# Related Article Suggestion with Diversity and Relevance Feedback on Turkish Wikipedia Corpus

Devrim Şahin

# Motivation and Description

Wikipedia is a good resource

Very useful if you know what you seek

But, **difficult to explore**

Well-structured pages have an 'also see' section, the rest of the articles lack such connections

# Motivation and Description

**Aim:** To provide an automated list of relevant articles for a given Wikipedia article in Turkish Wikipedia

**Why:** Because 'Vikipedi' is not very well-organised

Most of the articles lack a 'related pages' section

# Motivation and Description

We want these results to be **diverse**

We want to explore as many aspects as possible

Through **MMR**

We want the user to be able to **refine** the results

**Relevance feedback**

Through '**query expansion**'

# Methodology

User requests related articles for a query

Queries are also articles

10 **diverse** results are retrieved

Diversity through MMR

$$MMR = \arg\max_{D_i \in R \setminus S} \left[ \lambda \, Sim_1(D_i, Q) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j) \right]$$

# Methodology

User then chooses which of the results are relevant and which are irrelevant

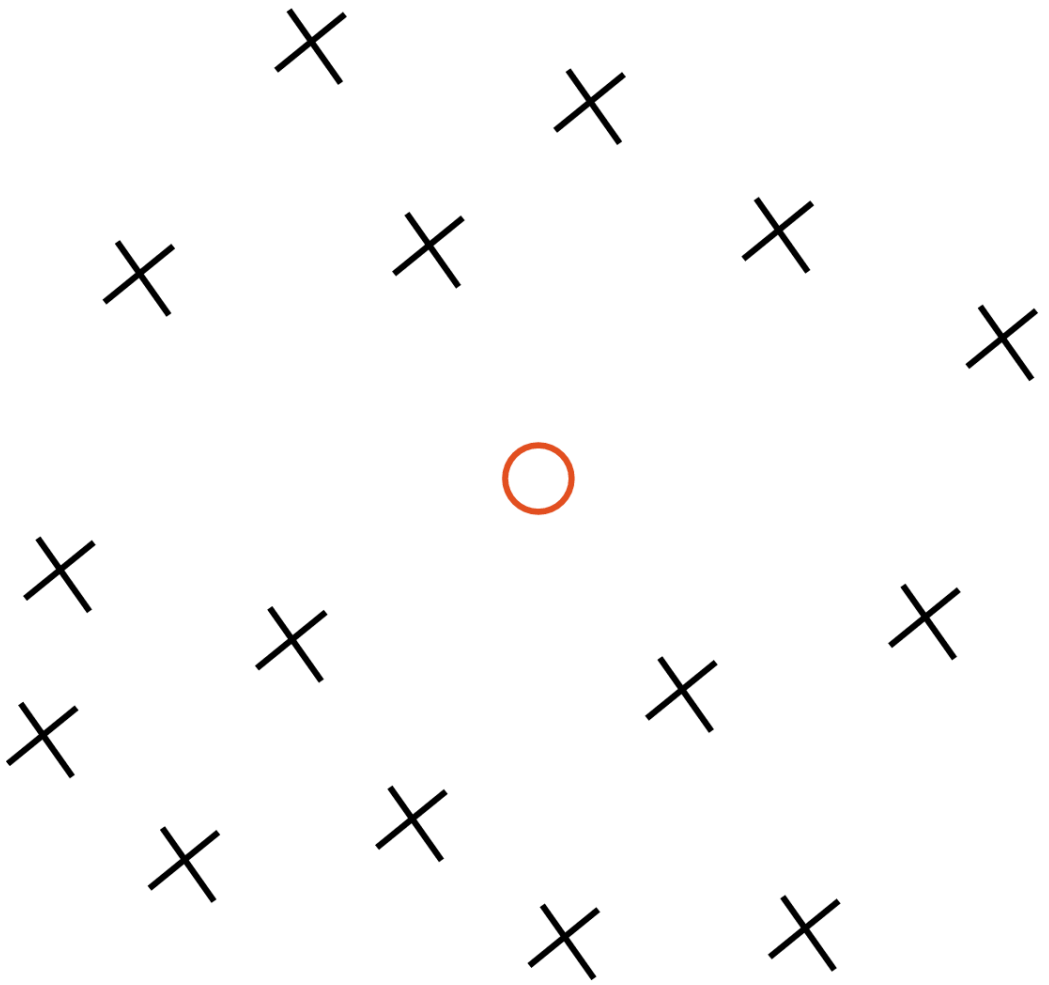
Using the relevance feedback, results are refined

