Industry network using large companies corpus

Project in data collection lab

Technion, 2022

Snir Lugassy, Alon Shiri

1. Introduction

Our goal is to process a large corpus scraped from websites of companies and to construct a weighted directed graph that represents the relations between different industries, we will refer to this graph as *Industry Network*.

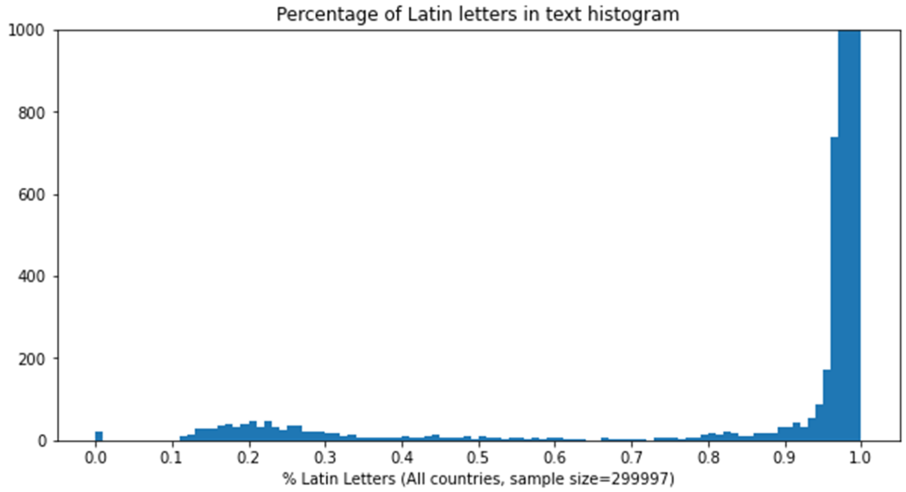
The Industry network brings insights about the corpus and underlying semantic relations between industries, moreover, it can be later used in other tasks such as clustering or recommendation systems.

Our pipeline can be easily transferred to other domains such as products, twits, or academic papers, to map the categories of the domain.

2. Methods

Text filtering

- We kept only companies from the USA
- We kept text with more than 90% Latin letters, justified by the following histogram:



The above filters aim to keep mostly English text to reduce the text vector space dimension and reduce the impact of foreign languages on our results.
Eventually, the corpus contains 1,216,967 documents.

Text normalization

- Remove HTML tags
- Remove special characters
- Remove digits
- Remove redundant whitespace
- Remove terms with less than 3 characters or more than 14 characters
- Remove stopwords
- Convert to lowercase

Word embedding

We used tf-idf representation for each document in the corpus, in previous assignments and small scale testings we conducted shows that tf-idf captures topics well and is suitable for the task at hand.

In the tf-idf model, we ignored terms with df greater than 70%, we considered them as empirical stopwords.

The final vector space dimension: 6,047,551

Pairing companies for companies network

Our first step for building the Industry network is building the Company network, which will contain relations between companies.

We will define the relation between companies using cosine similarity between the text vectors, and append a new column called “related_to” which is the nearest company in the corpus.

Finding the closest company is the Nearest Neighbor Search (NNS) problem which is practically not feasible in high dimensions in terms of space and time, therefore we must solve it using advanced methods.

Naive NNS:

The complexity of finding all closest pairs $O(n^2)$ where n is the number of documents.
Our matrix dimensions: 1,216,967 x 6,047,551

Required vectors operations: $\binom{1.21 \cdot 10^6}{2} \approx 7.32\ 10^{11}$

Efficiently, cosine similarity in this space takes around 10 ms, therefore on a single thread finding all closest pairs will converge in more than 5570 years.

Improved NNS:

There are improved methods to solve the NNS problem (e.g., using space partitions), but they suffer from bad performance on high dimensions ("curse of dimensionality") and lack of support in sparse vectors.

Approximate NNS:

Instead of finding the exact K Nearest Neighbors, find K neighbors which are close enough to the queried vector (suitable for most practical applications).
The distance between the query and the approximate NN is bounded by: $c \cdot min_x D(q, x)$

We used pysparnn library, developed by Facebook research.

Using a tree structure, number of candidates reduces to $\sqrt{N}$ using sparse vectors.
Now it takes around 37 ms to find 1-NN for one company.
That's how we found all 1-NN in around 11 hours, with over 1.21 million queries on a single CPU without distributed computations.

Constructing the Company network

Let C be the set of all companies.
Using the "related_to" column we can now construct a network of companies G':

$$G' = (C, E')$$

$$E' = \{(x, y): x, y \in C \wedge x.related\_to = y\}$$

Deriving the Industry network

Let I be the set of all industries, and for a company c let $Ind(c)$ be the industry of c.

