Vector representation of text

Lecture 2

Plan

- Basic NLP terminology: tokenisation, normaliztion, n-grams, stopwords, vocabularies
- The task of text vectorisation
- Sparse vector spaces, one hot encoding, TF-IDF
- Language Models

Dependencies

- Open shell
- git clone …
- cd aca-nlp
- cd lecture2
- pip install -r requirements.txt
- python -m spacy download en
- python -m spacy download en_core_web_sm

Tokenization

- **token** — a meaningful unit of the language with a specific meaning. Usually, but not always, word is a token

- **tokenisation** — a process of transformation of text to an ordered sequence of token

Can be achieve with the following:

- Simple split by spaces or other delimiters

- Regular expressions

- Complex models, specific to languages (nlkt, spaCy)

Exercise 1: tokenisation (10 min)

tokenize_by_split(text)

remove_punkt_and_tokenize_by_split(text)

tokenize_by_regex(text)

tokenize_by_punkt_model(text)

Normalisation

- **Normalisation** — process of brining the text to its canonical form
- Brining all to lower case (Hello -> hello)
- Stemming (working -> work)
- Removal of diacritic symbols (résumé -> resume)

No universal method applicable for all use cases

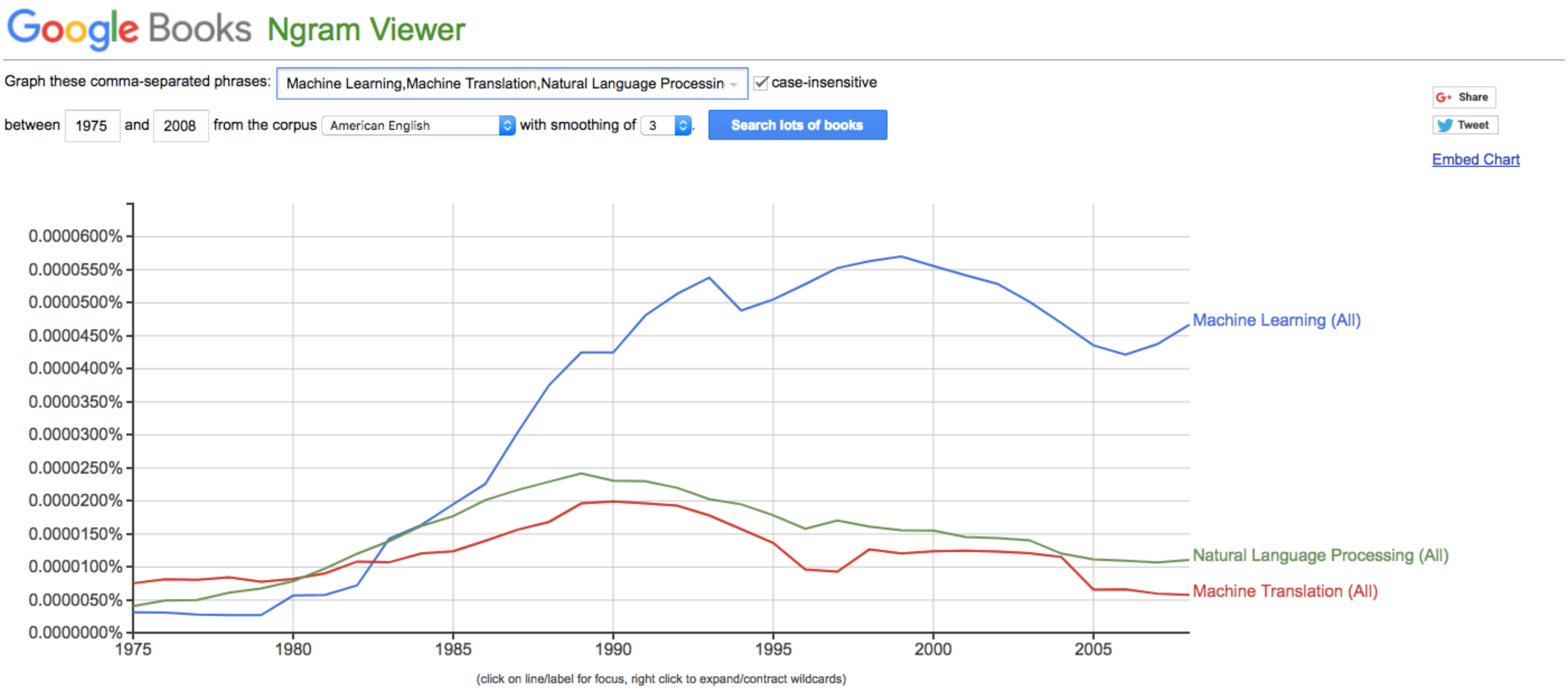
N-grams

- n-gram — a sequence of n elements from a certain set (a set of tokens, for example)
  - natural — unigram
  - natural language – bigram
  - natural language processing — trigram

Widely used in

- language modelling
- information retrieval

Google Ngram Viewer



https://books.google.com/ngrams

Stopwords

- Words that should be removed from the text before the analysis
- Usually, most frequent words of the language (a, the, to)
- articles, propositions, particles, etc