This step can be repeated many times by the user

**Method:** Query expansion

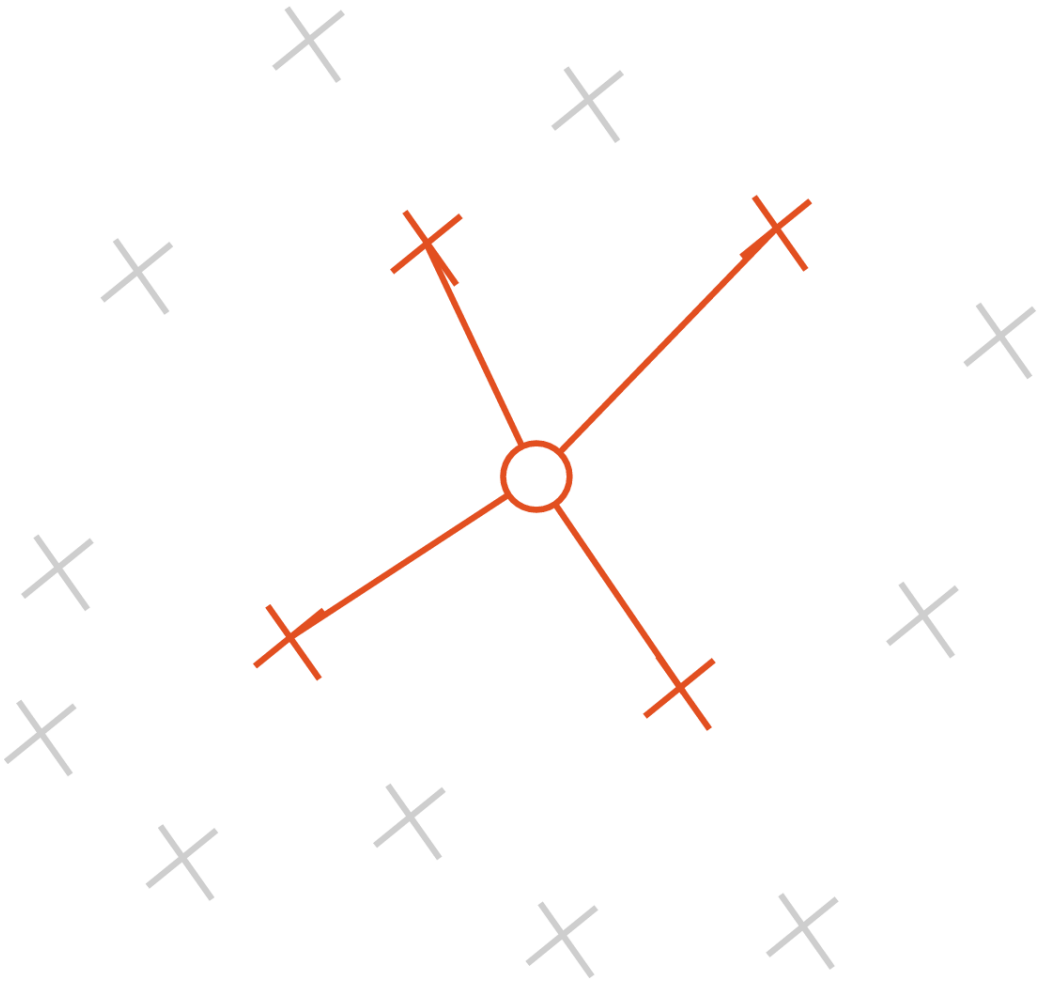
# Methodology

Relevance feedback through query expansion



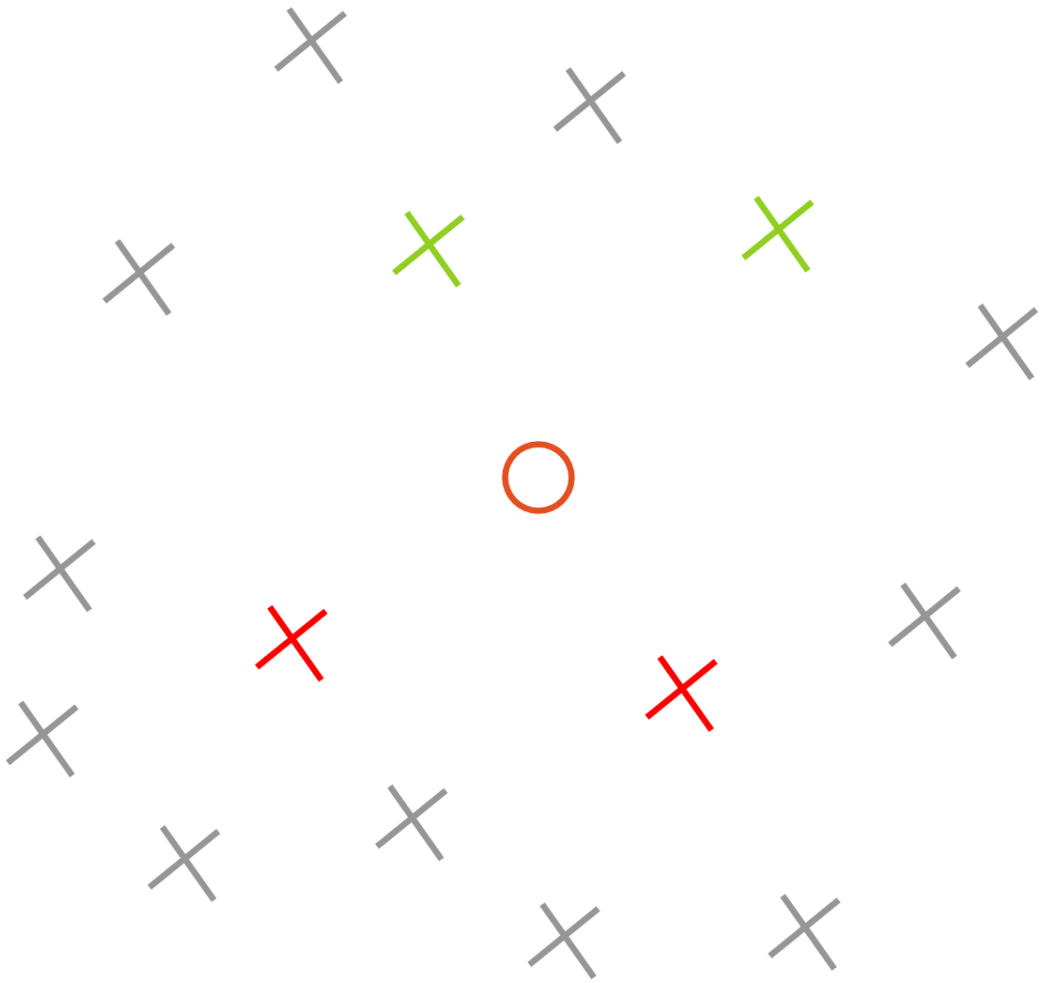
# Methodology

Relevance feedback through query expansion



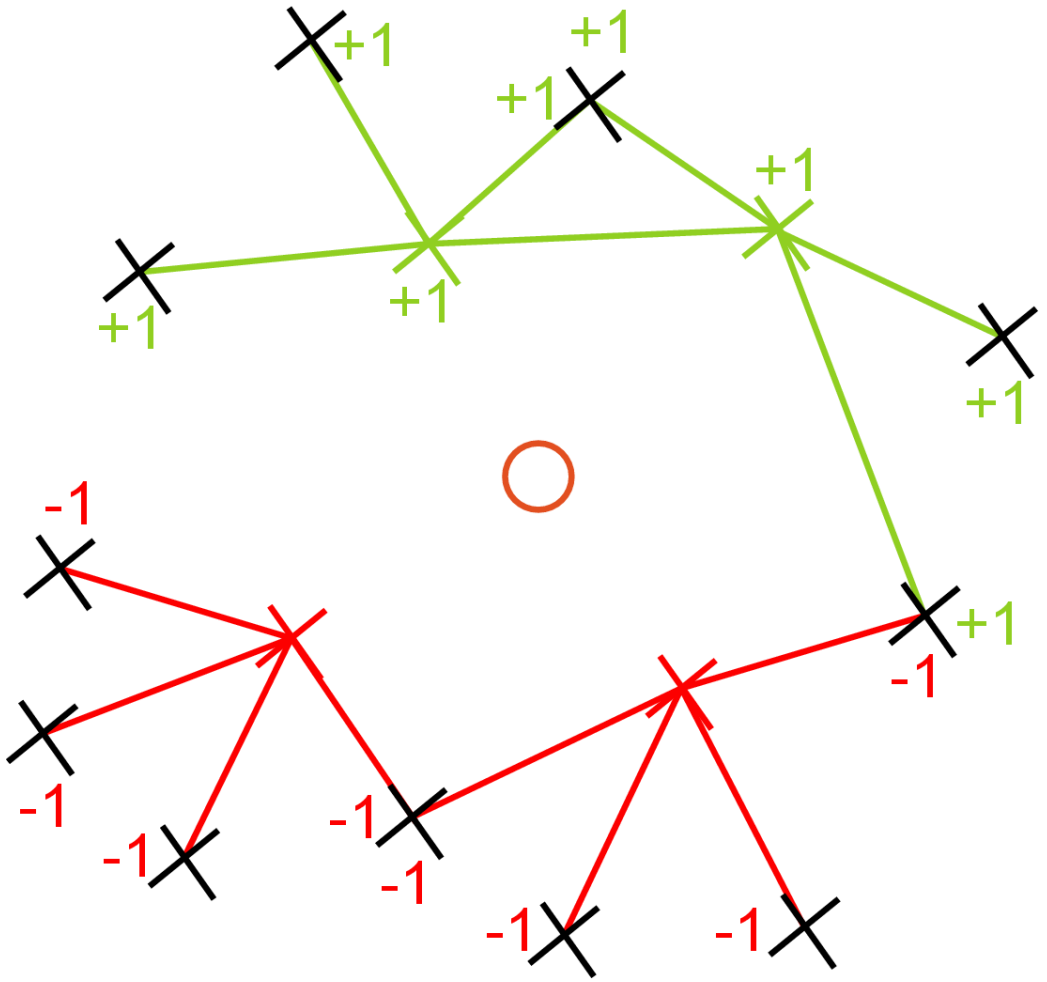
# Methodology

Relevance feedback through query expansion



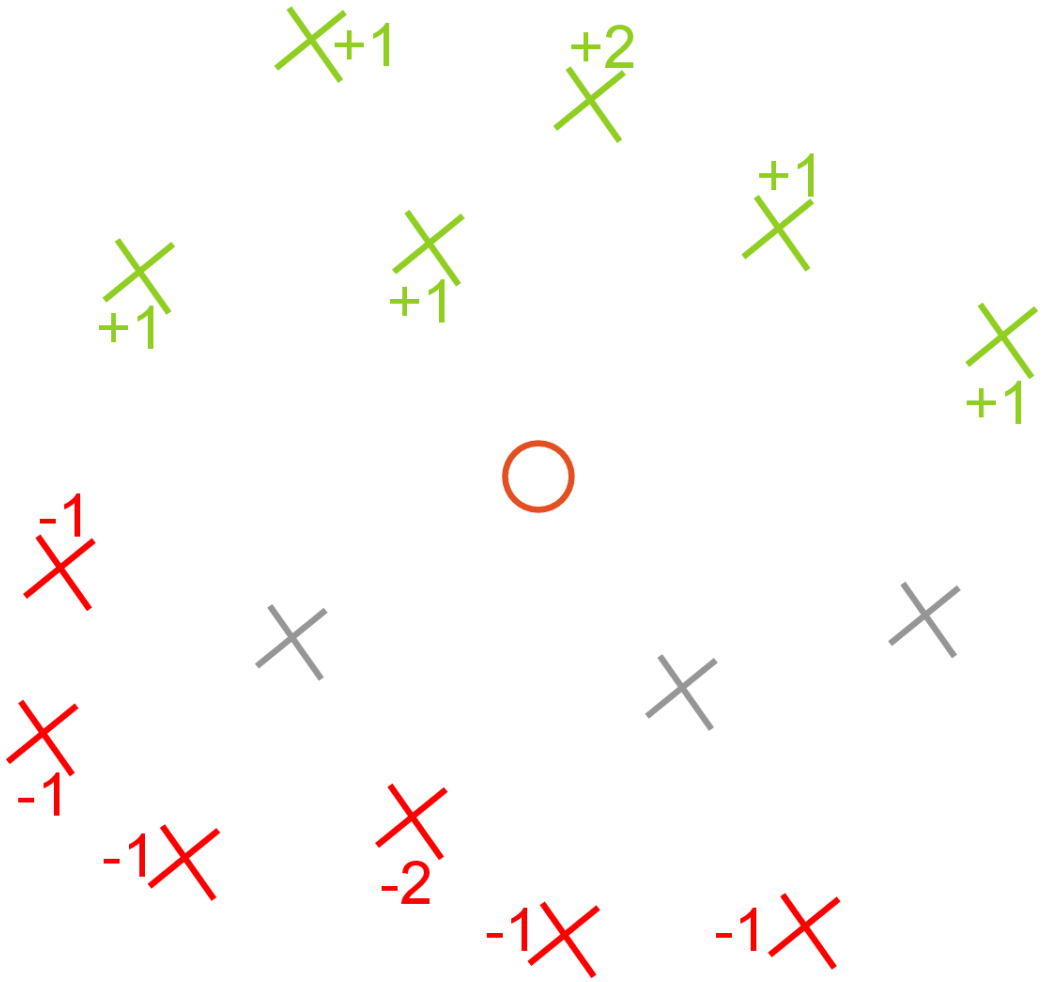
# Methodology

Relevance feedback through query expansion



# Methodology

Relevance feedback through query expansion



