Annotated Suffix Trees for NLP

Ekaterina Chernyak Dmitry Ilvovsky

Data Analysis and Artificial Intelligence Department
National Research University Higher School of Economics
Moscow, Russia

January 30, 2017

We are going to talk about two NLP tasks:

1. Text clustering
2. Profanity filtering

Text clustering

- Distance-based algorithms
  - Feature extraction $\Longrightarrow$ Each text is a vector of features $\Longrightarrow$ We know how to estimate distance between vectors
  - k-Means algorithm
- **Similarity-based algorithms**
  - We know how to estimate similarity between texts $\Longrightarrow$ Similarity matrix
  - Suffix tree clustering, k-Medoids, Normalized cuts, hierarchical algorithms
- Same measures can be used to compute both similarity and distance, since $\texttt{similarity} = \frac{1}{\texttt{distance}}$, but the formal definitions of a cluster and optimizing criteria are different

Text similarity

- Character-, **string**- and term-based similarity measures
  - $tf \times idf$ transformation and the cosine similarity

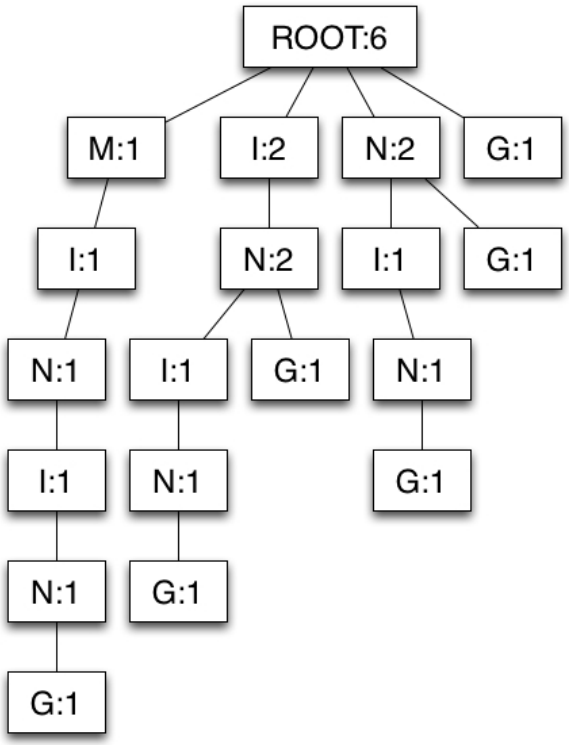
$$tf \times idf(t, d, D) = f(t, d) \times \frac{|D|}{\log |d \in D : t \in d| + 1}$$

$$\cos(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\| \|d_2\|}$$

- Corpus-based similarity measures
- Knowledge-based similarity measures

Annotated suffix tree

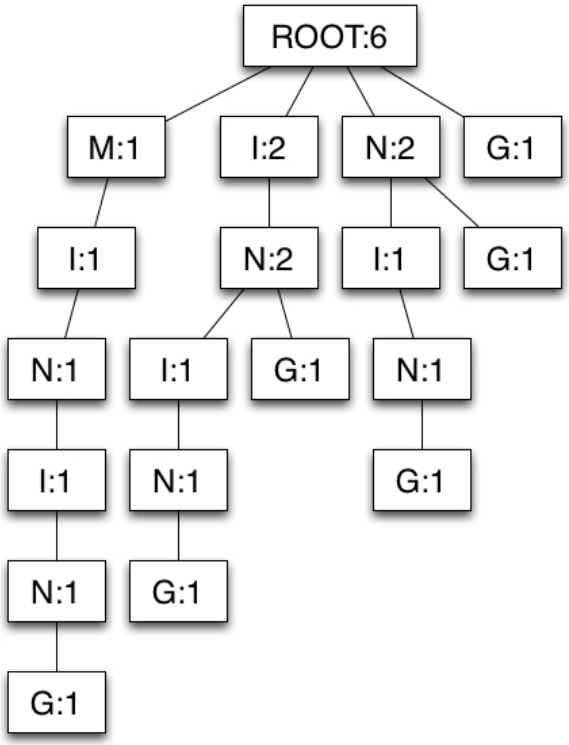
- Is used to store all fragments of the input text and their frequencies
- Is constructed in linear time and space
- Can be constructed for a text (split the text into word *n*-grams)
- Does not require complex preprocessing!



