Terminology-based Text Embedding for Computing Document Similarities on Technical Content

Hamid Mirisaee, Eric Gaussier, Cedric Lagnier, Agnes Guerraz

July 2019

Outline

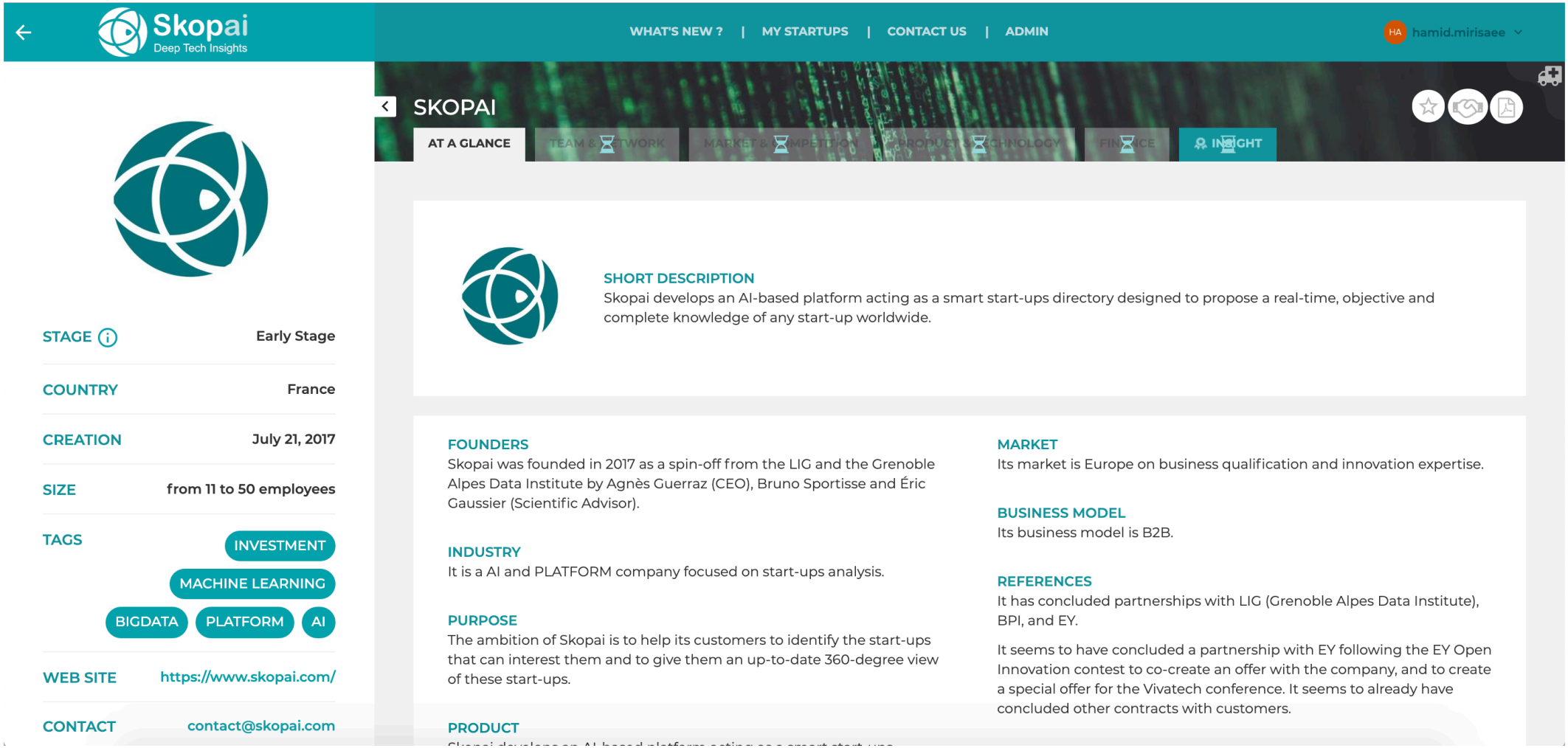
- Introduction
- Related work
- Proposed method
- Experiments
- Conclusion

