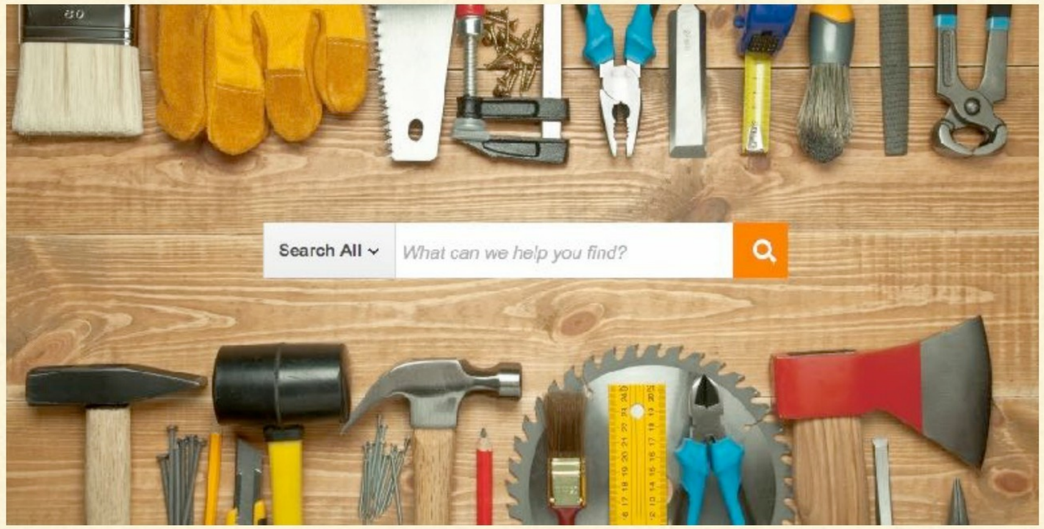
1092 WSM Project 1:
Search and Rank via **Vector Space Models**

The Task

- 7,034 English news



1. Calculate the relevance of each news by the given query.
2. Return the top relevant news IDs.

Processing Steps (1)

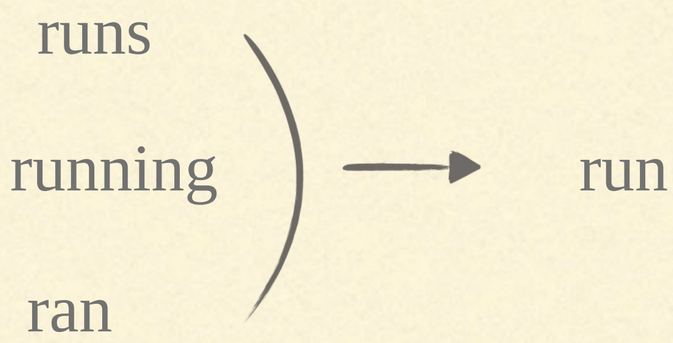
(1) Stemming & Removing Stop Words & Indexing
(2) Transfer Query into a Vector
(3) Transfer Documents into Vectors
(4) Calculate the similarity between the Query Vector and the Document Vectors
(5) Rank the Documents according to the similarity scores

```
News103561.txt  News107021.txt
News103563.txt  News107023.txt
News103566.txt  News107041.txt
News103570.txt  News107047.txt
News103572.txt  News107060.txt
News103573.txt  News107062.txt
News103576.txt  News107068.txt
News103577.txt  News107070.txt
News103581.txt  News107073.txt
News103582.txt  News107074.txt
News103594.txt  News107081.txt
```

```
Analysis: Wall Street brushes off political turmoil, looks to
 economic rebound
NEW YORK (Reuters) - The U.S. stock market is mostly unfazed
by the political turmoil in Washington and fears of violence
ahead of President-elect Joe Biden’s inauguration, with inves
tors squarely focused on the probability of another sizeable
stimulus package to boost economic growth and the rollout of
coronavirus vaccines.
```

