Adapting the graph2vec approach to dependency trees for NLP tasks

Oleg Durandin, Alexey Malafeev

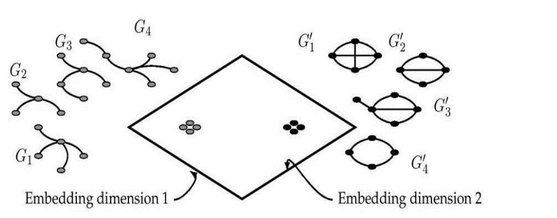
Higher School of Economics, Nizhny Novgorod

Introduction

The graph2vec approach applied to dependency trees is unsatisfactory, which is due to the WL Kernel. We adapt this kernel for dependency trees and also propose 3 other types of kernels that take into account the specific features of dependency trees. This new vector representation can be used in NLP tasks where it is important to model syntax (e.g. authorship attribution, intention labeling, targeted sentiment analysis etc.). UD treebanks[5] were clustered to show the consistency and validity of the proposed tree representation methods.

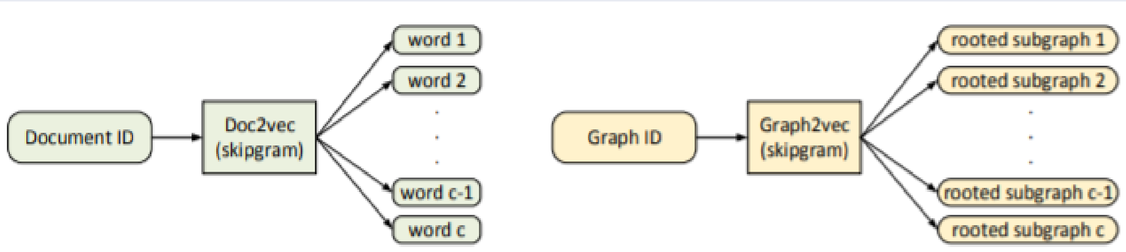
Graph2vec approach

The main idea of the **graph2vec** approach [1] is to view the entire graph as a document and the rooted subgraphs (that encompass a neighborhood of certain degree) around every node as words, that compose the document. In other words, different subgraphs compose graphs in a similar way that different words compose sentences/documents when used together.

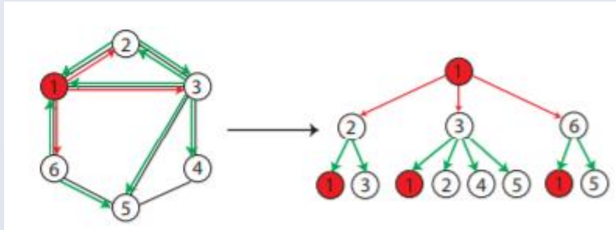


Graph embedding algorithms create low dimensional representations of whole graphs in an embedding space. Those graphs that have similar structural properties (labeled random walks, shortest paths, subtree patterns, graphlets) are located close in the embedding space. In this example G_1, G_2, G_3 and G_4 are all tree-like and because of this they are clustered together in the embedding space. Similarly, G'_1, G'_2, G'_3 and G'_4 are close to each other in the embedding space as the graphs themselves all have loops.

doc2vec model [2] - Given a document d, it samples c words from d and considers them as co-occurring in the same context (i.e., context of the document d) uses them to learn d's representation. **Graph2vec [1]** - Given a graph G, it samples c rooted subgraphs around different nodes that occur in G and uses them analogous to doc2vec's context words and thus learns representation of G.



In graph2vec [1] WL Kernel [3] was used as a method of proceed root subgraphs. WL-kernel is based on the idea of extracting subtree patterns of fixed length. Also, it should be noted that used only relabeling method of WL kernel.



Using graph2vec on dependency trees

The dependency tree is a representation of the syntactic structure of a given sentence. The dependency tree is a directed acyclic graph with words as nodes and relations as edges. Each word in the sentence either modifies another word or is modified by a word.

Properties of the tree: **connected**, **acyclic** and **oriented** graph, where **each node has a label** (the word itself or its POS-tag). We will also consider the **edge labels** – the type of **dependency** between the connected head and dependent word.