# Work Done

XML dump of the Turkish Wikipedia was downloaded (**289 MB**)

Plain text extraction: **WP2TXT** by Yoichiro Hasebe

Seven huge .txt files generated (**313MB** in total)

Python script to save to separate files, enumerated from 1 to **209681**. Also, titles are saved separately.

# Work Done

Obtain word stems, find a unique list of terms, extract **100** stop words

Stemming method: First-5, no numerals, Turkish characters are converted to their ASCII counterparts.

Terms shorter than 3 characters are also discarded

**248186** terms in the end

# Work Done

Stop words:

bir, ile, olara, olan, icin, bulun, yilin, taraf, sonra, yapil, ilk, vardi, basla, daha, arasi, bagli, tarih, birli, kulla, cok, buyuk, ise, sagla, yilla, oldug, gelen, ancak, gibi, gore, yer, kadar, iceri, zaman, calis, olup, yerle, uzeri, iki, alani, ilces, karsi, kurul, kendi, dunya, uzakl, ekono, okulu, koyun, yerin, hayva, yoktu, yol, ayni, bolge, ayric, genel, tarim, goste, merke, donem, ilini, kanal, koydu, sagli, iklim, elekt, yukse, yapti, ulasi, diger, bilgi, veril, olmus, koyde, baska, oldu, her, adi, sahip, suyu, ilkog, icind, etki, eski, yonet, kazan, hakki, dayal, sebek, egiti, son, telef, subes, sabit, icme, degis, ocagi, bilin, oneml, turk

# Work Done

Using the term list, index (**191 MB**) and inverted index (**204 MB**) are constructed

Finally, the diverse and non-diverse 10-nearest neighbor sets for all documents are calculated and stored in two different .npz files*

(*) psst, there is a problem here

# Work Done

Why store the diverse and non-diverse NN-sets?