Annotated suffix tree properties

The frequency of a parent node is equal to:

1. the sum of the frequencies of the children nodes
2. the sum of the frequencies of the underlying leaves



Scoring procedure

To score a string to an AST:

- Split the string into suffixes
- Find a match of every suffix
- Score the match:

$$score(match(suffix, ast)) = \sum_{node \in match} \phi(\frac{f(node)}{f(parent(node))})$$

- Sum the scores and average over all the matches:

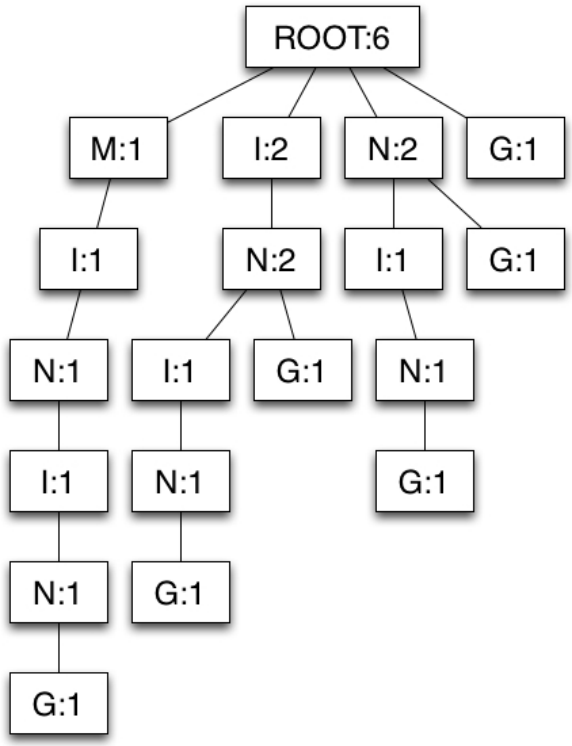
$$relevance(string, ast) = SCORE(string, ast)$$

$$SCORE(string, ast) = \frac{\sum_{suffix} score(match(suffix, ast))/|suffix|}{|string|}$$

Scoring procedure. Example

Let us score the string “dine”

- “dine” $\Longrightarrow$ no match
- “ine” $\Longrightarrow$ “I N”
  $\Longrightarrow 2/2 + 2/6 = 1.33$
- “ne” $\Longrightarrow$ “N”
  $\Longrightarrow 2/6 = 0.33$
- “e” $\Longrightarrow$ no match



$$\texttt{score}(\text{“}\textit{dine}\text{”}, \textit{ast}) = \frac{0 + 1.33/2 + 0.33/1 + 0}{4} = 0.25$$

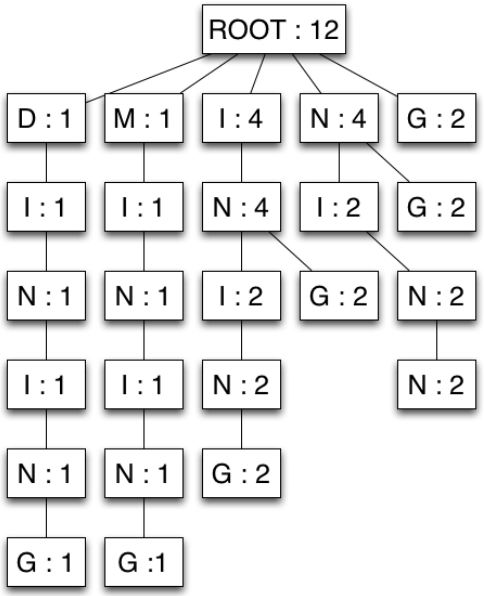
AST-based similarity measure (1)