Businesses need to work with startups
—> they need “good” info

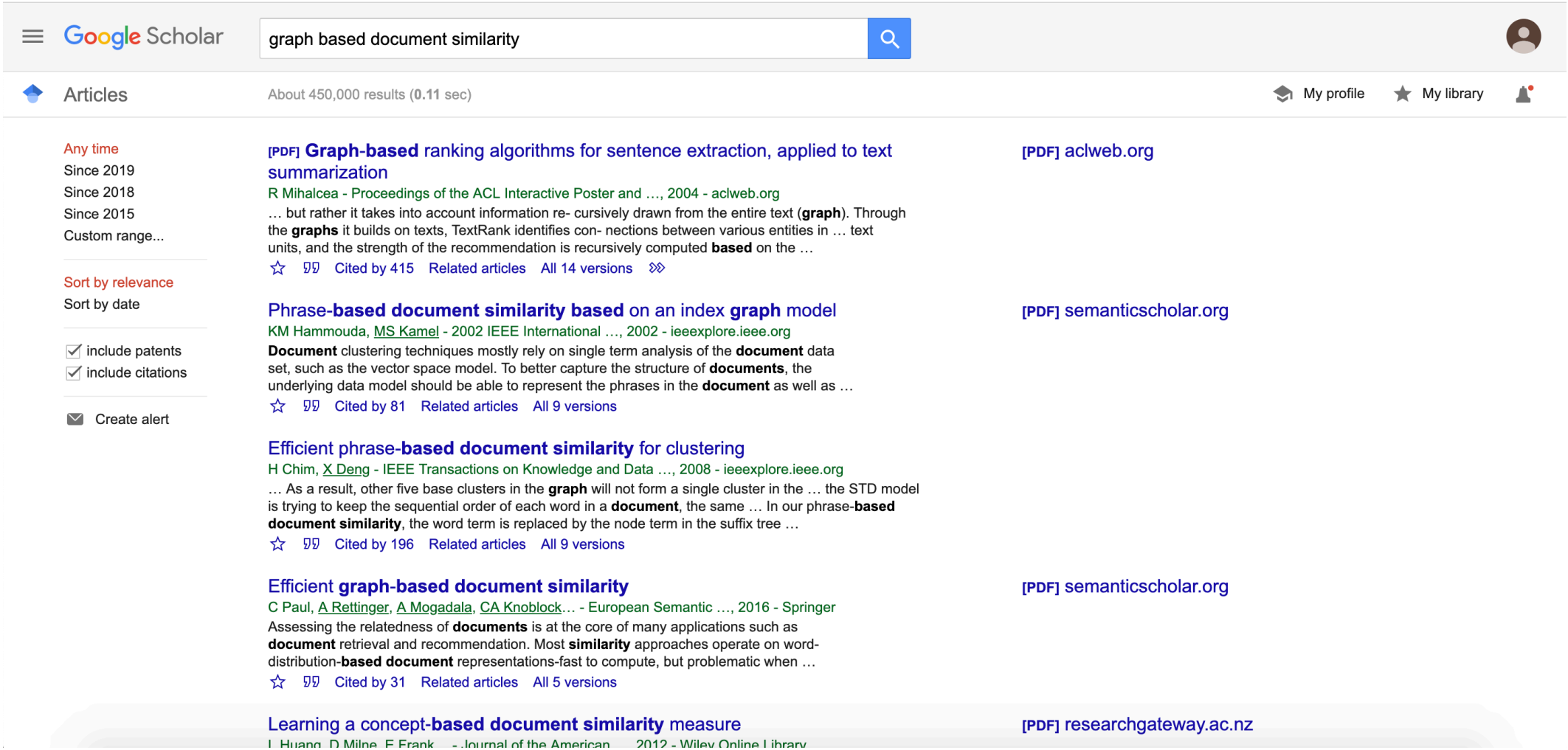
crawl the web (startups), process them, provide structured info

- Concours I-LAB.
- Concours de l’innovation.
- Partnership with LIG.

How does it look like?



Introduction



Introduction

- Finding relevant documents: on everyday basis.
- The principle is (almost) always the same:
    1. Define the space and the similarity measure.
    2. Take everything to that space.
    3. Find the closest documents in the space.
- Query ~ document:
    - closest papers to “Building Representative Composite Items” on arXiv?

How to define the space?