- Can improve or decrease the model's performance

Exercise 2: (10 min)

```
# remove stopwords from the list of 'lowered_tokens'
stopword_filtered_tokens = [tok for tok in lowered_tokens if tok not in stopwords]

# turn your filtered list of unigrams into a list of bigrams, joint by whitespace
filtered_bigrams = [' '.join(bigram) for bigram in list(nltk.ngrams(stopword_filtered_tokens, 2))]

# now count the occurances of bigrams using a new Counter instance
bigram_counter = collections.Counter(filtered_bigrams)
```

Vocabularies

- Modern english consists of 13M words
- Usually, you don’t need nor can use them to build NLP models
- We use the frequent words in the corpus to make up the vocabulary
- Usually, vocabulary is notates as |V|, V — vocabulary. Ranges in 10^4-10^6
- **vocabulary** sets each token a unique number in correspondence

Home exercise #1 : Vocabulary

On meaning of words

Definition: meaning

- Идеальное содержание, идея, сущность, предназначение, конечная цель (ценность) чего-либо
- Содержание знакового выражения; мысль, содержащаяся в словах (знаках, выражениях)

- How do we write down a meaning in a form that a computer can understand?

Text vectorisation

Machine learning models work with data as vectors

Given: a set of text documents $D = \{d_i, i = 1,2,...,n\}$

Goal: for each d_i from D set a coordinate s_i in Hilbert space S

Hilbert Space

Properties:

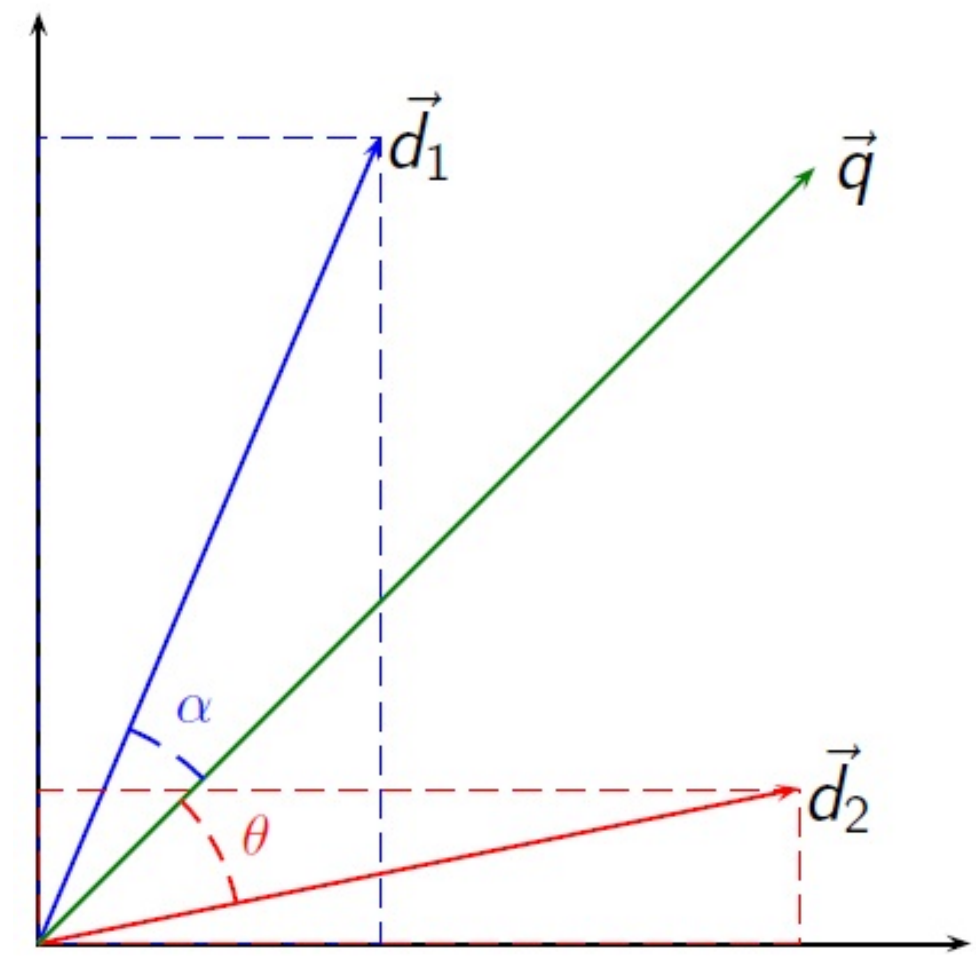
- for any two elements of the space **x, y** there is a rule to calculate a scalar product **(x, y)**
- this rule has to satisfy the following properties:
  - **(x, y) = (y, x)** - commutative property
  - **(x, y + z) = (x, y) + (x, z)** - distributive property
  - **(λx, y) = λ(x, y)** - for each real λ