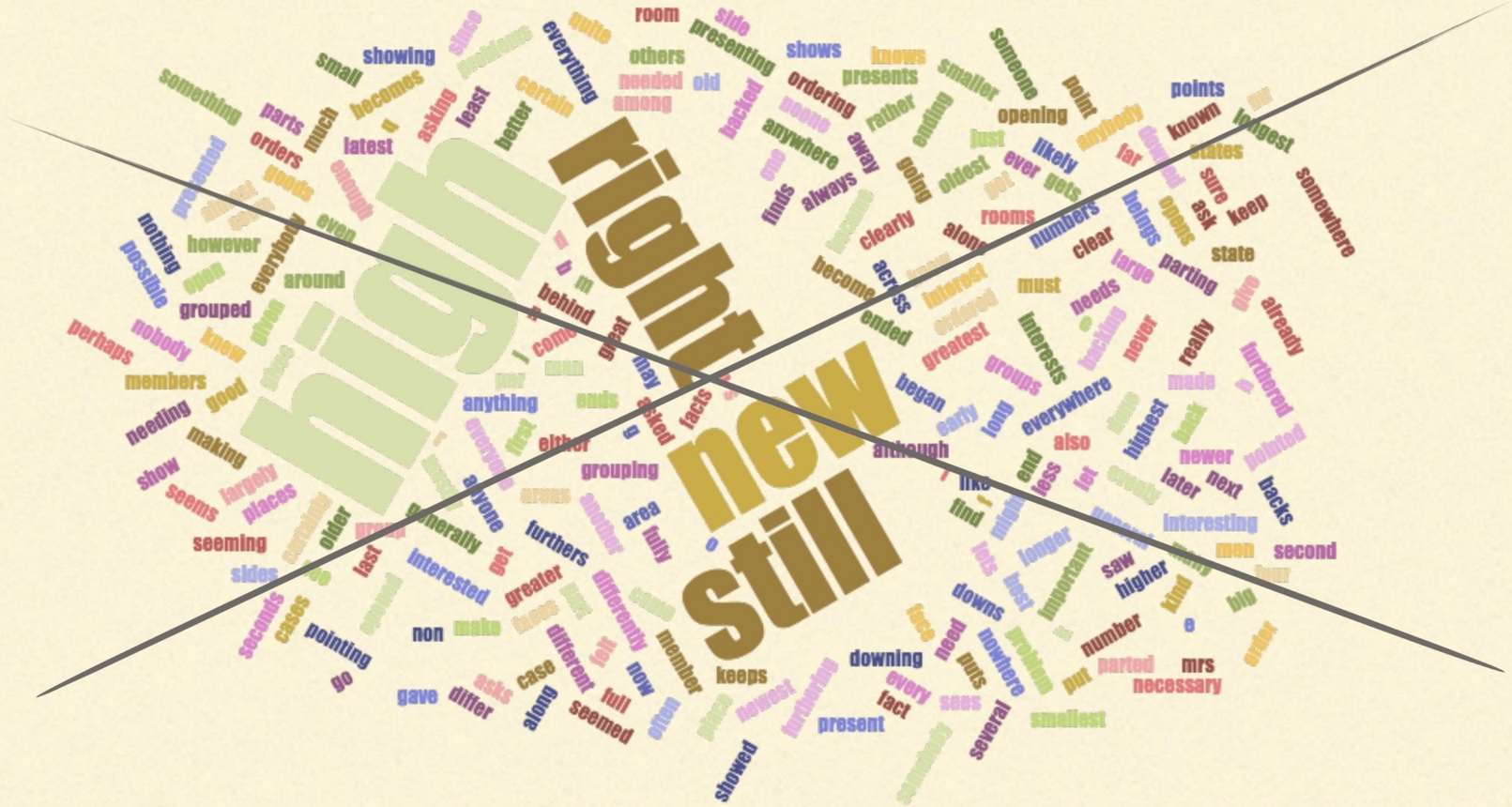
Processing Steps (2)

