НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

Graphs in NLP

Heterogeneous Information Fusion

Vadim Alperovich

LATNA, HSE NN

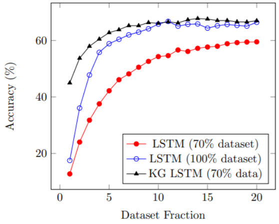
Nizhny Novgorod, 2020

Motivation

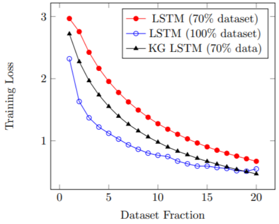
1. ACL 2019 and ACL 2020
   - There is a growing trend for using network structures
2. Intuition
   - It seems natural use in natural language not only syntax and semantics but some external common knowledge
   - In many NLP problems entities are connected by a range of relations, not only co-occurrence
3. Previous results and existent gaps
   - semi-supervized learning; less amount of labeled training data,when it has access to organized world knowledge

Motivation

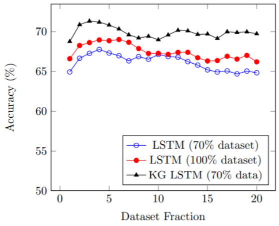
- Knowledge Graphs (KG) + LSTM [20]



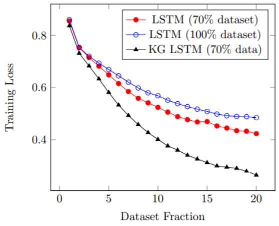
(a)



(b)



(c)



(d)

Objectives

1. Review of graph-based and network representations of natural language
2. Research of widespread approaches to transform graph structures to vector representations
3. Breakdown of some methods of incorporating graph-based information into embedding-based NLP model

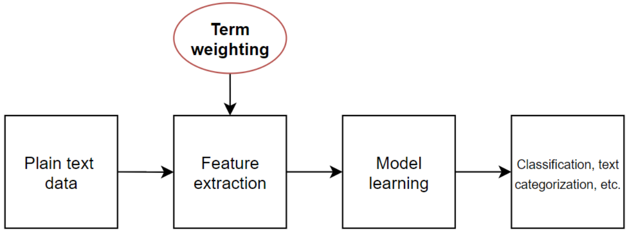
Content

Bag-of-Words issues

Graph-based Natural Language Representations

- Text is a set of terms (unique tokens) and its frequencies
- Term independence assumption

  BoW model representation doesn't consider the semantic relations between words
- Term frequency weighting



Graph-based document representation

Graph-based Natural Language Representations

- Graph construction [1]

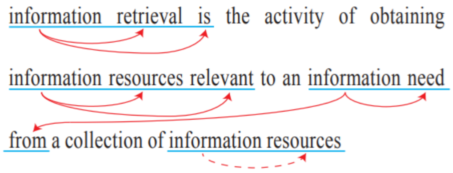
  each document $d \in D$ is represented by a graph G_d

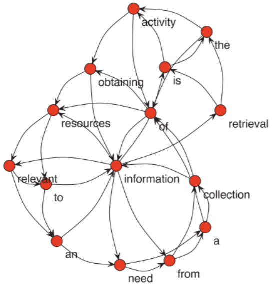
$$G_d = (V_d, E_d)$$

where V_d - the set of nodes represented by terms in d, E_d - the set of edges that capture co-occurrence relationships between terms within a fixed-size sliding window of size ω in d.

Graph-based document representation
Graph-based Natural Language Representations

- Idea: Replace term frequency with node centrality [2]


information retrieval is the activity of obtaining
information resources relevant to an information need
from a collection of information resources



- Deals with the *term independence and term frequency weighting assumptions*
- Taking into account **word dependence, order and distance**

Graph-based document representation

Graph-based Natural Language Representations

- InfraNodus: Generating Insight Using Text Network Analysis (2019) [3]

  Text analysis pipeline:

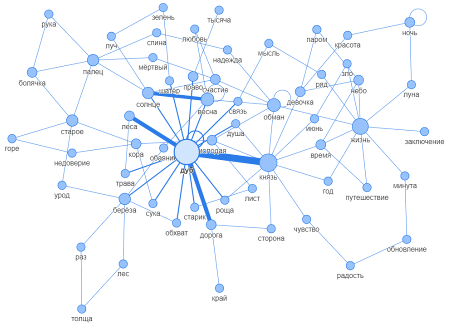
  1. Text Normalization
  2. Stop words removal
  3. Text-to-Network Conversion
  4. Extracting Most Influential Keywords Using Betweenness Centrality
  5. Topic Modelling Using Community Detection
  6. Summarization
  7. Discourse Structure and the Measure of Discourse Bias
  8. Insight Generation using Structural Gaps

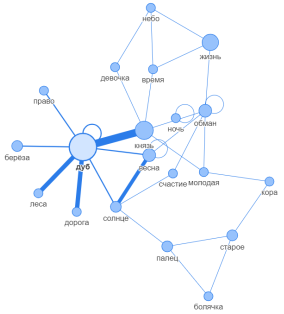
- https://infranodus.com/infranodus/ihaveadream

Graph-based document representation

Graph-based Natural Language Representations

- My implementation for Russian text (steps 1-3)
  Л.Н.Толстой "Война и мир" Встреча Болконского с дубом