Limitation of applying the WL kernel to dependency trees:

1. Assumption about the uniqueness of the labels of the nodes and edges of the graph (the same POS tags can correspond to different words in dependency trees).

2. The WL kernel does not take into account the use of edge labels, which represent, in our case, the type of relationship between a pair of words in a sentence.

3. The WL kernel is used on graphs in which the order of the nodes is not important, which is not usually the case with dependency trees.

We propose 3 kernels to process dependency trees:

1. The relaxed WL kernel – take into account the edge labels, lose the uniqueness assumption.
2. The contracted kernel – in relabeling procedure, we collapse pairs of nodes into phrases (ADJ + NOUN becomes ADJ_NOUN).
3. The simple path kernel – we collect all paths between each pairs of node in dependency tree. We take into account all dependency types as well as PoS-tags.

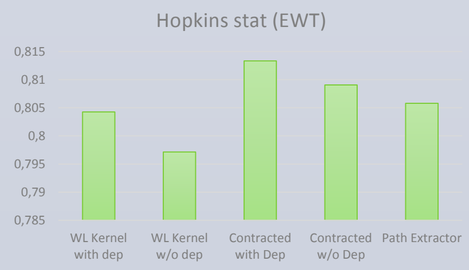
Tree banks

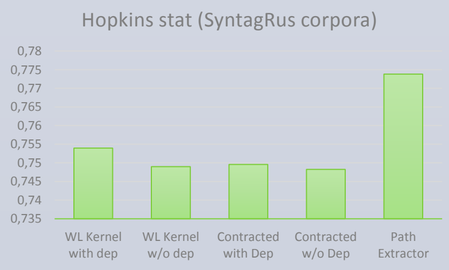
We use the following tree banks from the Universal Dependencies project [4]:

1. English Dependency Treebank UD English Web Treebank (>16 thousand sentences) [5]
2. Russian data from the SynTagRus corpus (>66 thousand sentences). [6]

Experiments

After processing the treebanks with proposed kernels, we obtain a set of embeddings for Russian and English sentences. Analysis of clustering structure with Hopkins statistics [9] reveals that the most suitable kernels are different for the two languages.





Experiments

We conduct cluster analysis for all set of embeddings. Embeddings were normalized, Kmeans++ [7] was used as the clustering algorithm, with silhouette analysis [8] for choosing the optimal number of clusters. After clustering, cluster centroids were chosen, and 20 embeddings were selected and manually analyzed.

- Different kernels capture different structural types of sentences.
- A larger number of clusters results in a more detailed syntax model in comparison with a smaller number.

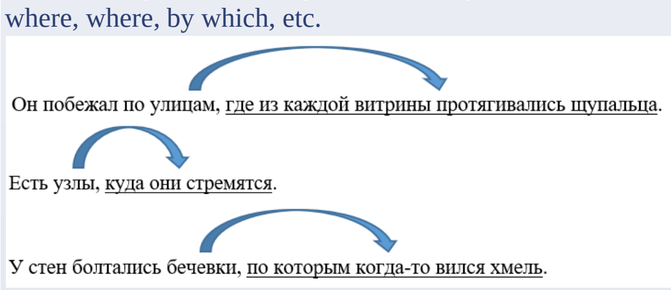
We'll consider only one clustering due to space restrictions.

Relabeling method	No. clusters	
	SynTagRus	EWT
Relaxed WL Kernel (with dep)	14	10
Relaxed WL Kernel (without dep)	10	19
Contracted (with dep)	10	29
Contracted (without dep)	22	18
Simple Path	20	10/14

Simple Path Kernel (SyntagRus)

Cluster#0: Complex sentences with relative clauses.

It is clearly seen that these sentences have similar structure (there is a subordinate clause), and the clause can be introduced by different conjunctions / conjunction words: where, where, by which, etc.

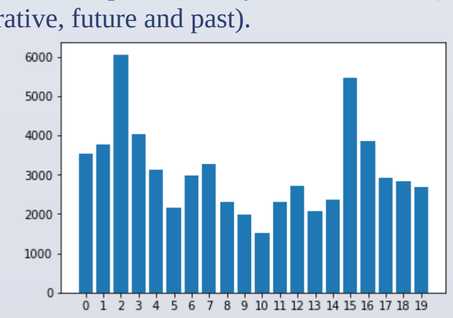


Cluster#17: Sentences with multiple verbs or participles, often in the same form.

