NLP Information Retrieval with Deep Learning

Nathan Bosch

Overview

1. What is information retrieval?
2. Information Retrieval of Text
3. Deep Learning IR approaches
4. Sentence Transformers
5. Problem Introduction

What is information retrieval?

- Example problem: I have a dataset of 10 thousand research papers potentially containing relevant information to my research
  - How can I find this relevant information?
    - Manual solution: Read through all ten thousand papers
    - Better: Identify all relevant documents and read through those (we will focus on this)
    - Best: Automatically retrieve an answer from relevant documents (the user does not need to read relevant documents)
  - Key Idea: Extract relevant information from large datasets
- This is relevant in retrieving any type of data (e.g., images and videos), but we will focus on textual data
  - The models used change depending on the application domain

What is information retrieval?

Simple problem definition:

- A query **Q**
- A set of documents $\boldsymbol{D_{1..n}}$
- A scoring function ***s*(Q,** $\boldsymbol{D_i}$**)**
  - **s(Q, D)** should be higher when D is a more relevant document to the user query
- For all documents, compute s(Q, D)
- Sort by score
- Return the top-N documents

Basic Information Retrieval

When it comes to text, how can we tell if a document is relevant for a query?

- Retrieve all documents which contain the same words as the words in the query
  - Rank based on percentage overlap
- Problem: What if some words are more relevant than others?
  - Extension: Assign scores to words based on their frequency: TF-IDF, BM-25

Basic Information Retrieval

TF-IDF = TF*IDF:

- TF: Term Frequency, how many times does a term appear in a document?
  - $TF(term, doc) = \frac{number\ of\ times\ term\ appears\ in\ doc}{number\ of\ terms\ in\ doc}$
  - TF will be high when the term appears more frequently in the document
- IDF: Inverse Document Frequency, how many times does a term appear in all documents?
  - Terms which appear less frequently in other documents will have a higher IDF
  - $IDF(term) = \log(\frac{number\ of\ documents}{number\ of\ documents\ in\ which\ term\ appears})$
  - The log is taken to reduce extreme IDF values
- TF-IDF for a term is high when the term appears frequently in a document and infrequently in other documents

Basic Information Retrieval

Example:

- 3 Documents
  - “I like to take long walks on the beach.”
  - “The house is dull.”
  - “Information retrieval is the best!”
- What is the TF-IDF of “the”?
  - IDF = log(3/3) = 0, hence for all documents TF-IDF is 0
  - Why? If a word appears in all documents it is essentially meaningless for retrieval
- What about “beach” in document 1?
  - IDF = log(3/1) = 1.1
  - TF = 1/9
  - TF-IDF = 0.12. Low, but relevant during search.

Basic Information Retrieval

Interesting thought:

- Say I have a vocabulary of words ***V***
  - E.g., [“the”, “great”, “dragon”]. Say IDF = [0.2, 1.4, 2.2]
- We could express each document as a vector
  - Sentence 1: “the dragon” = [0.1, 0.0, 1.1]
  - Sentence 2: “the great” = [0.1, 0.7, 0.0]
  - Sentence 3: “the great dragon… Dragon? Dragon!” = [0.04, 0.28, 1.32]
- Query sentence: “the… the dragon!” Which sentence is most similar/relevant?
  - Use cosine similarity: $\frac{||A||\cdot||B||}{||A||\times||B||}$
  - Measures the angle between two vectors, A and B
  - Commonly used for text data because vectors are often sparse

Basic Information Retrieval

- We could express each document as a vector
  - Sentence 1: “the dragon” = [0.1, 0.0, 1.1]
  - Sentence 2: “the great” = [0.1, 0.7, 0.0]
  - Sentence 3: “the great dragon… Dragon? Dragon!” = [0.04, 0.28, 1.32]
- Query sentence: “the… the dragon!” Which sentence is most similar/relevant?
  - Query TF-IDF: [0.133, 0.0, 0.733]
  - **Sentence 1 similarity: 0.996**
  - Sentence 2 similarity: 0.025
  - Sentence 3 similarity: 0.967
- We are now working in the vector space model

Basic Information Retrieval

TF-IDF and vectors are nice and efficient, but how can we capture:

a) Different words which mean the same thing (e.g., “dog” and “hound”)
b) Words which mean different things in different contexts
c) Word order and larger semantic meaning in sentences

The solution: We want to encode the context and dependencies between words with deep learning