**1st iteration:** “Provide 10 diverse results for $q$”

Simple lookup from the diverse NN dataset

**Following iterations:** Expand the result set using NN sets of each of the elements in the result set

Again, retrieve from the predefined non-diverse NN dataset

# Work Done

Finally, a Python-based web server is implemented using the Flask library

Uses the final k-NN lists only

Diverse lists for initial results, non-diverse lists for the expansions in the later iterations

No need for on-the-run distance calculations

# (*): Problem

The code is excruciatingly slow

Constructing the index and the inverted index takes up to a couple hours

Finding the neighors would take **days**

Parallelism and multithreading did not help

# Evaluations

For testing purposes, we worked on a subset of Wikipedia

Each 100th article was taken

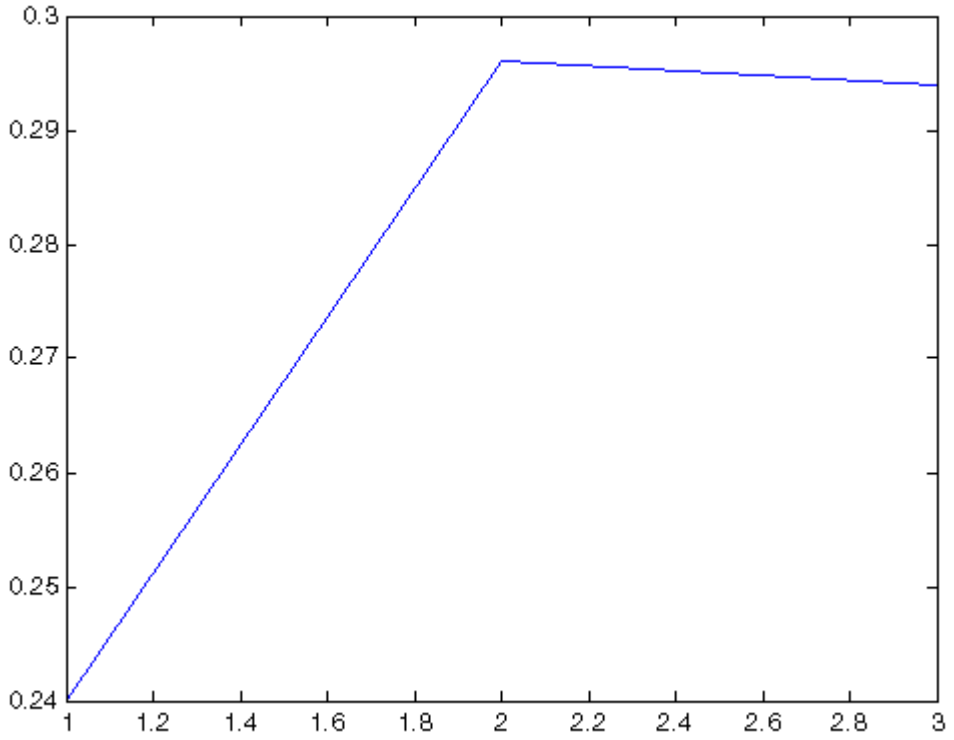
**Results are worse than expected:**

Most of the relevant documents are not included

# Evaluations

**Example:** Similarity to the query on 3 iterations

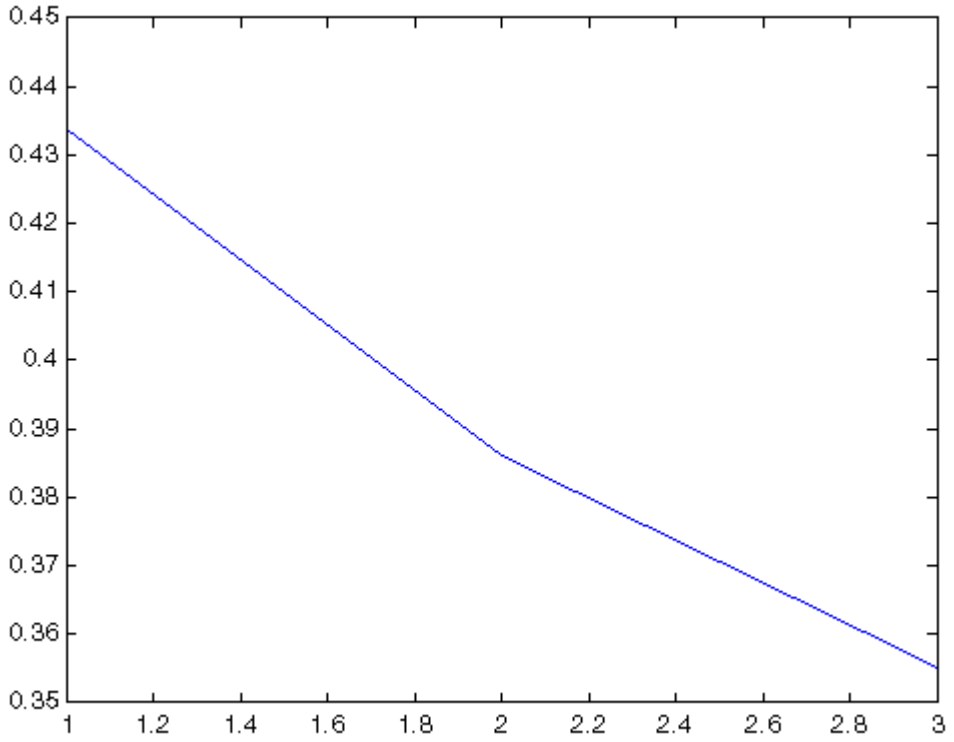
(Greater value = Better accuracy)  lambda = 0.25



# Evaluations

**Example:** Diversity within result set on 3 iterations

(Greater value = Better diversity) lambda = 0.25



# Evaluations

Clearly, accuracy is not acceptable

This is due to how we sampled the data

Most of the relevant documents are lost

Another way of sampling:

Pick 20 seeds with highest coupling factors

For each, include a 100-nearest neighbor set

2020 samples in total

# Evaluations

+10 hrs of preprocessing

Both accuracy and diversity increased significantly