- «Верь, повинуйся, борись!» (verbs in the imperative form)
- «Отбившиеся, прорвавшиеся поодиночке.» (past participles in the plural nominative form)
- «Здесь живут, здесь лечат и учат.» (verbs in the present plural form)

These sentences always contain 2-3 verbs or participles, usually in the same form, but not always:

- «Подожди, я дров подложу, дом настыл.» (verbs: imperative, future and past).



Full description of clusters

1. Complex sentences with relative clauses;
2. Simple sentences with one indirect object;
3. Elliptical (incomplete) sentences without verbs;
4. Elliptical (incomplete) sentences with verbs;
5. Simple sentences with an infinitive and one or more adverb;
6. Complex sentences with adverbial clauses and at least one numeral;
7. Incomplete sentences with proper nouns;
8. Complex sentences that start with a verb and have a noun clause as the object of this verb (importantly, the clause includes a verb and no adjectives, unlike Cluster 11);
9. Sentences containing at least one adjective that is a direct dependent of a verb (as in become silent);
10. Sentences with homogeneous objects (direct or indirect), expressed by nouns;
11. Complex sentences that start with a verb and have a noun clause as the object of this verb (the clause includes an adjective and no verbs, cf. Cluster 8);
12. Complex sentences with noun clauses that have an infinitive;
13. Sentences containing at least one adjective that is a direct dependent of a noun (as in electronic microscope);
14. Sentences with homogeneous subjects, expressed by nouns;
15. Sentences with a noun dependent on another noun (as in prichiny padeniya, which is equivalent to the reasons for the fall);
16. Short simple sentences with an auxiliary verb (e.g. byl = was) and an adjective that is its dependent, as well as sentences where the auxiliary verb is omitted (but is assumed);
17. Sentences with homogeneous predicates, expressed by verbs;
18. Sentences with a participle clause;
19. Sentences with adverbial and adverbial participle clauses;
20. Sentences with multiple pronouns.

Conclusions

We show the inefficiency of the WL kernel for processing dependency trees and propose an adaptation of the graph2vec models for their processing. Three kernels that take into account linguistic peculiarity are proposed. In our work, we construct vector spaces for the SynTagRus and the EWT treebanks and cluster this space.

From the linguistic viewpoint, the experiments suggest that different clusterings capture different structural types of sentences. The proposed methods completely isolate the morpho-syntax layer of the language system, considering it separately from the lexis and semantic layers.

Future work:

- Identify common patterns of vector space with Graph Mining tools.
- Reveal patterns for different languages;
- Using syntactic embeddings as a part of supervised learning tasks (MT, NLU problems) to prove the efficiency of such embeddings.

References

[1] A. Narayanan, et al. "graph2vec: Learning Distributed Representation of Graphs"

[2] Quoc Le and Tomas Mikolov (2014) "Distributed Representations of Sentences and Documents"

[3] N. Shervashidze et al. (2011) «Weisfeiler-Lehman Graph Kernels»

[4] De Marneffe et. al.(2014) «Universal Stanford Dependencies: A cross-linguistic typology» https://universaldependencies.org/

[5] Natalia Silveira et. al (2014) «A Gold Standard Dependency Corpus for English», https://github.com/UniversalDependencies/UD_English-EWT

[6] Droganova, K., Lyashevskaya, O., & Zeman, D. (2018). Data Conversion and Consistency of Monolingual Corpora: Russian UD Treebanks. https://github.com/UniversalDependencies/UD_Russian-SynTagRus

[7] Arthur, David, and Sergei Vassilvitskii (2007), "k-means++: The advantages of careful seeding"

[8] Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis".

[9] Richard G. Lawson, Peter C. Jurs (1990) "New index for clustering tendency and its application to chemical problems"

CONTACTS

Source code: https://github.com/OlegDurandin/dtree2vec

Oleg Durandin: oleg.durandin@gmail.com

Alexey Malafeev: amalafeev@yandex.ru