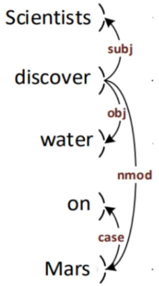


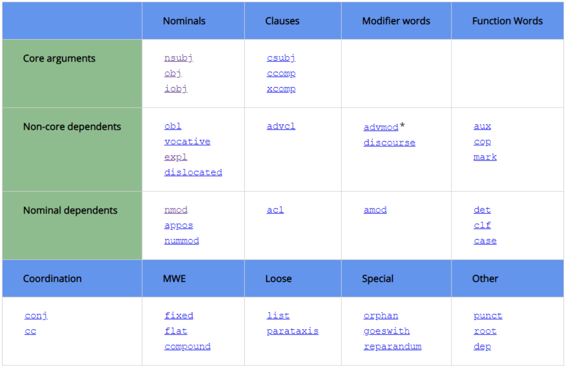


Syntax-based graph

Graph-based Natural Language Representations

- https://universaldependencies.org/u/dep/all.html



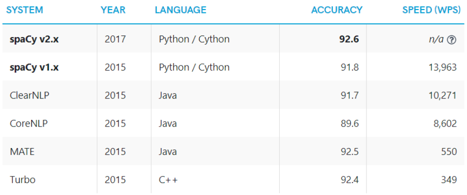


	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

Syntax-based graph

Graph-based Natural Language Representations

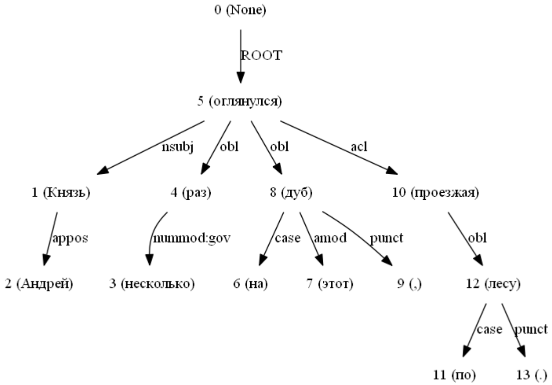
- Stanford Dependency [18]
- Universal Stanford Dependencies: A cross-linguistic typology [4]
- The syntactic context helps to capture functional similarity rather than topical similarity of texts
- Syntactic parsers comparison https://spacy.io/usage/facts-figures



SYSTEM	YEAR	LANGUAGE	ACCURACY	SPEED (WPS)
spaCy v2.x	2017	Python / Cython	**92.6**	*n/a*
spaCy v1.x	2015	Python / Cython	91.8	13,963
ClearNLP	2015	Java	91.7	10,271
CoreNLP	2015	Java	89.6	8,602
MATE	2015	Java	92.5	550
Turbo	2015	C++	92.4	349

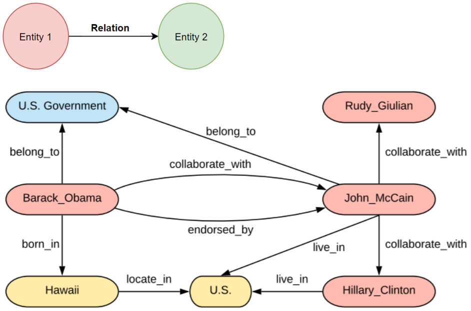
Syntax-based graph
Graph-based Natural Language Representations

- My implementation via StanfordCoreNLP [18]
  - Russian Tagging and Dependency Parsing Models for Stanford CoreNLP Natural Language Toolkit, ITMO University
- *Князь Андрей несколько раз оглянулся на этот дуб, проезжая по лесу.*



Knowledge graphs
Graph-based Natural Language Representations

- Knowledge graph contains of *fact triples*
- Fact triples - <entity, relation, entity> ,
  e.g. <London, is_capital, Britain>, <Turing, born_in , 1912>, etc.



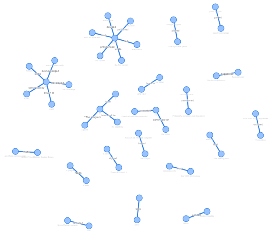
(c) A small fraction of a large knowledge graph.

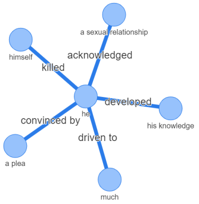
Tackling Graphical NLP problems with Graph Recurrent Networks, 2019 [4]

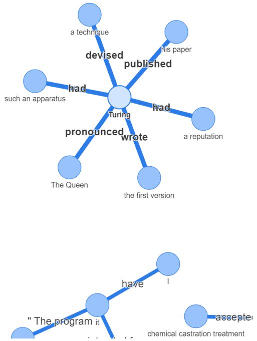
Knowledge graphs

Graph-based Natural Language Representations

- Implementation with SpaCy framework
- https://en.wikipedia.org/wiki/Alan_Turing



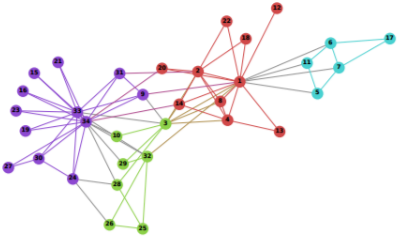




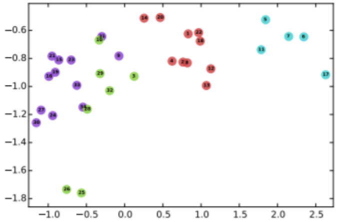
**co-reference problem*

Vector Representations of Graph Structures

- *How to find some effective numerical representation of nodes, communities and whole network...* [6]



(a) Input: Karate Graph



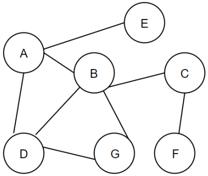
(b) Output: Representation

Random Walk and Node2vec

Vector Representations of Graph Structures

- Given a graph and a starting node, we select a neighbor of it at random, and move to this neighbor; then we select a neighbor of this node at random, and move to it etc.
- The sequence of nodes selected this way is a *random walk* on a graph.