How to represent documents?

Related work (motivation?)

- Classic: tf-idf + cosine.
- KNN with tf-idf —> text classification [2014].
- Text representation: tf-idf, LSI and multiword —> text classification [2011].
- tf-idf is nice & helps in many tasks such as topic modelling, but…
- Let's be more contextual:
    - Go to word level (word2vec).
    - Represent words such that they carry semantical features (based on co-occurrence).
    - vec(King) — vec(Man) + vec(Queen) ~ vec(Woman)
    - most_simlar(car) = [cars, vehicle, automobile]
    - Many variations: doc2vec, sent2vec, combine with tf-idf, …

Motivation

Given a document, find similar documents in a "selective" fashion.

Do you *seriously* read the introduction and/or related work section of papers all the time?

Focus on the important parts of the document.

As mentioned previously, calculating document similarity and, consequently, finding similar documents is at the core of many ML tasks. Sometimes, however, focusing on the entire document may not lead to capturing desired similar documents as not all parts of a document have the same importance level. For instance, if a document describes a novel device for people suffering from diabetes, then taking the entire document may not necessarily result in the similar documents talking about the same particular issue, but rather about the medical domain in general. Note that this issue is different from

With that in mind, one can simply use the nodes of the main core to construct the keyphrases using different approaches, by for instance taking the words corresponding to the connected nodes. We use a slight modification of the k-core method in order to, first, extract the keyphrases of size 2, *i.e.* combination of two and only two terms, and, then, rank the sentences. Using those ranked sentences, we propose a technique to embed the document such that it encodes the main technical content of the document. The proposed approach is detailed in the following.

Proposed method: the big picture!

1. Extract keywords and/or keyphrases (composite keywords) of the document.

2. Score the sentences of the document based on the (composite) keywords they contain.

3. Pick a way to embed the sentences.

4. Embed the document as weighted average of the embeddings of its sentences.

Extracting (composite) keywords: use graphs!

Graph = Nodes + Edges

- Many problems can be formulated and/or interpreted via graph structure.
- In NLP:
  - Nodes —> entities (words, sentences, paragraphs, etc).
  - Edges —> relation with them (semantic, co-occurrence, etc).
- [Rousseau et al.] graph-of-words:
  - Nodes —> terms of the documents.
  - Edges —> if two terms co-occur in a fixed-size window.

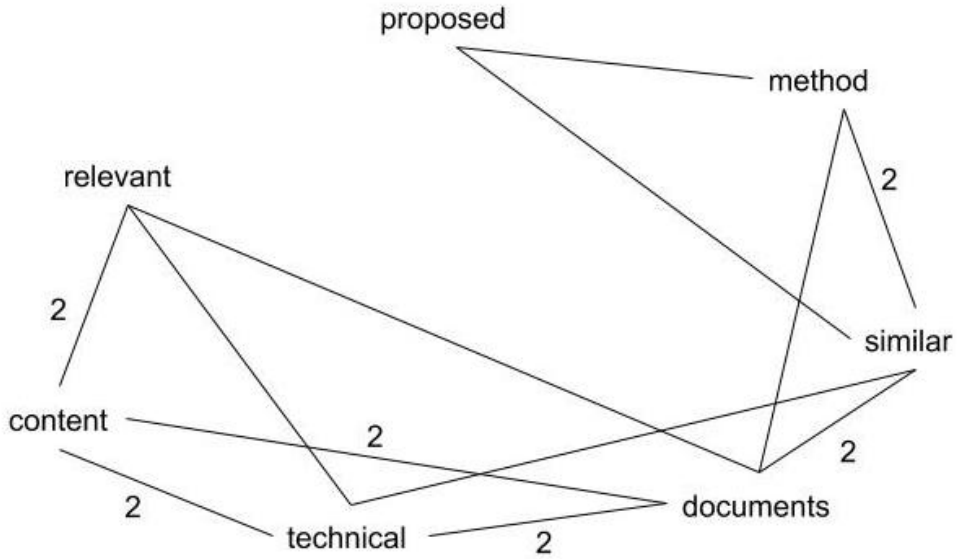
Graph-of-words

The proposed method can be used to find similar documents, particularly when the technical content is concerned for finding relevant documents.