Cosine distance

- The most frequently used metric of similarity between documents in vector space

- equivalent to scalar product, if documents vectors are normalised (norm == 1)

$$\cos\theta = \frac{\mathbf{d} \cdot \mathbf{q}}{\|\mathbf{d}\| \, \|\mathbf{q}\|}$$



one-hot encoding

- Treats each word as an atomic, independent symbols
- Each word is a space vector of dimension $\mathbf{R}^{|V| \times 1}$
- Word vector consists of zeros on all positions expect the value of the word in vocabulary
  V = {…
  ‘machine’: 5
  …}

  ‘machine’ =
  [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

model: Bag of Words

- We can use one-hot representation to encode whole text documents into vectors

- Example:
  (1) John likes to watch movies. Mary likes movies too.
  (2) John also likes to watch football games.

  (1) [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
  (2) [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]

model: TF-IDF

- Term Frequency - Inverse Document Frequency
- weighting scheme, used to utilise the importance of a word within the document
- *tf* * *idf* word's weight is proportional to the number of its usage in the document and inverse proportional to the frequency of it in all the document set

$$TF - IDF = f_{t,d} * \log(1 + \frac{N}{n_t + 1})$$

- decreases the weight of widely used words and increases the weight of the rare words, which results in a better vector representation

Exercise 4:
Information Retrieval

BoW: pros and cons

1. Computationally simple model based on linear algebra
2. Easy interpretation
3. Can incorporate the importance of a words
4. Can easily rank documents on it’s relevancy

1. High dimension of the space with a large vocabulary
2. No words semantics, words are independent
3. Ignores synonyms, polysemy
4. Can’t deal with typos
5. Loses the order of the words in the document

Language models

- Task: to get a probability of a given sequence of tokens
- Good for:

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

  - Machine translation P(I went home) > P(I went house)
  - Typos fix P (I went home) > P(I wens home)
  - Text generation, Q&A systems, chat-bots etc.

How to calculate it?

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P (“I went home for lunch”) =

P(I) x P(went|I) x P(home| I went) x P(for | I went home) x

P (lunch| I went home for)

How to really calculate it?

Count and divide?

$$P(\text{lunch} \mid \text{I went home for}) = \frac{\text{count(I went home for lunch)}}{\text{count(I went home for)}}$$

- Too many variations of possible sentences
- There is no such data to calculate a true probability

Markov assumption

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

P(lunch | I went home for) ~ P(lunch | for) (bi-gram)
P(lunch | I went home for) ~ P(lunch | home for)

Let's estimate the bigram probability

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $\quad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $\quad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $\quad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $\quad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

Home exercise #2: language model