(1). **Stemming** & Removing Stop Words & Indexing
(2). Transfer Query into a Vector
(3) Transfer Documents into Vectors
(4)Calculate the similarity between the Query Vector and the Document Vectors
(5). Rank the Documents according to the similarity scores



Processing Steps (3)

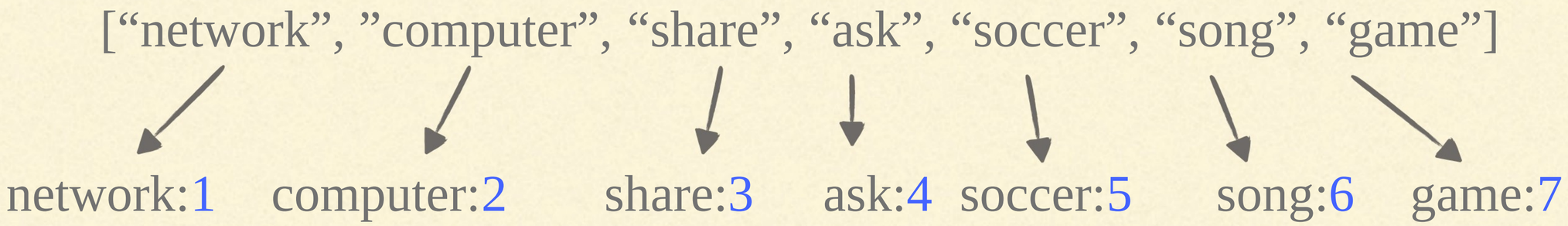
(1). Stemming & **Removing Stop Words** & Indexing



You can find stop words list from any place

Processing Steps (4)

(1) Stemming & Removing Stop Words & **Indexing**
(2) Transfer Query into a Vector
(3) Transfer Documents into Vectors
(4)Calculate the similarity between the Query Vector and the Document Vectors
(5). Rank the Documents according to the similarity scores



Processing Steps (5)

