Exploiting Russian Word Embeddings for Automated Grammeme Prediction

Alexey Romanov
alexey.romanov@phystech.edu

ABBYY
Moscow Institute of Physics and Technology

31 May 2017

Grammemes in the Russian Language

- **Grammatical category** — grammatical property with mutually exclusive possible values.
- **Grammeme** — specific value of grammatical category.
- Example grammatical categories:
  - Noun: case, number, gender...
  - Verb: mood, number, tense...
- Example grammemes:
  - *языкам*: dative, plural, masculine...
  - *говорит*: indicative, singular, present...

Grammemes in the Russian Language

- Grammatical categories can be of two types:
  - **Classifying** grammatical categories have single grammeme for all lexeme forms.
  - **Modifying** grammatical categories vary while changing lexeme forms.
- Examples:
  - classifying: noun gender, verb transitivity
  - modifying: noun case, adjective gender
- Modifying grammemes in Russian can be guessed by word's appearance.
- Not so rosy for classifying grammemes:
  - noun animacy: *проектор/проректор*, *повестка/невестка*
  - verb transitivity: *иметь/сметь*, *ворошить/ворожить*

Formal Problem Statement

Grammeme prediction as a supervised learning task:

- **Given:** labeled set $X^m = \{(x_i, y_i)\}_{i=1}^m$
  - lexemes $x_i \in X$
  - grammemes $y_i \in Y$ of a grammatical category
- **Task:** find $a : X \to Y$ minimizing error rate $\sum_{i=1}^m [a(x_i) \neq y_i]$

Focus on 2 binary classification tasks:

- noun animacy
- verb transitivity

Formal Problem Statement

Grammeme prediction as a supervised learning task:

- **Given:** labeled set $X^m = \{(x_i, y_i)\}_{i=1}^m$
  - lexemes $x_i \in X$
  - grammemes $y_i \in Y$ of a grammatical category
- **Task:** find $a : X \to Y$ minimizing error rate $\sum_{i=1}^m [a(x_i) \neq y_i]$

Focus on 2 binary classification tasks:

- noun animacy
- verb transitivity

Disclaimer: We use binary labels, although in fact they **are not** binary.

- homonymy & polysemy
- no "strictly transitive" words

Possible Applications

Possible applications include:

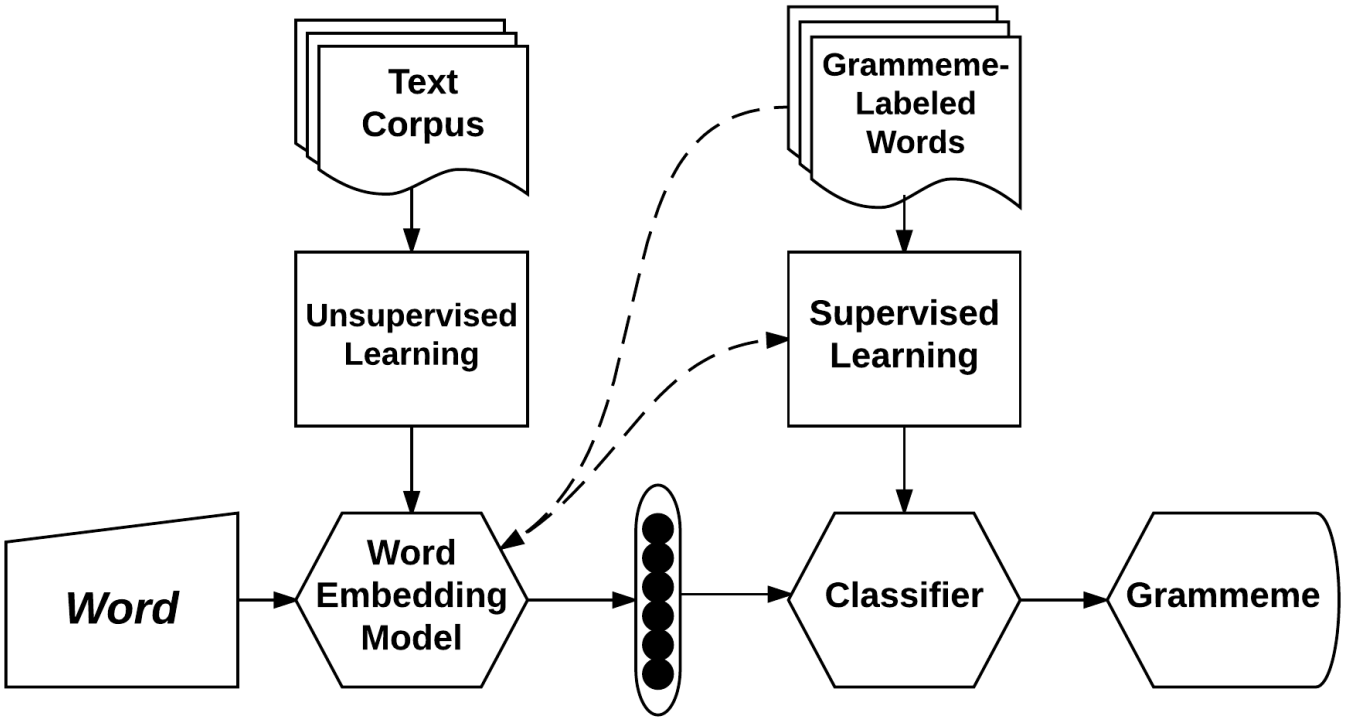
- Automatic and semi-automatic morphological annotation
  - corpus annotation for languages lacking labeled data
  - prediction of morphological features for out-of-vocabulary words
- Intermediate feature generation for more complex tasks
  - syntactic parsing
  - text similarity estimation
  - relation extraction
  - machine translation etc.

Related Work

1. Bowman et al. *Automatic Animacy Classification* (2012)
   - *Language:* English
   - *Data:* labeled noun phrases
   - *Features:* syntactic functions and dependency types
   - *Accuracy:* 93%
2. Bloem et al. *Automatic animacy classification for Dutch* (2013)
   - *Language:* Dutch
   - *Data:* full syntactic dependency trees
   - *Features:* dependency frequency ratio
   - *Accuracy:* 92%
3. Qiu et al. *Investigating language universal and specific properties in word embeddings* (2016)
   - *Language:* Arabic, Chinese, Hindi...
   - *Data:* text corpora labeled with POS tags
   - *Features:* word embeddings
   - *Accuracy:* 90%

Proposed Method Description

- *Language:* Russian
- *Data:* **unlabeled** text corpus
- *Features:* word embeddings trained on the corpus
  - `word2vec`
  - `FastText`



Implementation Details

- *Data:* 1.3M Wikipedia articles (100M tokens)
- *Preprocessing:*
  - split into sentences
  - lowercasing
  - removal of punctuation & non-Cyrillic letters
  - lemmatization
- *Software:*
  - `pymorphy2` for lemmatization and grammeme labeling
  - `gensim` & `fasttext` for word embeddings
  - `scikit-learn` for classification

- skip-gram context/word likelihood:

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{W} e^{s(w_t,j)}}$$

- `word2vec` skip-gram similarity: $s(w_t, w_c) = \mathbf{u}_{w_t}^{\mathsf{T}} \mathbf{v}_{w_c}$
- $\mathcal{G}_w$ — N-gram set of w:
  - $3 \leqslant N \leqslant 6$
  - $\mathcal{G}_{words} = \{wor, ord, rds, word, ords, words\}$
- `FastText` similarity: $s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^{\mathsf{T}} \mathbf{v}_c$
- `FastText` model can predict vectors for unseen words based on their N-gram vectors.

