Context similarity in huge corpora

PA154 Language Modeling (8.2)

Pavel Rychlý

pary@fi.muni.cz

April 16, 2024

Meanings

The word (and some of its parts) are the basic carriers of meaning

- word without context – no meaning, many meaning potentials
- the same word in different contexts – different meanings
- word in similar contexts – same meaning
- what is context?

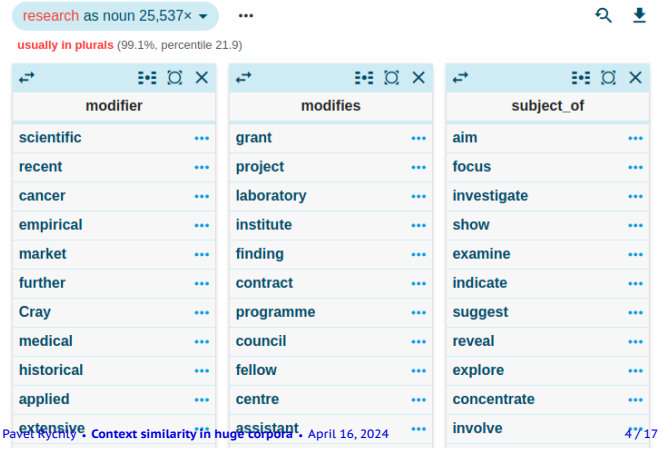
What is context?

Context is the words around the keyword.

- What surroundings?:
  - the following word
  - previous word
  - window: +1 to +5
  - window: -5 to -1
- Not all words around are important.
- How do we determine importance?
  - the most common collocation – but that's "the"
  - (statistically) most significant – what formula?

Word Sketch

One-page summary of word behaviour try online



Word Sketch

How to create it

- Large balanced corpus
- Find grammatical realtions (subjects, objects, heads, modifiers etc)
- List of collocations for each grammatical session
- Statistics for sorting each list

We can create a thesaurus from Word Sketch.

Grammatical Relations Definition

- plain text file
- a set of queries for each GR
- queries contain labels for keyword and collocate
- processing options

Grammar relation definitions

```
# ‘modifier’ and ‘modify’ gramrels definition
*DUAL
=modifier/modify
   2:"AJ." 1:"N.."

# ‘and/or’ gramrel definition
=and/or
*SYMMETRIC
   1:[] [word="and"|word="or"] 2:[] & 1.tag = 2.tag

# ‘adverb’ gramrel definition
=adverb
   1:[] 2:"AV."
   2:"AV." 1:[]
```


Association score

- number of occurrences $(word_1, gramrel, word_2)$
- $AScore(w_1, R, w_2) = 14 + \log_2 Dice\left(\frac{||w_1,R,w_2||}{||w_1,R,*||}, \frac{||w_1,R,w_2||}{||*,*,w_2||}\right) = 14 + \log_2 \frac{2 \cdot ||w_1,R,w_2||}{||w_1,R,*||+||*,*,w_2||}$

Similarity coefficient

- comparison of word sketches w_1 and w_2
- only important (significant) contexts
- what is the common
- counts $(word_1, (gramrel, word_i))$ and $(word_2, (gramrel, word_i))$

$$Sim(w_1, w_2) = \frac{\sum_{(tup_i, tup_j) \in \{tup_{w_1} \cap tups_{w_2}\}} AS_i + AS_j - (AS_i - AS_j)^2/50}{\sum_{tup_i \in \{tup_{w_1} \cup tup_{w_2}\}} AS_i}$$

Data Sizes

Corpus sizes, their vocabularies and word counts in contexts

Corpus	Size	Words	Lemat	Different ctx	All ctx
BNC	111m	776k	722k	23m	63m
SYN2000	114m	1.65m	776k	19m	58m
OEC	1.12g	3.67m	3.12m	84m	569m
Itwac	1.92g	6.32m	4.76m	67m	587m

Vocabulary sizes and the number of different contexts grow sublinearly with the size of the corpus.

Matrix size

- Similarity of all pairs of lemmas
- Matrix of size N^2, where N is 700k – 5m
- Number of elements in orders of tera (10^{12})
- Matrix is fortunately very sparse
- Most values are 0 or “almost” 0
- Even most of the whole rows/columns are empty

Practical data sizes

- Computation only for words with minimum frequency
- Better to limit the number of contexts than the number of occurrences
- Take only statistically significant contexts

Corpus	MIN	Lemmat	KWIC	CTX
BNC	1	152k	5.7m	608k
BNC	20	68k	5.6m	588k
OEC	2	269k	27.5m	994k
OEC	20	128k	27.3m	981k
OEC	200	48k	26.7m	965k
Itwac	20	137k	24.8m	1.1m

Practical data sizes

- Matrix of size N^2, where N is 50k – 200k
- Number of elements in orders of giga (10^{10})
- The value of each element is created by applying the similarity function to vectors of length K = 500k – 1m.
- The straightforward algorithm for computing the whole matrix has a time complexity $O(N^2K)$.
- The complexity is polynomial, but the algorithm is practically unusable for given ranges of values.
- Estimated calculation times are in months or years.
- Heuristics reduce the sizes of N and K at the expense of accuracy the resulting values.
- The calculation time is then in the order of days with an error of 1–4%.

Efficient algorithm

- Even the smaller matrix is very sparse
- No need to calculate similarity for words that have nothing together,
- they have no common context.
- The main loop of the algorithm is not through words, but through contexts.

Efficient algorithm

- Input: list of all possible words in contexts, $\langle w, r, w' \rangle$, with frequencies of occurrences in the corpus
- Output: word similarity matrix $sim(w_1, w_2)$

for $\langle r, w' \rangle$ **in** CONTEXTS:
  WLIST = set of all w where $\langle w, r, w' \rangle$ exists
  for w_1 **in** WLIST:
    for w_2 **in** WLIST:
      $sim(w_1, w_2) += f(frequencies)$

Optimization

- If $|WLIST| > 10000$, skip the context.
- We do not keep the matrix $sim(w_1, w_2)$ in memory during the calculation.
- Repeated runs of the main loop for the limited range w_1.
- Instead of $sim(w_1, w_2) += x$ we generate $\langle w_1, w_2, x \rangle$ to the output.
- We then sort the output list and add the individual xs.
- Use of TPMMS (Two Phase Multi-way Merge Sort) with continuous by summation.
- Instead of several hundred GB, we sort a few GB.

Results

- Algorithm is orders of magnitude faster than straightforward algorithm. (18 days × 2 hours)

Corpus	MIN	Lemmat	KWIC	CTX	Time
BNC	1	152k	5.7m	608k	13m 9s
BNC	20	68k	5.6m	588k	9m 30s
OEC	2	269k	27.5m	994k	1h 40m
OEC	20	128k	27.3m	981k	1h 27m
OEC	200	48k	26.7m	965k	1h 10m
Itwac	20	137k	24.8m	1.1m	1h 16m

- Without changes in precision
- Possibilities of easy parallelization.