(1). Stemming & Removing Stop Words & Indexing

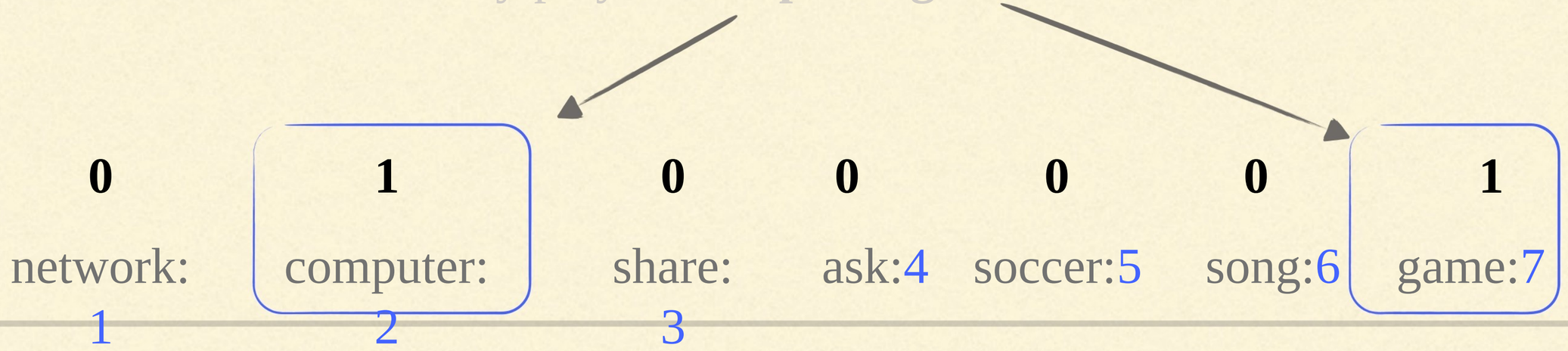
(2). Transfer Query into a Vector

(3) Transfer Documents into Vectors

(4)Calculate the similarity between the Query Vector and the Document Vectors

(5). Rank the Documents according to the similarity scores

Jimmy plays a **computer game** …



Processing Steps (5) cont'd

(1). Stemming & Removing Stop Words & Indexing

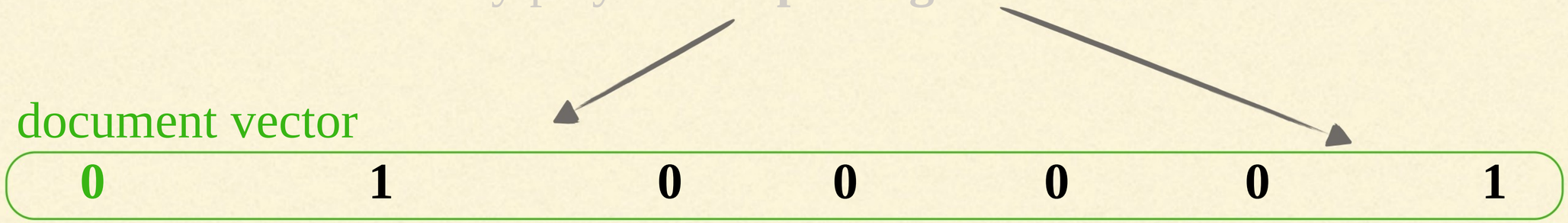
(2). Transfer Query into a Vector

(3) Transfer Documents into Vectors

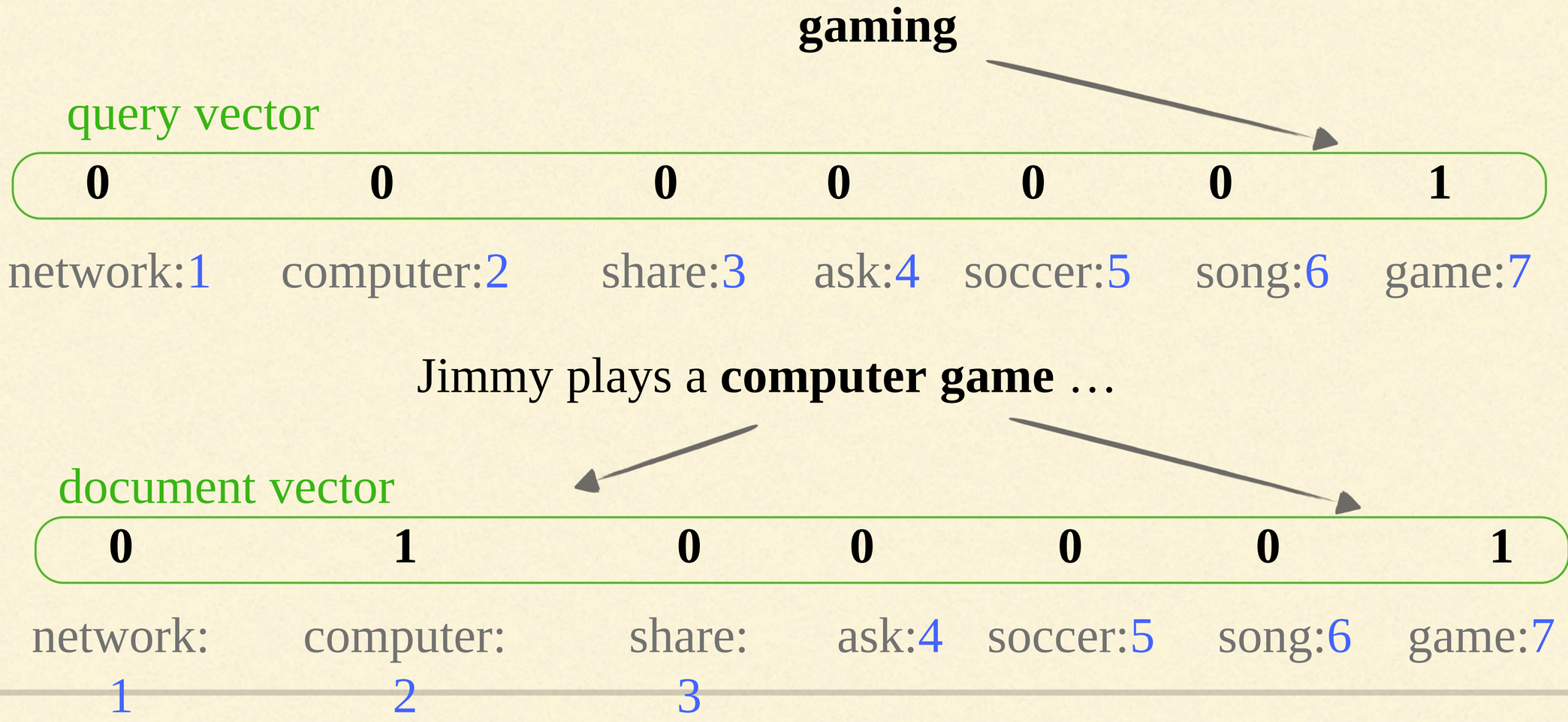
(4)Calculate the similarity between the Query Vector and the Document Vectors