$$rw_A^{l=5} = [A, B, D, G, B]$$
$$rw_B^{l=5} = [B, C, F, C, B]$$

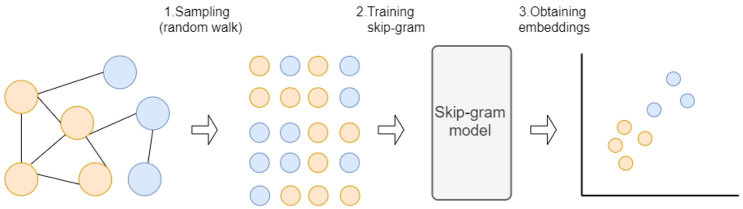


- Capturing some local information of a node

Random Walk and Node2vec

Vector Representations of Graph Structures

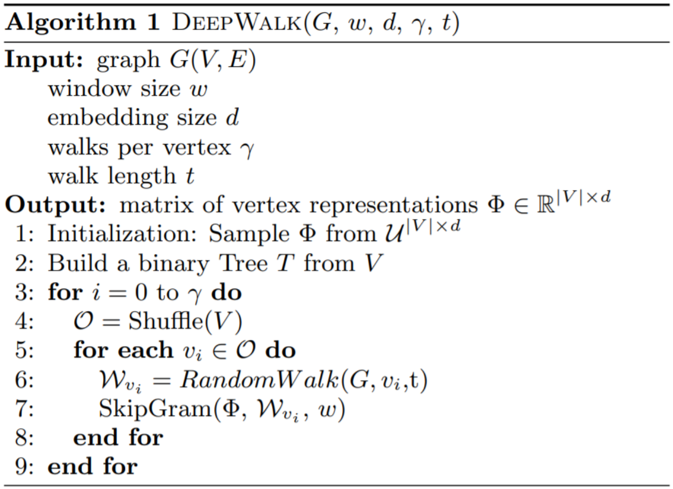
- Graph $G = (V, E)$
  $rw_v = \{u^1, u^2, \dots, u^N\}$ - random walk on G,
  where $v \in V, u^i \in \mathcal{N}(v)$, $N(x)$ - a set of neighbours of node x
- *Assume that in u^i is a token and rw_v for $v \in V$ is a set of sentences*, e. g. we can use Word2Vec [5]
- Yeah, we have DeepWalk model (Perozzi et al., 2014) [6] [7]



Random Walk and Node2vec

Vector Representations of Graph Structures

- DeepWalk [6]



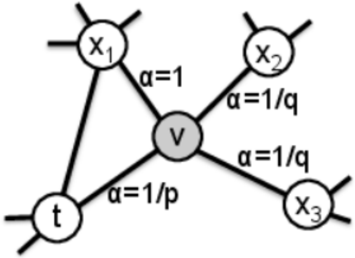
Algorithm 1 DEEPWALK(G, w, d, γ, t)

```
Input: graph $G(V, E)$
    window size $w$
    embedding size $d$
    walks per vertex $\gamma$
    walk length $t$
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$
 1: Initialization: Sample $\Phi$ from $\mathcal{U}^{|V| \times d}$
 2: Build a binary Tree $T$ from $V$
 3: for $i = 0$ to $\gamma$ do
 4:     $\mathcal{O} = \text{Shuffle}(V)$
 5:     for each $v_i \in \mathcal{O}$ do
 6:         $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
 7:         SkipGram($\Phi$, $\mathcal{W}_{v_i}$, $w$)
 8:     end for
 9: end for
```

Random Walk and Node2vec

Vector Representations of Graph Structures

- Node2Vec [8]
- Node2vec is a modification of DeepWalk with the small difference in random walks. It has parameters P and Q.
- Parameter Q defines how probable is that the random walk would discover the undiscovered part of the graph, while parameter P defines how probable is that the random walk would return to the previous node.
- node2vec: Scalable Feature Learning for Networks



Random Walk and Node2vec
Vector Representations of Graph Structures

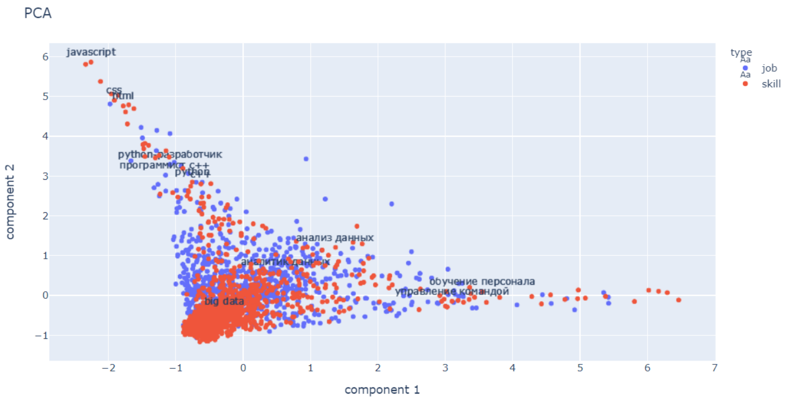
- A bipartite graph jobs to skills from hh.ru
- 'Программист', 'Разработчик', 'Аналитик', 'Data analyst', 'Data Scientist', 'IT-специалист'

-
  ```
  random_walk(net,'python')
  Out:['python',
   'программист-разработчик phyton',
   'ооп',
   'ведущий web-разработчик',
   'asp.net',
   'full stack разработчик',
   'asp.net',
   'программист c#',
   'css',
   'разработчик vba',
   'html']
  ```