We will reduce the company network G' to our target Industry network using the following graph definition:

$$G = (I, E)$$

$$E = I \times I$$

$$w(i,j) = |\{(x,y) \in G' : Ind(x) = i \wedge Ind(y) = j\}|$$

If $w(i,j) = 0$ we will ignore the edge (i,j)

In other words, we defined the weight between two industries as the count of companies related between the industries.

Since text scraped from websites is likely to contain many general phrases and will not always hold information about the industry, the relations between industries will be noisy and at the single company level will not always be correct, we need to set a threshold for the weight which will leave only the strong relations.

Let t be the threshold, We will adjust the weights as follows:

$$w_t(i,j) = w(i,j) \cdot I\{w(i,j) > t\}$$

The threshold isn't necessarily a constant, but can also be some function of i or j, and the adjusted weights will be:

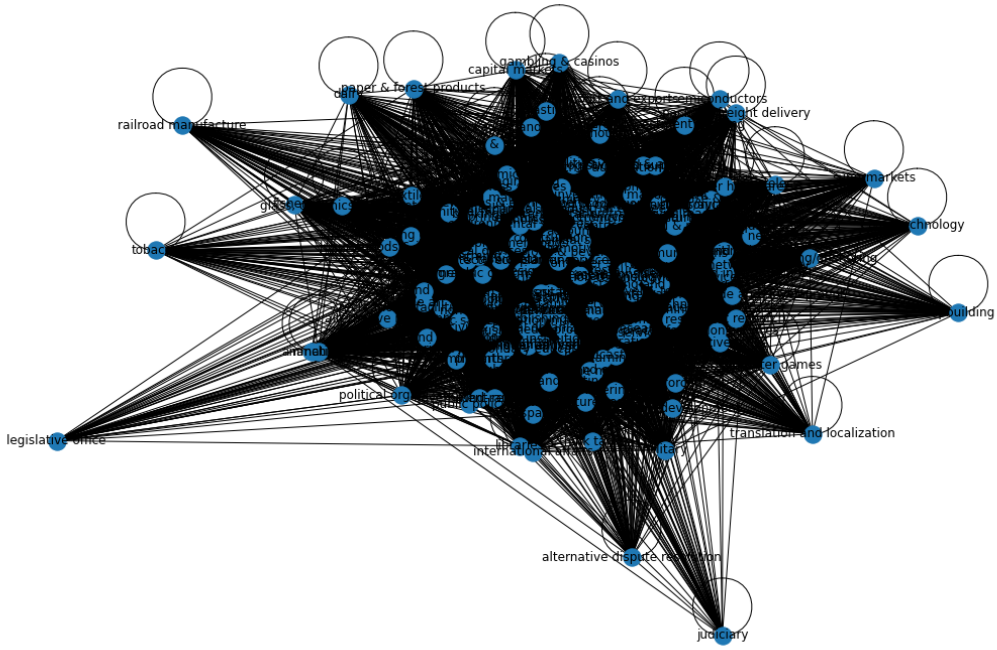
$$w_t(i,j) = w(i,j) \cdot I\{w(i,j) > t(i,j)\}$$

In the network analysis section, we will justify this adjustment and demonstrate picking some empirical threshold based on the raw network.

3. Results

Network analysis

Raw network (without threshold)