proposed method similar documents technical content relevant documents

- proposed method similar —> {proposed, method}, {proposed, similar}, {method, similar}
- method similar documents —> {method, similar}, {method, document}, {similar, document}
- similar documents technical —> …
- documents technical content —> …
- technical content relevant —> …
- content relevant documents —> …

(un)weighted
(un)directed



K-core

$\mathcal{H}_k = (\mathcal{V}', \mathcal{E}')$ is called a k-core or a core of order k of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ iff $\mathcal{E}' \subset \mathcal{E}$, $\mathcal{V}' \subset \mathcal{V}$ and $\forall v \in \mathcal{V}'$, $Deg(v) \geq k$, and $\mathcal{H}_k$ is the maximal graph with such property.

- Core with max K —> **main core**
- **Idea**: it's important to be central, but your neighbours are also important!
- [Rousseau & Vazirgiannis]:
  - Main core —> **keywords & keyphrases**
  - Better than HITS and PageRank.
  - No hyperparameter.

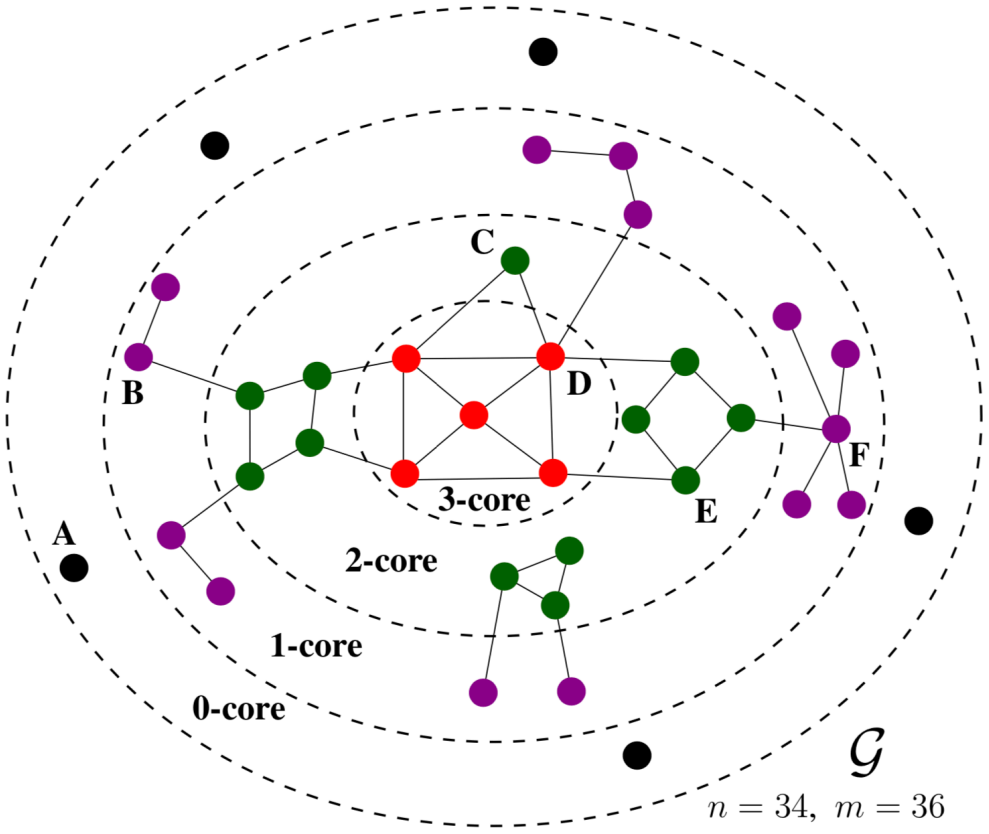
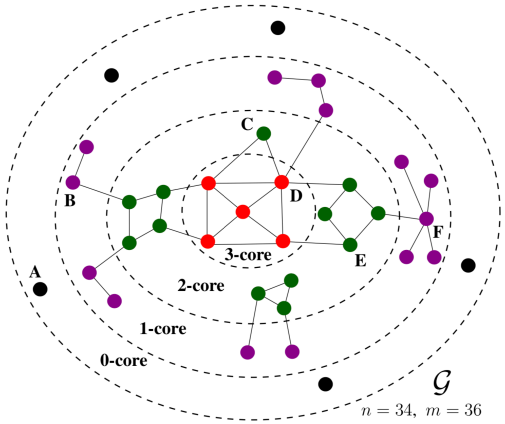


image from http://frncsrss.github.io/papers/rousseau-dissertation.pdf

TDE: Terminology-based Document Embedding (informally)