Random Walk and Node2vec

Vector Representations of Graph Structures

- A bipartite graph jobs to skills from hh.ru with gensim.models.Word2Vec



Random Walk and Node2vec

Vector Representations of Graph Structures

- A bipartite graph jobs to skills from hh.ru

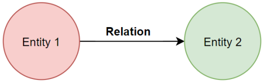
-
  ```
  embeddings.most_similar('c++', topn=5)
  Out:
  [('qt/qml developer', 0.997),
  ('computer vision intern (open model zoo)', 0.993),
  ('инженер-программист', 0.988),
  ('deep learning software engineer', 0.987),
  ('c/c++', 0.986)]
  ```

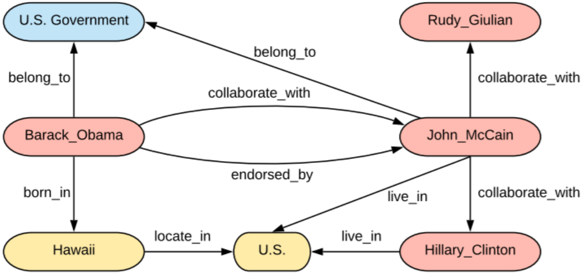
-
  ```
  embeddings.most_similar(
      positive=['c++', 'python-разработчик'],
      negative=['python'], topn=3)
  Out:[('программист c++', 0.99),
       ('разработчик vba', 0.989),
       ('программист c#/unity (руководитель группы)', 0.987)]
  ```

Knowledge Graph Embeddings

Vector Representations of Graph Structures

- KG - a is a directed graph which relation types have domain-specific semantics
- KGs consist of *fact triples*, e.g. <head, relation, tail>





(c) A small fraction of a large knowledge graph.

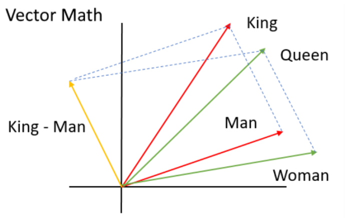
Knowledge Graph Embeddings
Vector Representations of Graph Structures

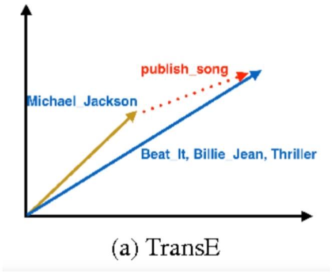
- Idea: an embedding of the head entity + some vector of the relation is close to an embedding of the tail entity
- Translation based embedding model or **TransE** [9]
- $d(h+r,t)$ - L_1 or L_2 norm
- $L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} [\gamma + d(h+r,t) - d(h'+r,t')]_+ \to min$
  where $[x]_+$ denotes the positive part of x, $\gamma > 0$ is a margin hyperparameter
  $S' = \{(h',r,t) : h' \in E\} \cup \{(h,r,t') : t' \in E\}$ - the set of corrupted triples, i.e. triplets with either the head or tail replaced by a random entity **but not at the same time**

Knowledge Graph Embeddings

Vector Representations of Graph Structures

- Translation based embedding model or **TransE** [9]





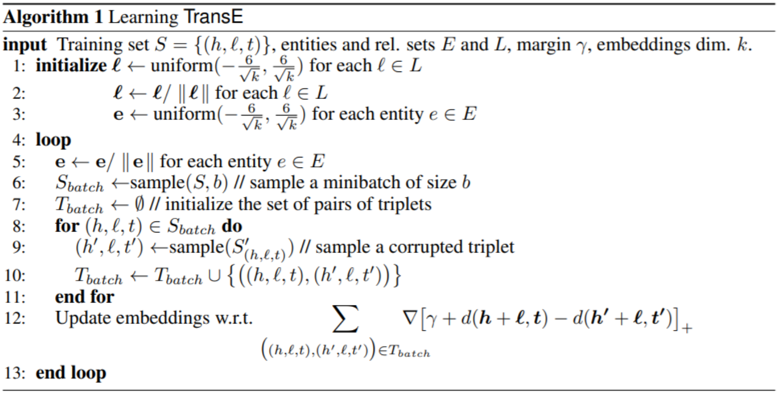
(a) TransE

- Sources: Text analysis: fundamentals and sentiment analysis, and Knowledge Graph Embedding by Flexible Translation, 2016 [10]

Knowledge Graph Embeddings

Vector Representations of Graph Structures

- Translation based embedding model or **TransE** [9]

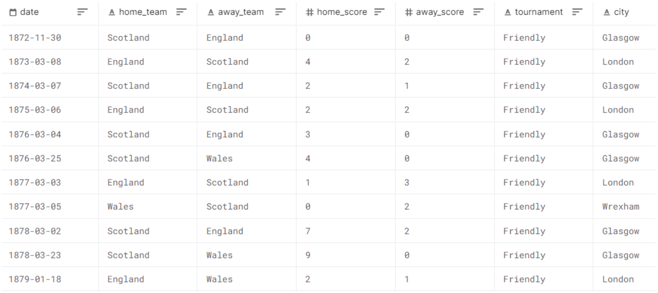


Algorithm 1 Learning TransE

```
input Training set $S = \{(h, \ell, t)\}$, entities and rel. sets $E$ and $L$, margin $\gamma$, embeddings dim. $k$.
 1: initialize $\boldsymbol{\ell} \leftarrow$ uniform($-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}$) for each $\ell \in L$
 2:            $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$
 3:            $\mathbf{e} \leftarrow$ uniform($-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}$) for each entity $e \in E$
 4: loop
 5:    $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$
 6:    $S_{batch} \leftarrow$sample$(S, b)$ // sample a minibatch of size $b$
 7:    $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets
 8:    for $(h, \ell, t) \in S_{batch}$ do
 9:       $(h', \ell, t') \leftarrow$sample$(S'_{(h,\ell,t)})$ // sample a corrupted triplet
10:       $T_{batch} \leftarrow T_{batch} \cup \{((h, \ell, t), (h', \ell, t'))\}$
11:    end for
12:    Update embeddings w.r.t. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla [\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})]_+$
13: end loop
```