- Previously: LSTMs/GRUs, Sequence to Sequence models
- Now: Transformers (most frequently BERT-based)

Deep Learning Intuition

- Words and sentences are very complex (and vectors quickly become very high dimensional)
    - We want words (and eventually sentences/paragraphs) which contain similar semantic information to look similar as vector (i.e., have high cosine similarity)
    - For example, the sentences “That is a dog” and “That is a hound” should look very similar, but TF-IDF will miss the semantic relationship between “dog” and “hound”
    - Other approaches exist to find these relationships, e.g., LSI (often used in information retrieval)
    - However, with deep learning we can get much better

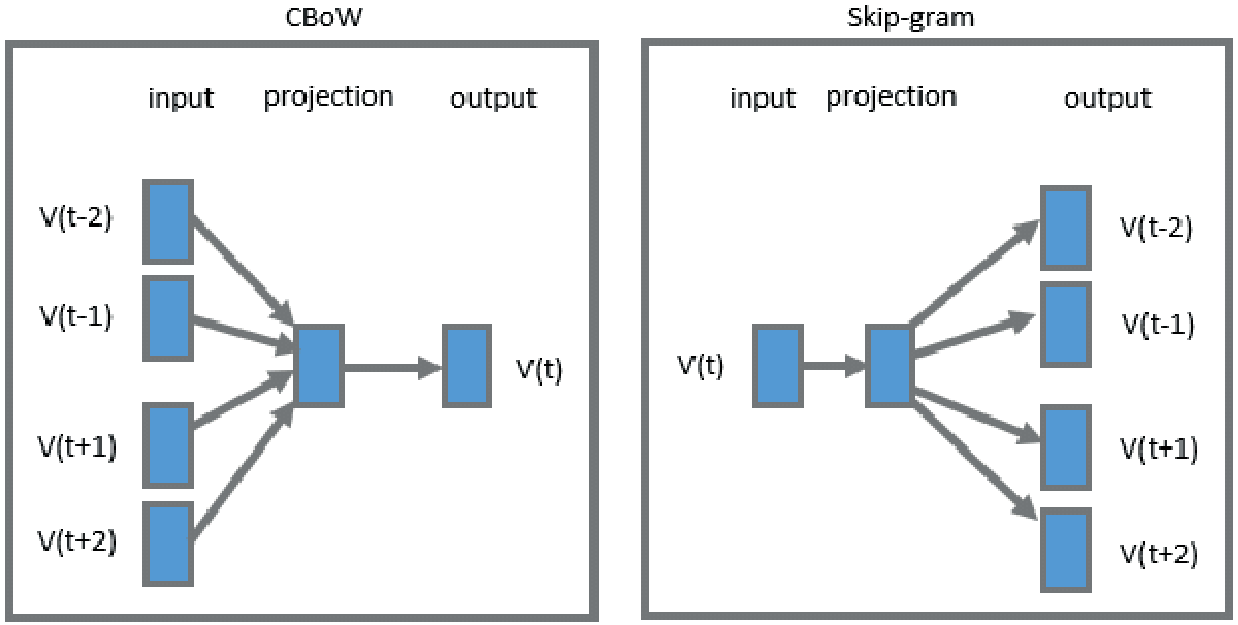
Example: Word2Vec

- Two options:
  - Use a word to predict the words around it (i.e., input="dog", output="that", "fat", …, "is", "awesome"
  - Use context words to predict a word (i.e., input="that", "fat", …, "is", "awesome", output="dog"

- Very influential in NLP, because we find that the model learns interesting semantic information in the text