Preprocessing

Graph-of-words



Extracting keyphrases (of all cores)

artificial pancreas
insulin pump
clinical test

Score them based on their core & their edge weight

artificial pancreas (23.2)
insulin pump (19.4)
clinical test (14.6)

Score the sentences based on their keyphrases

We develop artificial pancreas which acts like an insulin pump.
(score = 23.2 + 19.4 = 42.6)

Via a clinical test, we evaluated our insulin pump.
(score = 19.4 + 14.6 = 34)

Do the math!

$$\vec{d} = \frac{42.6}{(42.6+34)} \times \vec{s_1} + \frac{34}{(42.6+34)} \times \vec{s_2} = 0.55 \times \vec{s_1} + 0.45\ \vec{s_2}$$

TDE: Terminology-based Document Embedding (formally)

$$\mathcal{C} = \{c_1, \cdots, c_k\}$$

$$\mathcal{T}_{c_i} = \{(t, t') | t \in c_i \wedge t' \in c_i\}$$

$$\vec{d} = \frac{1}{\sum_{s \in S} \Gamma(s)} \sum_{s \in S} \vec{s} \times \Gamma(s)$$

$$\Gamma(s) = \sum_{i=1}^{i=k} \sum_{\substack{(t,t') \in \mathcal{T}_{c_i} \\ (t,t') \in s}} \phi\big((t, t')\big)$$

$$\phi\big((t, t') \in \mathcal{T}_{c_i}\big) = Deg(t, t') \times F(c_i)$$

$$F(c_i) = (k - i + 1)^{-1}$$

ALL CORES
ONLY KEYPHRASES
ONLY 2-WORD

Algorithm 1 Terminology-based Document Embedding

Input: Set S containing all sentences of document d,
$\mathcal{T}_{c_i}$ $(1 \leq i \leq k)$: keyphrases of each core
Output: $\vec{d}$: the embedding of d

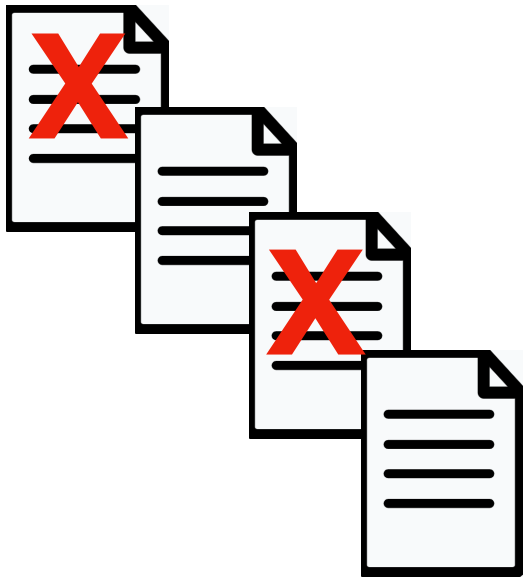
$w = 0$
$\vec{d} = \vec{0}$
for all $s \in S$ **do**
  $w_s = 0$
  for $i = 1$ to k **do**
    for all $(t, t') \in \mathcal{T}_{c_i}$ **do**
      if $(t, t') \in s$ **then**
        $w_s = w_s + \frac{Deg(t,t')}{i}$
      end if
    end for
  end for
  $\vec{d} = \vec{d} + (w_s \times \vec{s})$ \\ $\vec{s}$ is the embedding of s
  $w = w + w_s$
end for
$\vec{d} = \frac{\vec{d}}{w}$
RETURN $\vec{d}$

Experiments

- Baselines:
  - doc2vec**:** directly embed a document.
  - TWA**:** tf-idf weighted average of words of the document.
- TDE: how to represent a sentence?
  - sent2vec: learn a model to embed sentences. TDE_{s2v}
  - (tf-)idf weighted average of it's words. TDE_{iw}

Experiments: dataset

- Crawling websites of 68K startups (3.4M pages).
- Filter non-English, take pages with texts —> 43K startups with 2.8M pages.
- Document = combination of ***some*** pages of the startup.