Knowledge Graph Embeddings
Vector Representations of Graph Structures

- AmpliGraph: a Library for Representation Learning on Knowledge Graphs, 2019 [12]
- Kaggle international-football-results-from-1872-to-2017

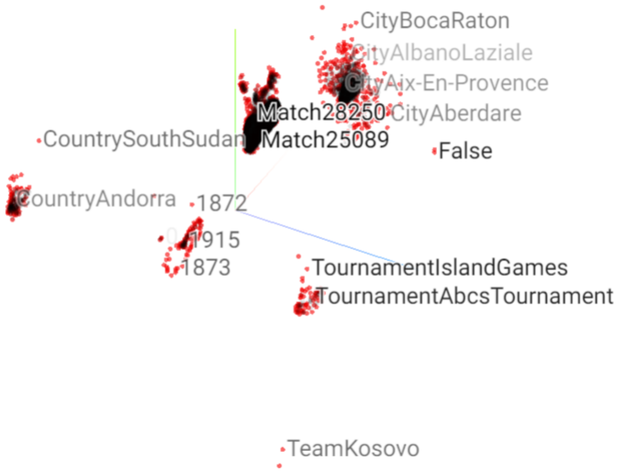


date	home_team	away_team	home_score	away_score	tournament	city
1872-11-30	Scotland	England	0	0	Friendly	Glasgow
1873-03-08	England	Scotland	4	2	Friendly	London
1874-03-07	Scotland	England	2	1	Friendly	Glasgow
1875-03-06	England	Scotland	2	2	Friendly	London
1876-03-04	Scotland	England	3	0	Friendly	Glasgow
1876-03-25	Scotland	Wales	4	0	Friendly	Glasgow
1877-03-03	England	Scotland	1	3	Friendly	London
1877-03-05	Wales	Scotland	0	2	Friendly	Wrexham
1878-03-02	Scotland	England	7	2	Friendly	Glasgow
1878-03-23	Scotland	Wales	9	0	Friendly	Glasgow
1879-01-18	England	Wales	2	1	Friendly	London

Knowledge Graph Embeddings

Vector Representations of Graph Structures

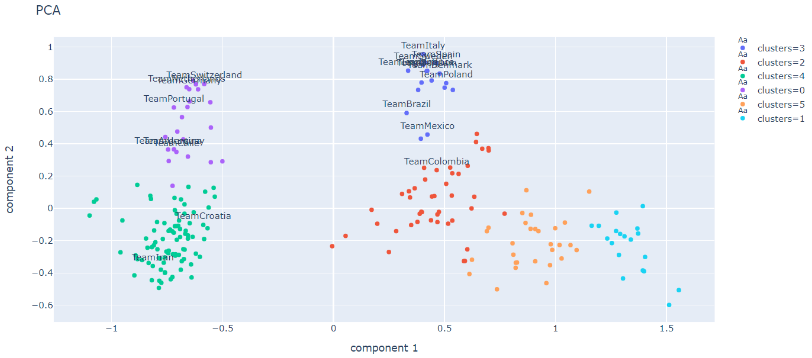
- **392854 triples, 37931 entities, 12 relations,** $TransE(k = 15, epoch = 200, batch = 100)$



Knowledge Graph Embeddings

Vector Representations of Graph Structures

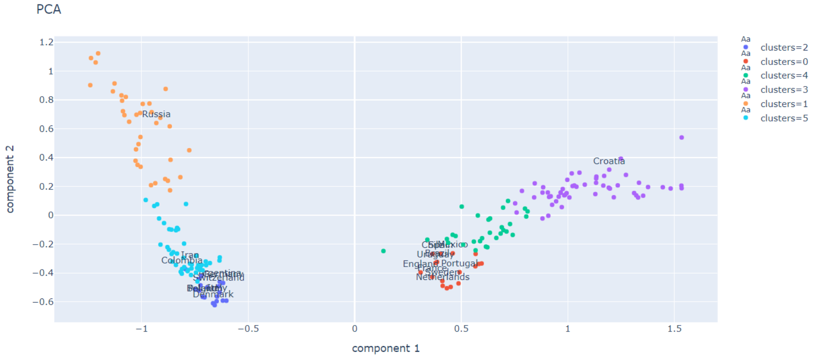
- 392854 triples, 37931 entities, 12 relations, $TransE(k = 15, epoch = 200, batch = 200)$
- Visualize only "team"entities in 2D and try to cluster it



Knowledge Graph Embeddings

Vector Representations of Graph Structures

- 392854 triples, 37931 entities, 12 relations, $TransE(k = 15, epoch = 200, batch = 200)$
- Visualize only "country"entities in 2D and try to cluster it



Knowledge Graph Embeddings

Vector Representations of Graph Structures

- Translation based embedding model or **TransE** [9]
- Learning Entity and Relation Embeddings for Knowledge Graph Completion or **TransR** [10]
- Knowledge Graph Embedding by Translating on Hyperplanes or **TransH** [11]

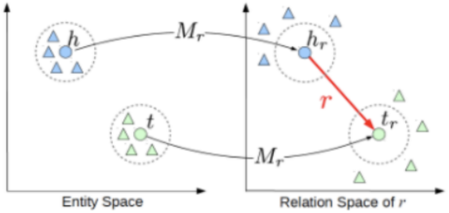
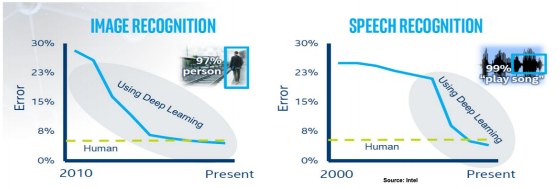


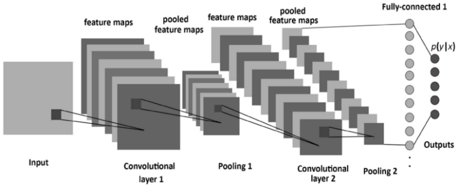
Figure 4: TransR projecting different aspects of an entity to a relationship space.