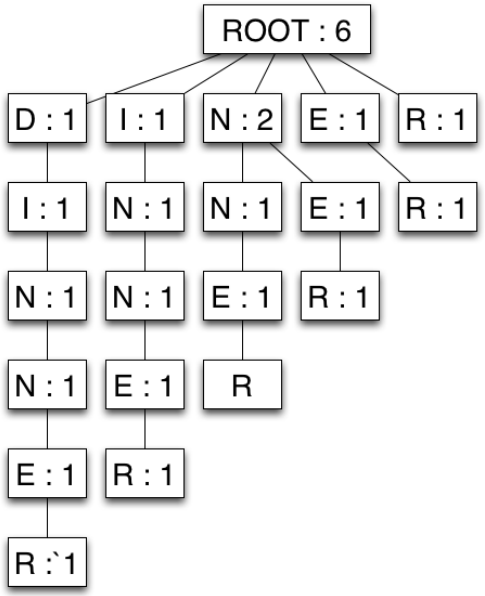
Analogous to *tf*

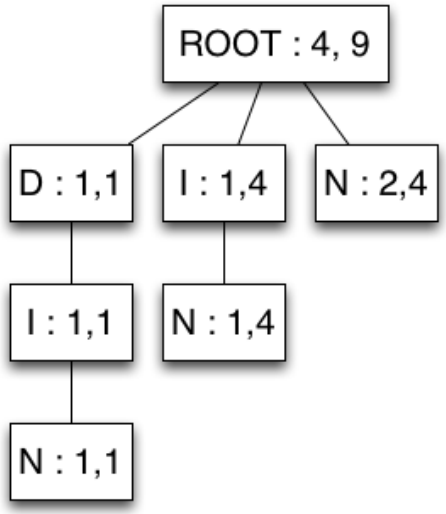
Text similarity measure is based on scoring common subtree of two ASTs:

- depth-first search for the common chains of nodes that start from the root of the both ASTs
- a new node of a common subtree is annotated by the minimum and the maximum frequencies
- weighting each node by computing the mean between two frequencies
- scoring every chain of the common subtree
- summing up all chain scores and standardizing them by dividing by the number of chains

Common subtree







(1)

AST-based similarity measure (2)

Analogous to *idf*

- A global AST for the whole text collection
- Let V be a set of unique n-grams for the text collection
- Construct AST from V

Final chain scoring

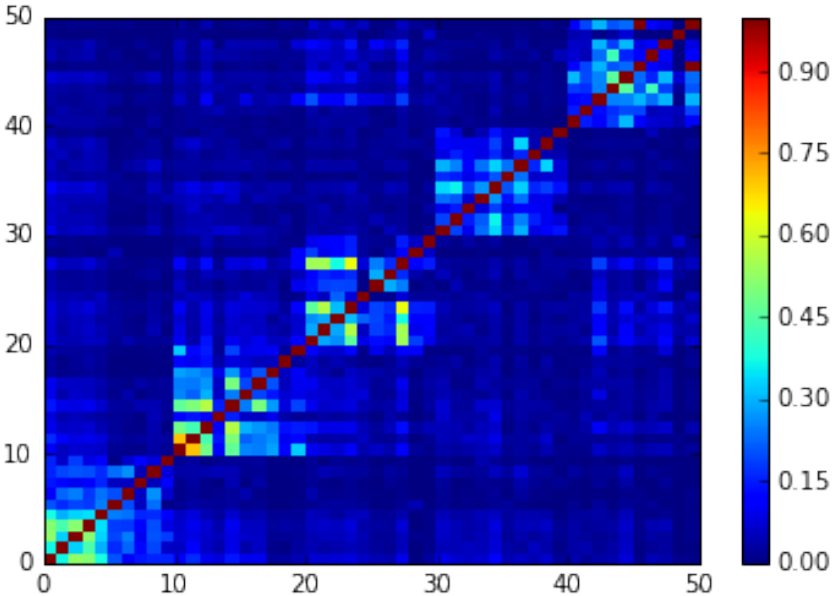
$$\texttt{score}(chain) = \frac{\sum_{node \in chain} \frac{f^{node}}{f^{parent}} \times \frac{df^{node}}{df^{parent}}}{|chain|}$$

$$\texttt{SCORE}(subtree) = \frac{\sum_{chain \in subtree} \texttt{score}(chain)/|chain|}{|string|}$$