(5). Rank the Documents according to the similarity scores

Jimmy plays a **computer game** …



Processing Steps (5) cont'd

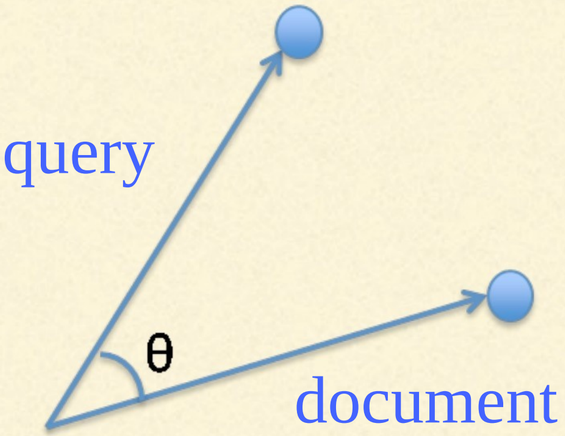


Processing Steps (6)

(1). Stemming & Removing Stop Words & Indexing
(2). Transfer Query into a Vector
(3) Transfer Documents into Vectors
(4)Calculate the similarity between the Query Vector and the Document Vectors
(5). Rank the Documents according to the similarity scores

query vector document vector

$$sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



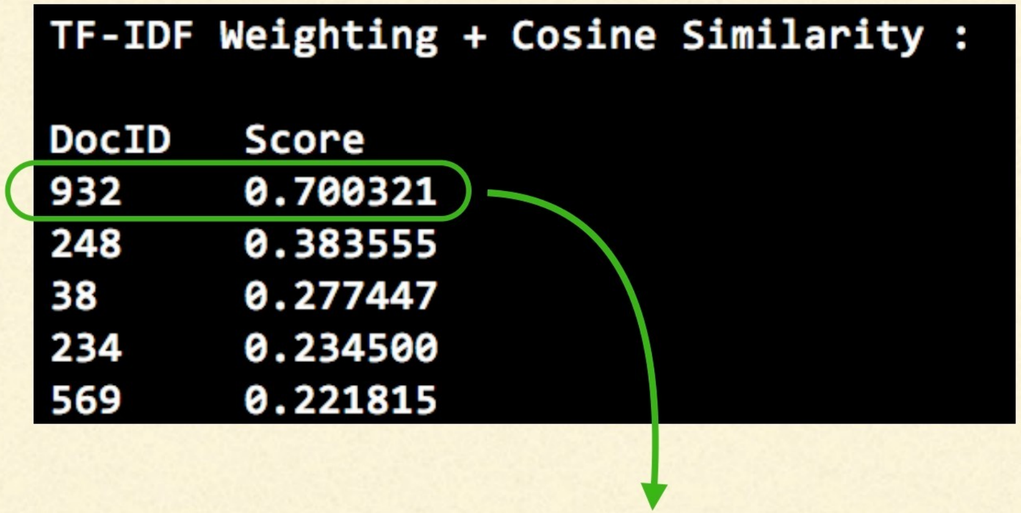
Processing Steps (7)