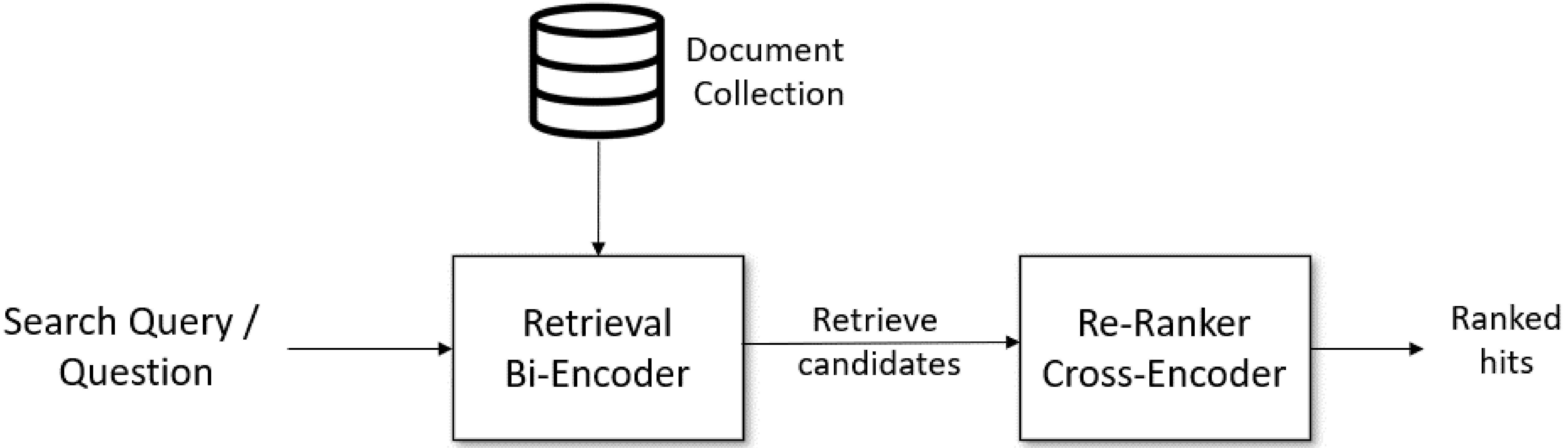
- Example:
  - When taking the vectors of "Queen", "Woman", and "Man" we can compute vec("Queen")-vec("Woman")+vec("Man")
  - The vector closest to the output of that operation is the vector for "King"

Imagine the extension of this to sentences and full documents with LSTMs and, thereafter, transformers



Neural Ranking Approaches

1. Representation Based Retrieval
2. Interaction Based Retrieval
3. Hybrid/Combined Representation and Interaction Based Retrieval