Experiments: training, evaluation & results

- **100 documents**, four **domains**: {medical, agriculture, energy, biology}, **scores** $\{1, \ldots, 5\}$.
- Metric: Normalized Discounted Cumulative Gain, **NDCG@1**, **NDCG@5**.

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} \qquad NDCG_p = \frac{DCG_p}{IDCG_p}$$

- Train the doc2vec on the dataset.
- Train the sent2vec on all the sentences (omitting the stopwords) of the dataset.

	D2V	TWA	TDE_{iw}	TDE_{s2v}
NDCG@1	0.26	0.54	0.63	**0.69**
NDCG@5	0.24	0.60	0.60	**0.65**

Experiments: more details

HERE

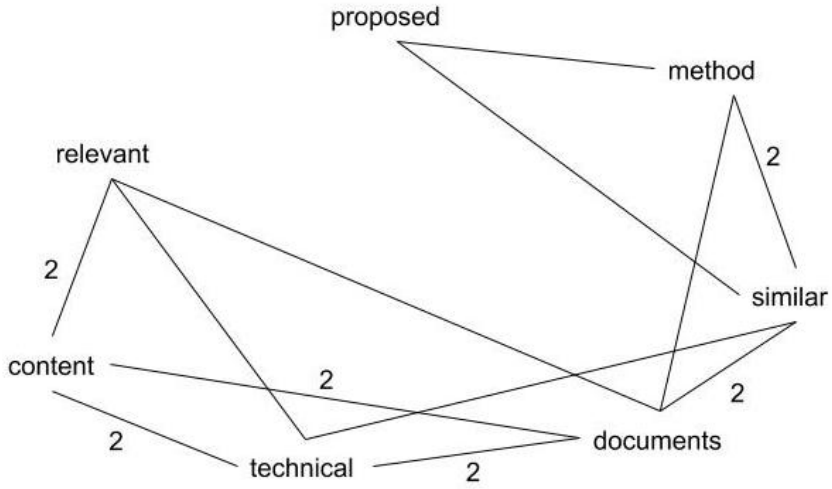
sentences —> only English, no stopwords.

IN PRACTICE

- We use TDE_{iw}
- Multilingual parallel w2vec models (EN, FR, DE, ES).
- Only nouns & adj.
- Check if the keyphrase *is actually* a keyphrase.
  - language specific RE checking (NN-NN, NN-ADJ, …)
  - imprimente 3D, 3D printer, satellite de communication, …

> The proposed method can be used to find similar documents, particularly when the technical content is concerned for finding relevant documents.

- proposed method similar
- method similar documents
- similar documents technical
- documents technical content
- technical content relevant
- content relevant documents



Experiments: examples

SKOPAI (top-3 sentences)

We have a strong experience of the innovation ecosystem and how it works: research and technology transfer in academia, R&D in tech or industry corporations, venture capital or government innovation policy.

Nous construisons une plate-forme de référence pour la technologie, fournissant en temps réel une connaissance complète sur toute startup dans le monde entier.

Startup assessment depends on the quality and context of the person performing it – for example chief innovation officer, product managers, R&D engineers, investors, buyers, legals, etc.

SKOPAI (top-3 similars)

We build on a longstanding experience in corporate, startup, not-for-profit and public service organizations. We stand for collaboration that allows businesses to thrive. This is why we focus on enabling alliances that foster innovation and redesign business models.

We are Building the only all-in-one innovation & start-up ecosystem platform for the cloud power future.

startup assessment depends on the quality and context of the person performing it – for example chief innovation officer, product managers, R&D engineers, investors, buyers, legals, etc.

Conclusions

- Use graph-based methods to extract similar documents.
- The general framework is valid for any sort of sentence embedding.
- Focus on the technical content (via keyphrases).
- Outperform the state-of-the-art in terms of NDCG.
- Could be also used to rank sentences.

Skopai
Deep Tech Insights

Thank you!

We are hiring…

https://www.skopai.com/join-us/

Full Stack Engineer
Data Scientist