Connected components: 1

Nodes: 147

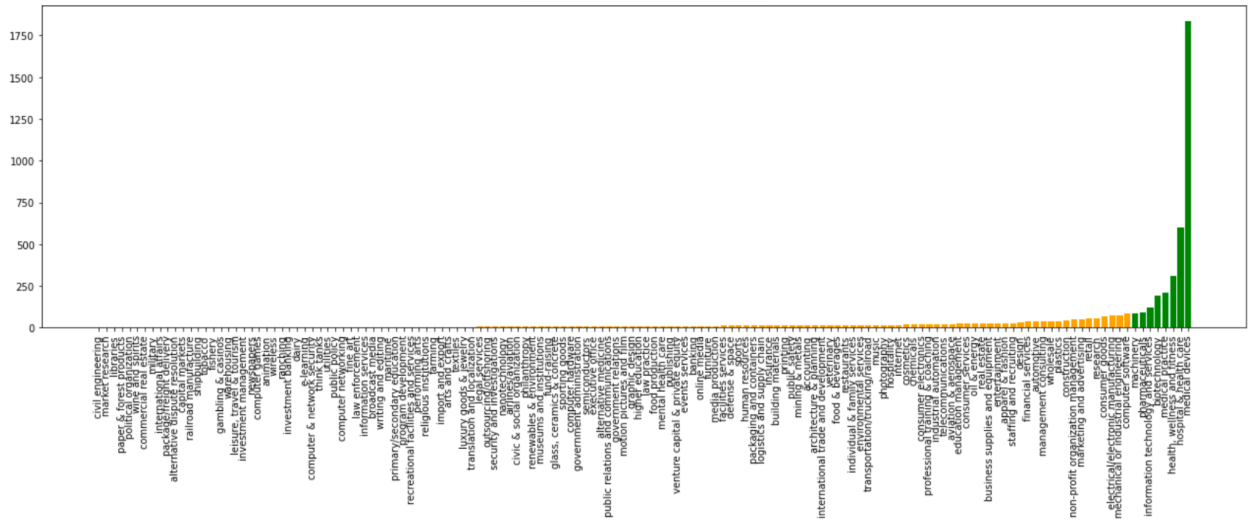
Edges: 10010

Mean degree: 136.19

The raw network is tangled, each industry is related to most of the other industries which will distract our visual inference of the graph.

We randomly picked nodes from the graph, examined their outgoing edges distribution and concluded that they share the same property of long-tail distribution.

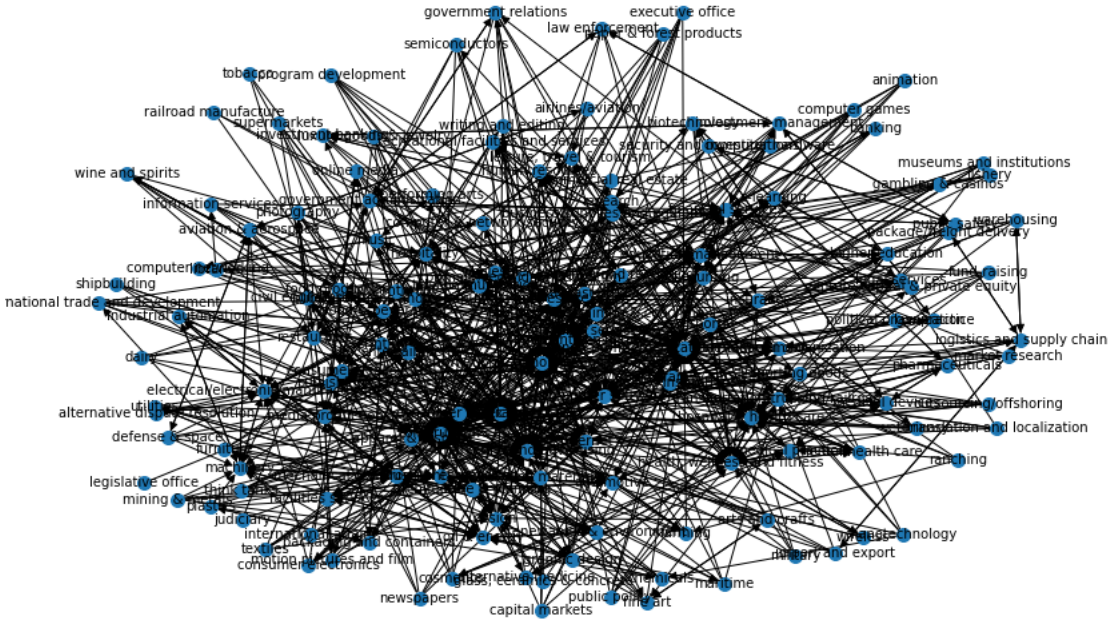
Medical devices - outgoing edges weights distribution



The above plot shows the weights of outgoing edges of the industry “medical devices”, we colored in green the top 5% weights and in orange the lower 95% weights.
First, it demonstrates the long-tail distribution of outgoing edges weights, and second, the industries colored in green are known to be related to medical devices (e.g., Biotechnology) and the industries in the lower weights are less related (e.g., Real-estate).

Empirically, the major drop in weights was observed around the 0.95 quantile, therefore we chose the threshold for (i, j) as a function i, In other words, we ignored weights smaller than the 0.95 quantile.