For example, articles from “Klasik Türk müziği makamları” category are all included, and retrieved as similar articles

(with similarity: 0.75, diversity: 1.4)

# Evaluations

However, the similarity of these groups degrade sharply

First few groups: “Tarihler”, “Anlam ayrım sayfaları”, “Allah'ın sıfatları”, “Kategori sayfaları”

These have accuracy levels up to 1.0

After this, accuracy drops below 0.4

# Conclusions

First-5 stemming + Cosine similarity did not properly work on Wikipedia data because:

Articles which are phrased in a repetitive way were deemed very similar

For the rest of the articles, having the same words did not mean having the same **context**

# Conclusions

**Future work:**

The algorithm should be tested on various other datasets, for which we have a ground truth

For Wikipedia, we might change our approach from a term-based similarity to trying to discover link connections (e.g. Tracking user 'paths' might provide better information on how articles are connected)

# Thank you for listening

Questions & Comments

# References


- Jaime Carbonell and Jade Goldstein. “The use of MMR, diversity-based reranking for reordering documents and producing summaries.” 1998
- Michael E Houle et al. “Can shared-neighbor distances defeat the curse of dimensionality?” In: Scientific and Statistical Database Management. Springer. 2010, pp. 482–500.
- TSCorpus <http://www.tscorpus.com>
- Bahaeddin Eravci and Hakan Ferhatosmanoglu. “Diversity based Relevance Feedback for Time Series Search”. In: Proceedings of the VLDB Endowment 7.2 (2013).