rank list of sentences based on similarity (analogous to a page-rank model).

## Optimization

Length of the abstract and text are marginally different in sizes.

We create a summary that accurately depicts the text to reduce the size & improve speeds

# Word Mover Distance Calculation

Uses the flow algorithm to return an integer that numerically represents the similarity between the two vectors.

Works on the principle that similar sentences comprise of common word structures & contrasting sentences have unmatching word compositions.



# Search Engine For Queries

## Complications

- Requires fast query times, but are difficult to produce due to large volumes of data.
- Most documents will be dissimilar to query.
  - Inefficient to compare for matches with these documents.
  - Requires more efficient approach.
- Finding the nearest neighbour in our DGM is an NP-hard problem.





# Search Engine For Queries

## Our Solution

### Document Graph Model (DGM)

- A weighted undirectional graph is generated.
  - Each node is an individual document
  - Each edge weight is the similarity between its corresponding documents (nodes).
  - Edges are created between documents only if their WMD is above a threshold frequency

### Approximate Nearest Neighbour (ANN)

- A binary tree is created by repeatedly diving the  $n$  dimensional vector space into  $n-1$  hyperplanes.
- BFS is run on the binary tree, and the leaf nodes are taken as more accurate approximate neighbours.

# Future Enhancements

## Web App Deployment

- UI Layer (Similar to any search engine UI)
- Data Model (for the local data storage of documents)
- Rating System for search results to improve the model (potentially switching to a supervised learning model)