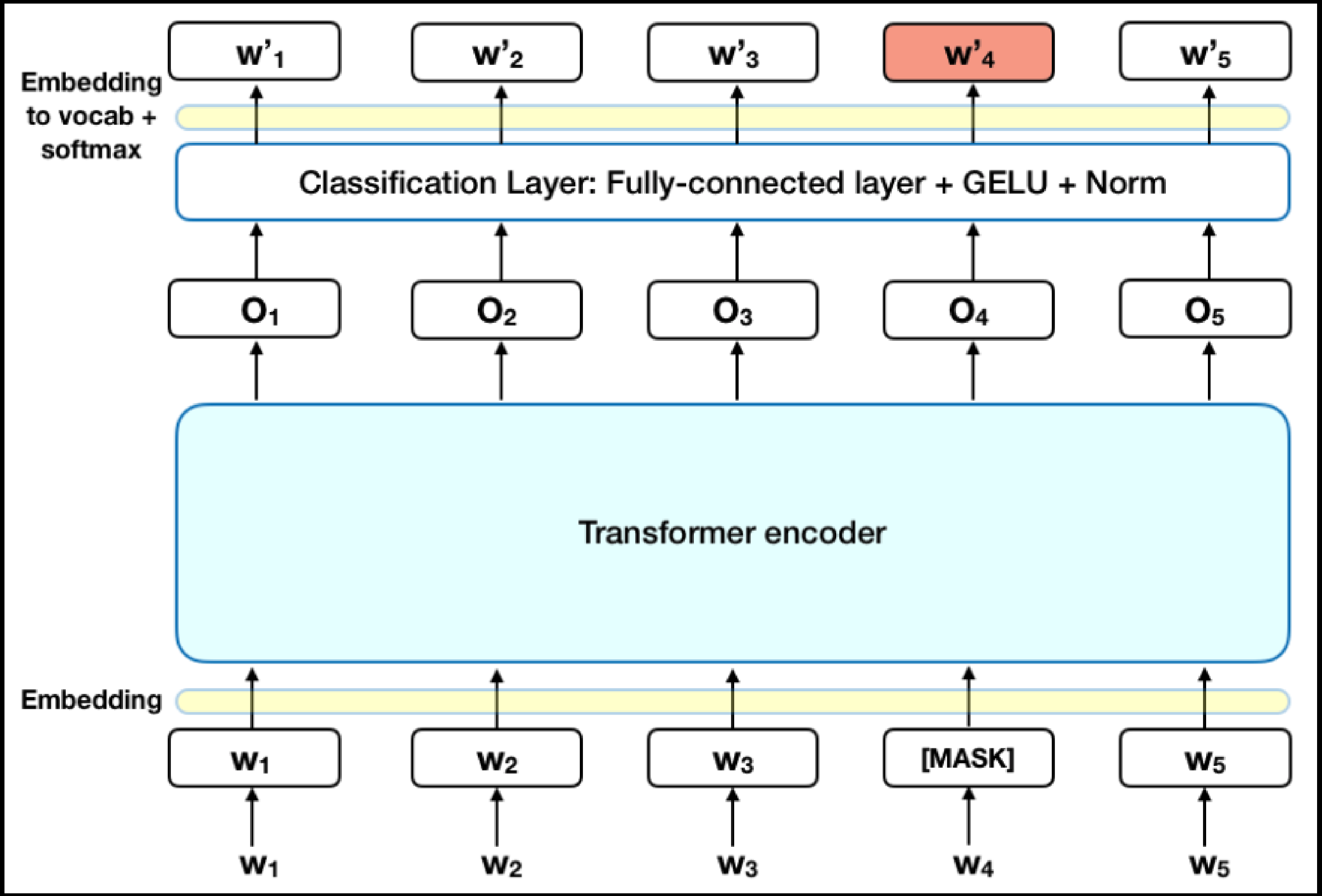
Deep Learning Approaches

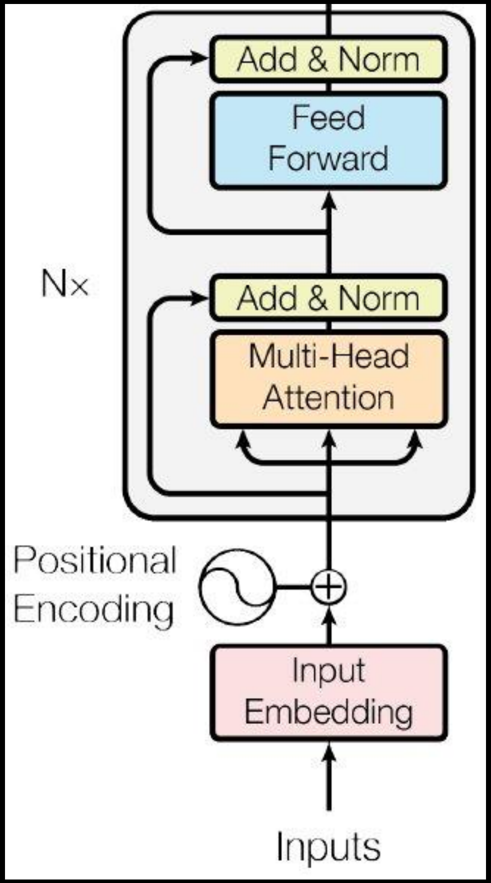
- Representation Based Approach
  - The vector representation of the query and the most relevant documents should be the same
- General Idea:
  - Generate a vector representation (embedding) of the query
  - Generate a vector representation (embedding) of all documents
  - Compare the query embedding with all document embeddings through cosine similarity
    - Cosine Similarity (Q, D): $\frac{Q \cdot D}{||Q||\,||D||}$
  - Sort and retrieve the top documents based on the similarity

Vector Representation with BERT Models

- BERT uses only the encoder of a transformer network
- Given a sequence of words, mask 15% of words
  - s="The [mask] brown fox [mask] over the lazy [mask]"
  - BERT(s) = "The [quick] brown fox [jumps] over the lazy [dog]"
- During training, perform next sentence prediction
  - s="[CLS] The [mask] brown fox [SEP] [mask] over the lazy [mask] [SEP]"
  - Model is trained on sentence pairs, returning the probability of if a sentence follows from the previous sentence
- Next sentence prediction and masking is used during training
- With sufficient data and parameters, this achieves great performance