Graph Convolutional Networks
Vector Representations of Graph Structures

- Convolutional Neural Networks' big impact [13]



- *convolution and pooling functions*



- *sparse interaction, multiple layers, translation invariance*

Graph Convolutional Networks
Vector Representations of Graph Structures

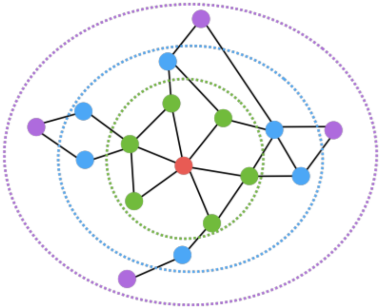
- **Graph Convolutional Networks** (GCN) is a NNs that operate on graphs [14] [15]
- input two matrices: $F_{N\times K}$ feature matrix of nodes, $A_{N\times N}$ adjacency matrix of G
- let $G = (V, E)$, then hidden states (neighborhood aggregation) $\forall v \in V$:

$$h_v = f\left(\frac{1}{|\mathcal{N}(v)|} * \sum_{u\in\mathcal{N}(v)} Wx_u + b\right),$$

  where W (filter matrix), b (bias) - model parameters ,x_u - initial feature for a node u, $\mathcal{N}(v)$ - set of neighbours for node v and f - non-linear activation function (ReLU)
- but h_v capture only 1-hop nodes...

Graph Convolutional Networks
Vector Representations of Graph Structures

- Idea: Consider k-hop node neighbors
- Multi-hop node similarity [16]



- **Red:** Target node
- **Green**: 1-hop neighbors
  - $\mathbf{A}$ (i.e., adjacency matrix)
- **Blue:** 2-hop neighbors
  - $\mathbf{A}^2$
- **Purple:** 3-hop neighbors
  - $\mathbf{A}^3$

Graph Convolutional Networks

Vector Representations of Graph Structures

- **Graph Convolutional Networks** (GCN) is a NNs that operate on graphs [14] [15]
- Basic neighborhood aggregation, h_v^k capture k-hop nodes information

$$h_v^{k+1} = f\left(\frac{1}{|\mathcal{N}(v)|} * \sum_{u \in \mathcal{N}(v)} W^k h_u^k + b^k\right),$$

- GCN neighborhood aggregation

$$h_v^{k+1} = f\left(W_k \sum_{u \in \mathcal{N}(v) \cup v} \frac{h_u^k}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(v)|}}\right)$$

- *more parameter sharing, down-weights high degree neighbors*

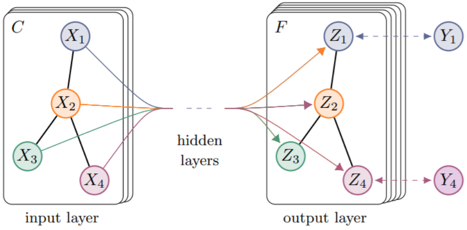
Graph Convolutional Networks

Vector Representations of Graph Structures

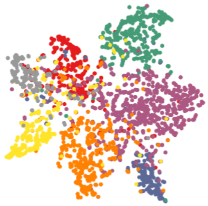
- two-layer GCN for semi-supervised node classification [15]

$$Z = f(X, A) = softmax(\hat{A} ReLU(\hat{A} X W^{(0)}) W^{(1)}),$$

where $\hat{A} = \tilde{D}^{\frac{1}{2}} \tilde{A} \tilde{D}^{\frac{1}{2}}$, $\tilde{D} : \tilde{d} = \sum_j \tilde{a}_{ij}$, $\tilde{A} = A + I_N$



(a) Graph Convolutional Network

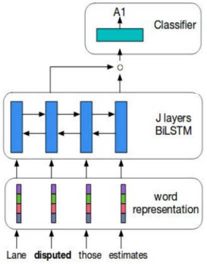


(b) Hidden layer activations

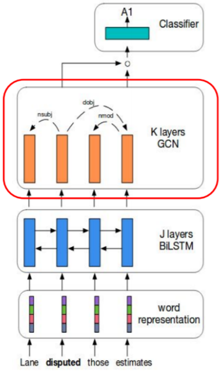
Graph Convolutional Networks

Vector Representations of Graph Structures

- **GCN**: example model architecture for SRL task [13]



Standard Deep Learning Architecture for NLP problems (above is for Semantic-Role Labeling (SRL))



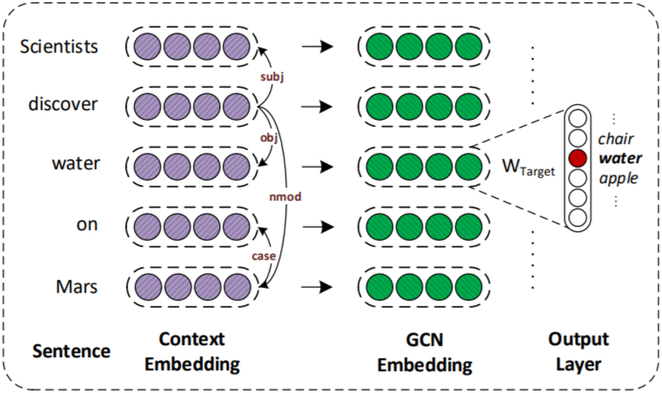
Model with GCN as part of the network

GCN weights are trained based on the final objective

SynGCN

Heterogeneous Information Fusion

- Incorporating Syntactic and Semantic Information in Word Embeddings using Graph Convolutional Networks or **SynGCN** [17]