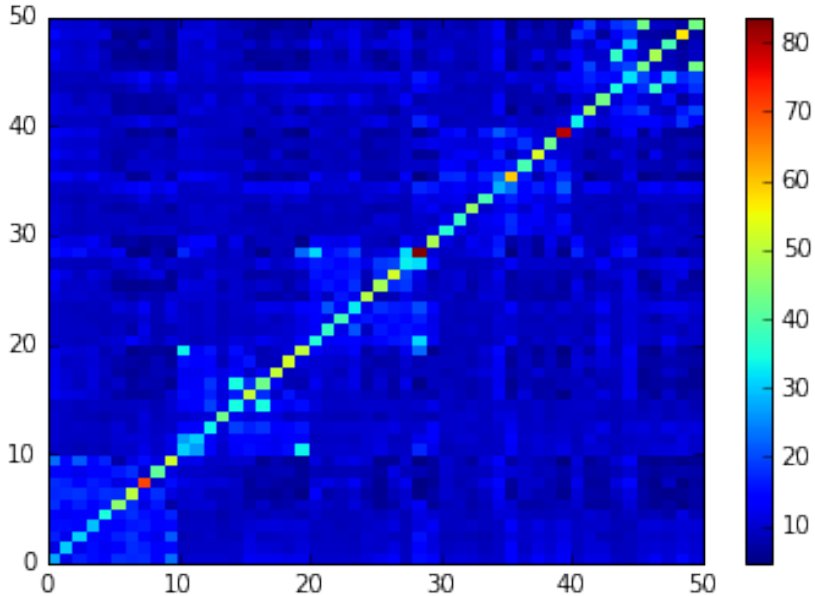
Experiments

- 50 documents in Russian
- Every 10 text devoted to different definitions of the word “jaguar”: an animal, a car, a beverage, a film or a sewing machine
- 2 similarity measures
  1. the $tf \times idf$ transformation and the cosine similarity
  2. the AST technique, presented above
- Normalized cuts with default parameters to find 5 clusters
- Quality measured by accuracy and number of cluster errors

Heat maps of similarity matrices



Heat map of the cosine similarity matrix



Heat map of the AST similarity matrix

Clustering quality

	Accuracy	# of errors
cosine similarity	0.88	6
AST similarity	0.96	2

Conclusion and future work

- New text similarity measure proposed
- Clear advantages in comparison to the baseline cosine similarity, because of being fuzzy and thus more robust
- We need to improve the formal definition, because the AST-based similarity of a text to itself is not equal to unity
- ... and conduct more experiments

Profanity filtering

- Given a list of stop-words, find all of these words in a text
- The task is close to IR task, but instead of hight precision we are more interested in high recall.
- Similarity measures:
  - looking for stems or lemmata;
  - Jaccard coefficient of symbolic *n*-grams;
  - Edit distance;
  - AST-based similarity measure: AST-score is higher than a given threshold;

Data collection

- A list of words, prohibited for naming in RF url zone (4023 words);
- A text collection, which consists of research papers on etymology of russian mat, texts of Yu. Aleshkovsky, I. Guberman, V. Sorokin, lyrics by Leningrad and Krasnaya Plesen', poems by S. Esenin, V. Mayakovsky, A. Pushkin, posts by A. Lebedev, random texts from Lurkmore, social media and etc.;
- 294916 tokens, 60868 types;
- Evaluated by F_2-measure,precision, recall, accuracy.

Results

	Pr	R	acc	F_2
	0.7288	0.1363	0.9810	0.2297
lemmatization				
pymorphy2	0.6492	0.2466	0.9815	0.3574
mystem3	0.6807	0.3195	**0.9827**	0.4349
stemming	0.6113	0.4028	0.9822	**0.4856**
, $= 0.2$	0.1578	**0.6201**	0.9233	0.2516
Jaccard coefficient				
3-grams	0.6799	0.1633	0.9810	0.2634
4-grams	0.7126	0.1475	0.9810	0.2430
5-grams	0.7168	0.1284	0.9808	0.2179
6-grams	0.6989	0.0975	0.9803	0.1711
Edit distance				
$d < 8$	0.0234	**0.9127**	0.8086	0.0456
$d < 5$	0.0209	**0.9825**	0.9629	0.0409

Conclusion and future work

- An efficient profanity filter proposed
- Clear advantages in terms of recall
- New data source created
- How to deal with multiword expressions?
- Applications for sentiment analysis and opinion mining.