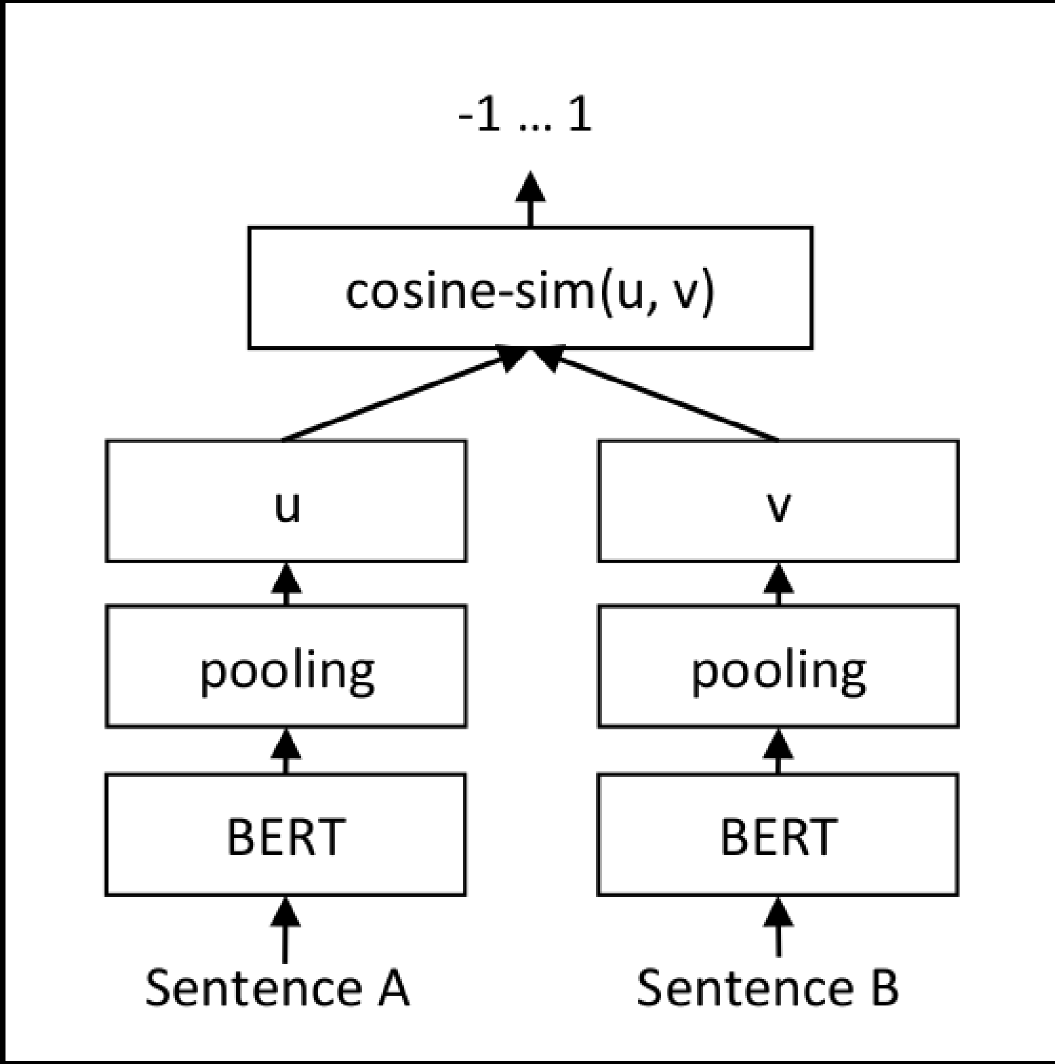
Vector Representation with BERT Models





Vector Representation with BERT Models

- Using a Siamese network training structure, BERT models can be used to maximize the similarity between a query and document, even when different words are used
- Say Sentence A = Q and Sentence B = D1
  - D1 may be much larger than the query, so the similarity found by a standard BERT model might be quite low
  - With this Siamese network training we can avoid this problem
- Models like this can be trained with question answer data, such as MS Marco, e.g.,
  - Q = What is a corporation?
  - A (or D) = A corporation is a company or group of people authorized to act as a single entity and recognized as such in law.



Sentence Transformers Library

- Can be installed through pip or conda
  - pip install -U sentence-transformers
  - https://www.sbert.net/docs/installation.html
- Has several pretrained deep learning models available, some trained on question answer data in a Siamese network structure
  - https://www.sbert.net/docs/pretrained_models.html#semantic-search
- Lots of information and examples on information retrieval, neural ranking, semantic search
  - https://www.sbert.net/examples/applications/semantic-search/README.html
  - https://www.sbert.net/examples/applications/retrieve_rerank/README.html

Task Description

- We have 10 thousand papers from arXiv that we want to build a deep learning retrieval system around
- Given a query, such as “bidirectional transformer networks”, find related papers using pretrained deep learning models in the sentence_transformer package
- We have some sample code and data available which you can download

- Feel free to ask questions, we’ll try to help out!
- We’ll have a discussion in 15-30 minutes