Experiments

Experiments: General Feasibility

Embedding model parameters:

- dimension: 250, **500**
- context window: 0+3, 3+0, 2+2, **5+5**
- `word2vec` & `FastText` **skip-gram** / CBOW
- `FastText` prediction for unseen words

ML algorithms:

- SVM (linear kernel), Random Forest, Multi-Layer Perceptron

Metrics: weighted F1

	transitivity			animacy		
model	SVM	RF	MLP	SVM	RF	MLP
`w2v`, best param	0.833	0.748	0.831	**0.888**	0.870	0.873
`ft`, best param	0.859	0.834	**0.868**	0.848	0.820	0.830
`ft`, predict	0.840	0.825	0.862	0.797	0.789	0.811

Experiments: Stacking w2v Matrices

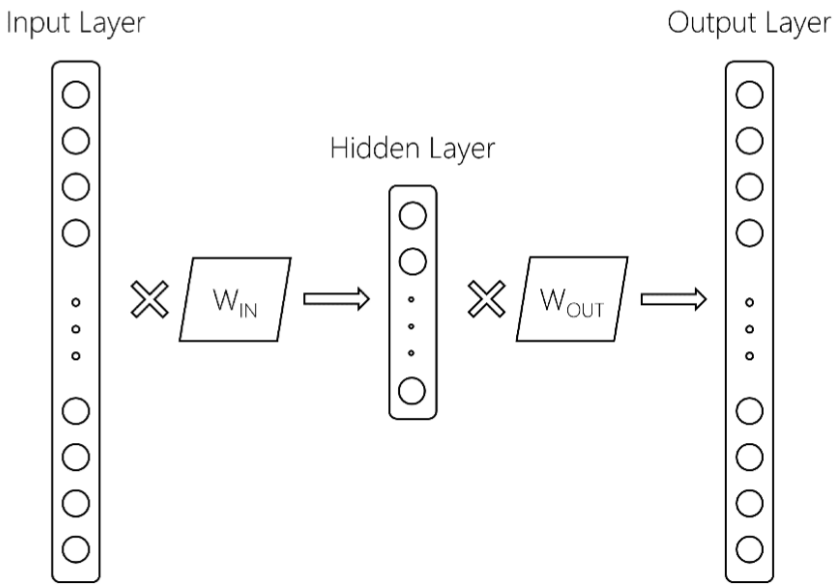


Рис. 1: Principle scheme of w2v architecture

	transitivity			animacy		
features	SVM	RF	MLP	SVM	RF	MLP
W_{in}	0.833	0.748	0.831	0.888	0.870	0.873
W_{out}	0.848	0.755	0.844	0.886	0.871	0.865
$[W_{in}, W_{out}]$	**0.852**	0.841	0.862	**0.893**	0.876	0.883

Discussion

- `FastText` outperforms `w2v` when N-grams matter:
  - e.g., it can capture suffixes of transitive verbs
- Non-linear relations of embedding components can be useful.
  - MLP $>$ SVM for transitivity
- `FastText` prediction works when N-grams matter
  - 50K seen nouns, 70K unseen nouns
  - 6K seen verbs, 6K unseen verbs
- Auxiliary `w2v` training information can be useful.
  - Both W_{in} and W_{out} matrices work.

Error Analysis

- Homonymy and polysemy
  - *изменить* — trans. “to change” or intrans. “to cuckold”
  - *везти* — trans. “to carry” or intrans. “to be lucky”
  - *барак* — “a barrack” and *Барак* — “Barack”
- Rare words with few occurrences in the corpus
  - *дефилировать* — “to sashay”
  - *хлебопашец* — “a sodbuster”
- Transitive verbs used without a direct object
  - *петь* — “to sing”
- Inanimate proper names that can be seen as human names
  - *Бредфорд* — “Bradford”

Conclusion & Future Plans

- The method showed its feasibility for fast and accurate classification of unlabeled data.
- The results are comparable to previous work with more complex classification features.
- `FastText` enables prediction of grammemes for unseen data.
- Future planes include:
  - further investigation of word embedding models
  - applying pooling schemes on unlemmatized data
  - extension of the approach to various languages

Thanks for your attention

Alexey Romanov

alexey.romanov@phystech.edu