(1). Stemming & Removing Stop Words & Indexing
(2). Transfer Query into a Vector
(3) Transfer Documents into Vectors
(4)Calculate the similarity between the Query Vector and the Document Vectors
(5). Rank the Documents according to the similarity scores

```
TF-IDF Weighting + Cosine Similarity

NewsID          Score
----------      --------
News108813      0.386978
News104913      0.386978
News116613      0.386978
News103134      0.366336
```

Relevance Feedback



Extract Feedback Vector

Relevance Feedback (cont’d)

TF-IDF Weighting + Cosine Similarity :

DocID	Score
932	0.700321
248	0.383555
38	0.277447
234	0.234500
569	0.221815

Feedback Vector

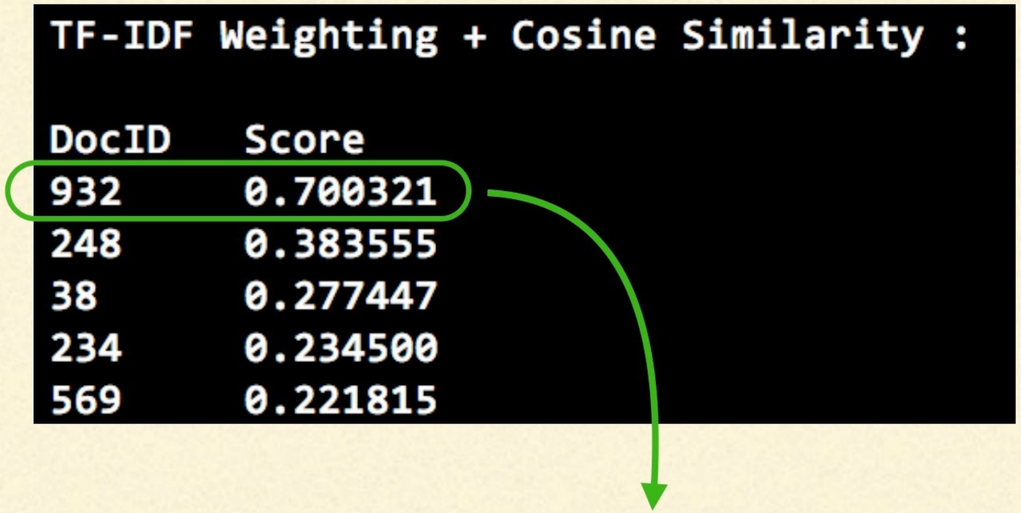
Leave only Nouns & Verbs

Feedback Queries + TF-IDF Weighting + Cosine Similarity:

DocID	Score
932	0.900000
624	0.191011
234	0.176471
336	0.165563
25	0.163121

Query Vector + Feedback Vector

Relevance Feedback (cont’d)



Feedback Vector

Leave only Nouns & Verbs

You are allowed to use any toolkit for such grammatical tagging.

keywords:
part-of-speech tagging
POS-tagging
simply tagging

Submission Format:

1. You are asked to handle a zip file (named in < 學號 >.zip) to wm5 with every related flies (including programs and data) you need.
2. Please write a ReadMe to introduce your code.
3. For people who use Python:
    1. The main execute file should be named main.py
    2. My execute command will always be “python main.py --query <query>”,
       please make sure all your program can be run like this. You can use “argparser” or other packages to achieve this.
4. For people who “do not” use Python:
    1. Write the executing way and the packages you need in ReadMe.

Any Questions?