Filtered network (with threshold)



Strongly connected components: 47

Weakly connected components: 1

Nodes: 147

Edges: 914

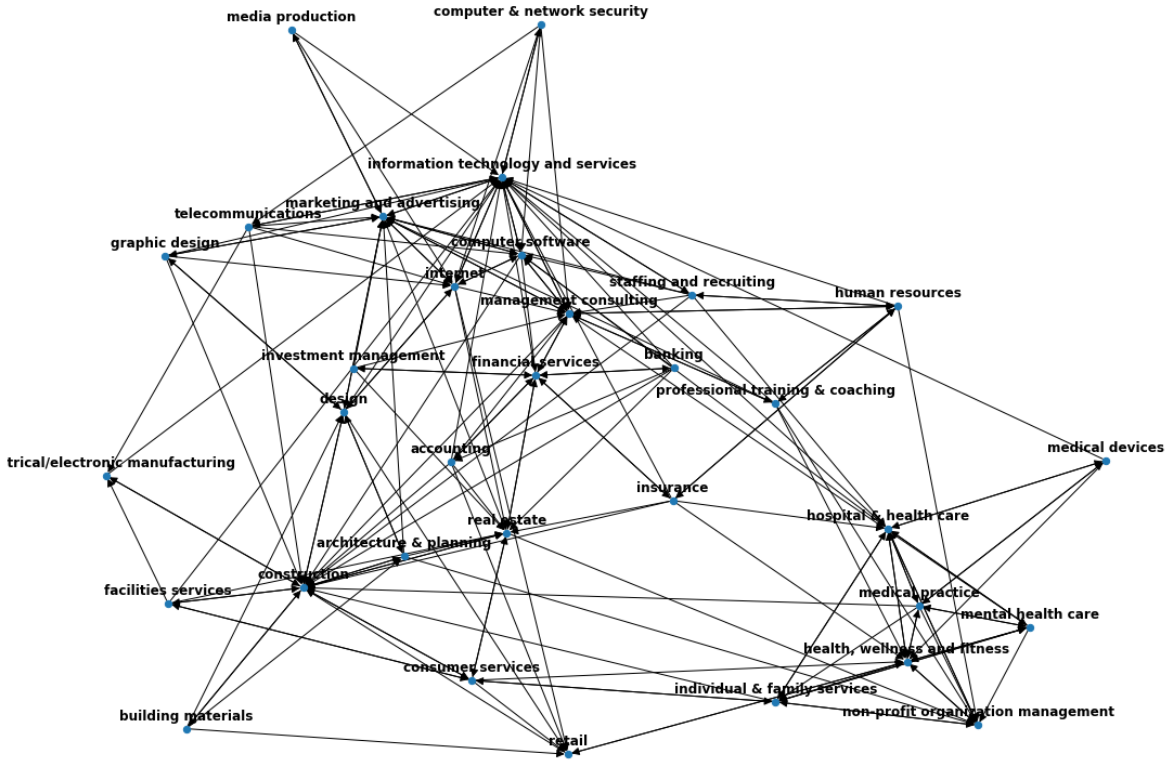
Mean degree: 12.43

Setting the threshold resulted in a more concise network that only holds the important relations, which separated the network into 47 strongly connected components.

Moreover, we removed self-loops since they hold trivial information (each industry is related to itself).

Analysis of a subnet

Let us take a closer look at a subnet, which is constructed by taking “computer software” and second-degree neighbors.



We observe that the Industry network captures groups of industries such as IT related in the upper sections of the network and medical related in the lower right section, which settles with our knowledge about the world and assumptions.

Distance between industries

Using the industry network we can now search for the distance between two industries using weighted shortest-path:

- “Construction” to “Medical practice”: hops distance of 4, and goes through:
  Construction ⇒ Consumer services ⇒ Health, wellness and fitness ⇒ Medical practice
- “Graphic design” to “Semiconductors”: There isn’t a path between these industries that suggests that they are not related at all.

4. Discussion

Despite the sparsity of the vector space, and very noisy texts the industry network settles with our knowledge about industrial relations and brings insights about the corpus.
We were careful at each step to preserve and refine the underlying topics, in our case the industry of the company.

Although we couldn't evaluate our network, we are satisfied with the results based on visual insights and testing the output against common knowledge.

We learned about solving complicated problems regarding the processing of large text and sparse data on a basic machine. Moreover, we improved our understanding of network analysis.

Further advancements

1. Using translation methods to include data written in foreign languages.
2. Using snippets of text instead of the whole document, as we practice in previous assignments.
3. Setting the thresholds using automated processing instead of manual values based on empirical results.
4. Comparing results while using different word embedding methods (e.g., fasttext)
5. Testing the pipeline on multiple textual datasets.
6. Implement a recommendation system while using the network as a baseline

5. Responsibilities

Alon: More responsible for the text processing and embedding
Snir: More responsible for the network construction and analysis

We worked both together and in parallel during this project.

References

Andoni, Alexandr, et al. "Approximate Nearest Neighbor Search in High Dimensions." *ArXiv.org*, 26 June 2018, https://arxiv.org/abs/1806.09823.

Facebook Research. 2018. "pysparnn", Approximate Nearest Neighbor Search for Sparse Data in Python, https://github.com/facebookresearch/pysparnn