SynGCN (Sentence-level)

- **SynGCN** [17]
- For a given sentence $s = (w_1, w_2, \ldots, w_n)$, extract its dependency parse graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ using Stanford CoreNLP parser [18]
- Similar to CBOW model [5], where the context of a word w_i is $C_{w_i} = \{w_{i+j} : c \leq j \leq c, j \neq 0\}$ for a window of size c, define the context as its neighbors in $\mathcal{G}_s$, i.e., $C_{w_i} = \mathcal{N}(w_i)$.
- unlike CBOW, which takes the sum of the context embedding of words in C_{w_i} to predict w_i apply directed GCN on $\mathcal{G}_s$ with context embeddings of words in s as feature input
- Thus, for each word w_i in s, we obtain a representation h_i^{k+1} after k-layers of GCN

$$h_i^{k+1} = f\left(\sum_{j \in \mathcal{N}(i)} g_{l_{ij}}^k \times (W_{l_{ij}}^k h_j^k + b_{l_{ij}}^k)\right),$$

SynGCN

Heterogeneous Information Fusion

Theorem

SynGCN is a generalization of Continuous-bag-of-words (CBOW) model.

Proof

For a given sentence s, take the neighborhood of each word w_i in $\mathcal{G}$ as it sequential context, i.e., w_i is $\mathcal{N}_{w_i} = \{w_{i+j} : c \leq j \leq c, j \neq 0\}\ \forall w_i \in s$. Now, if the number of GCN layers are restricted to 1 and the activation function is taken as identity ($f(x) = x$), then Equation 1 reduces to

$$h_i = \sum_{c \leq j \leq c, j \neq 0} g_{l_{ij}} \times (W_{l_{ij}} h_j^k + b_{l_{ij}}),$$

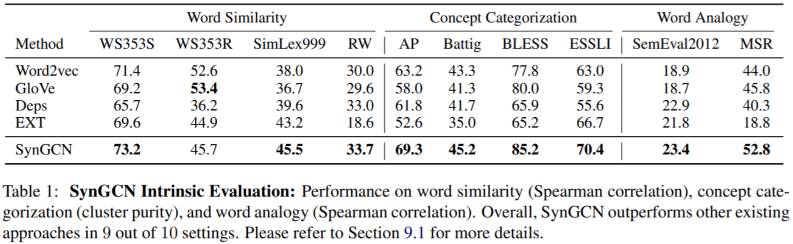
Finally, $Wk_{l_{ij}}$ and $b_{l_{ij}}$ can be fixed to an identity matrix $(\mathbf{I})$ and zero vector $(\mathbf{0})$, edge-wise gating $g_{l_{ij}}$ can be set to 1. This gives

$$h_i = \sum_{c \leq j \leq c, j \neq 0} \mathbf{I} h_j + \mathbf{0} = \sum_{c \leq j \leq c, j \neq 0} h_j \quad \square$$

SynGCN

Heterogeneous Information Fusion

- **SynGCN** [17]



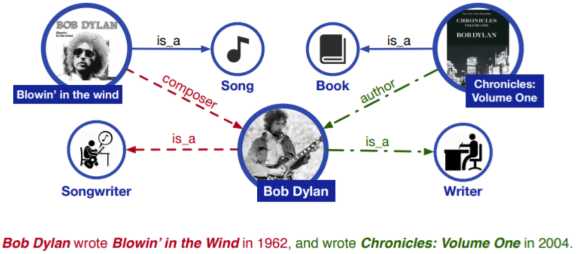
	Word Similarity				Concept Categorization				Word Analogy	
Method	WS353S	WS353R	SimLex999	RW	AP	Battig	BLESS	ESSLI	SemEval2012	MSR
Word2vec	71.4	52.6	38.0	30.0	63.2	43.3	77.8	63.0	18.9	44.0
GloVe	69.2	**53.4**	36.7	29.6	58.0	41.3	80.0	59.3	18.7	45.8
Deps	65.7	36.2	39.6	33.0	61.8	41.7	65.9	55.6	22.9	40.3
EXT	69.6	44.9	43.2	18.6	52.6	35.0	65.2	66.7	21.8	18.8
SynGCN	**73.2**	45.7	**45.5**	**33.7**	**69.3**	**45.2**	**85.2**	**70.4**	**23.4**	**52.8**

Table 1: **SynGCN Intrinsic Evaluation:** Performance on word similarity (Spearman correlation), concept categorization (cluster purity), and word analogy (Spearman correlation). Overall, SynGCN outperforms other existing approaches in 9 out of 10 settings. Please refer to Section 9.1 for more details.

- https://github.com/malllabiisc/WordGCN

ERNIE

Heterogeneous Information Fusion



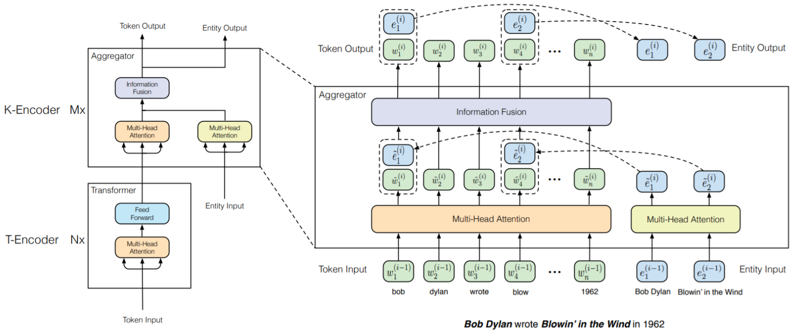
BOB DYLAN
is_a
is_a
Blowin' in the wind
Song
Book
CHRONICLES VOLUME ONE BOB DYLAN
Chronicles: Volume One
composer
author
is_a
is_a
Songwriter
Bob Dylan
Writer
Bob Dylan wrote Blowin' in the Wind in 1962, and wrote Chronicles: Volume One in 2004.

- Without knowing Blowin' in the Wind and Chronicles: Volume One are song and book respectively [20]
  1. it is difficult to recognize the two occupations of Bob Dylan, i.e. songwriter and writer, on the entity typing task.
  2. it is nearly impossible to extract the fine-grained relations, such and composer and author on the relation classification task
  3. "UNK wrote UNK in UNK"

ERNIE

Heterogeneous Information Fusion

- **ERNIE**: Enhanced Language Representation with Informative Entities [20]



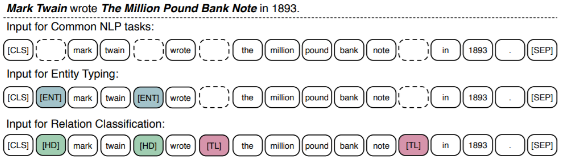
(a) Model Achitecture (b) Aggregator

- pre-training corpus: 4,500M subwords and 140M entities from English Wikipedia

ERNIE

Heterogeneous Information Fusion

- **ERNIE** consists of two stacked modules: [20]
  1. the underlying textual encoder (T-Encoder) responsible to capture basic lexical and syntactic information from the input tokens, and (2) the
  2. knowledgeable encoder (K-Encoder) responsible to integrate extra token-oriented knowledge information into textual information from the underlying layers



Mark Twain wrote *The Million Pound Bank Note* in 1893.
Input for Common NLP tasks:
[CLS] mark twain wrote the million pound bank note in 1893 . [SEP]
Input for Entity Typing:
[CLS] [ENT] mark twain [ENT] wrote the million pound bank note in 1893 . [SEP]
Input for Relation Classification:
[CLS] [HD] mark twain [HD] wrote [TL] the million pound bank note [TL] in 1893 . [SEP]

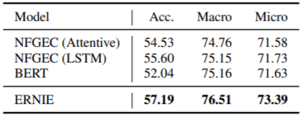
Figure 3: Modifying the input sequence for the specific tasks. To align tokens among different types of input, we use dotted rectangles as placeholder. The colorful rectangles present the specific mark tokens.

- based on state-of-the-art BERT model [21]

ERNIE

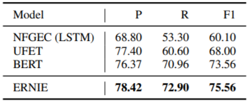
Heterogeneous Information Fusion

- **ERNIE** [20] experimental results:
  The experimental results demonstrate that ERNIE has better abilities of both denoising distantly supervised data and fine-tuning on limited data than BERT



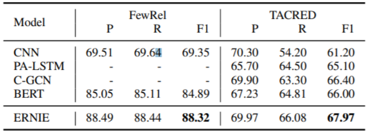
Model	Acc.	Macro	Micro
NFGEC (Attentive)	54.53	74.76	71.58
NFGEC (LSTM)	55.60	75.15	71.73
BERT	52.04	75.16	71.63
ERNIE	**57.19**	**76.51**	**73.39**

Table 2: Results of various models on FIGER (%).



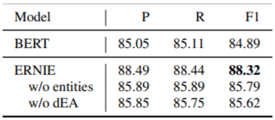
Model	P	R	F1
NFGEC (LSTM)	68.80	53.30	60.10
UFET	77.40	60.60	68.00
BERT	76.37	70.96	73.56
ERNIE	**78.42**	**72.90**	**75.56**

Table 3: Results of various models on Open Entity (%).



Model	FewRel			TACRED		
	P	R	F1	P	R	F1
CNN	69.51	69.64	69.35	70.30	54.20	61.20
PA-LSTM	-	-	-	65.70	64.50	65.10
C-GCN	-	-	-	69.90	63.30	66.40
BERT	85.05	85.11	84.89	67.23	64.81	66.00
ERNIE	88.49	88.44	**88.32**	69.97	66.08	**67.97**

Table 5: Results of various models on FewRel and TACRED (%).



Model	P	R	F1
BERT	85.05	85.11	84.89
ERNIE	88.49	88.44	**88.32**
w/o entities	85.89	85.89	85.79
w/o dEA	85.85	85.75	85.62

Table 7: Ablation study on FewRel (%).

References

- https://github.com/VirtualRoyalty/graphs-in-nlp-2020

1 Fusing Document, Collection and Label Graph-based Representations with Word Embeddings for Text Classification
2 Graph-based Text Representations: Boosting Text Mining, NLP and Information Retrieval with Graphs
3 InfraNodus: Generating Insight Using Text Network Analysi
4 Tackling Graphical NLP problems with Graph Recurrent Networks
5 Distributed Representations of Words and Phrases and their Compositionality
6 DeepWalk: Online Learning of Social Representations
7 Efficient Representation Learning Using Random Walks for Dynamic Graphs
8 node2vec: Scalable Feature Learning for Networks
9 Translating Embeddings for Modeling Multi-relational Data
10 Learning entity and relation embeddings for knowledge graph completion
11 Knowledge Graph Embedding by Translating on Hyperplanes
12 AmpliGraph: a Library for Representation Learning on Knowledge Graphs
13 A Tutorial on Graph Neural Networks for Natural Language Processing
14 The graph neural network model
15 SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS
16 Standford: Representation Learning on Networks
17 Incorporating Syntactic and Semantic Information in Word Embeddings using Graph Convolutional Networks
18 The Stanford CoreNLP Natural Language Processing Toolkit
19 Learning beyond datasets: Knowledge Graph Augmented Neural Networks for Natural language Processing
20 ERNIE: Enhanced Language Representation with Informative Entities
21 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding