Russian Word Sense Induction by Clustering Averaged Word Embeddings

Andrey Kutuzov
andreku@ifi.uio.no

University of Oslo

May 31, 2017

Contents

Introducing the task

TL:DR

- This is our experience of participating in RUSSE'18 shared task (knowledge-free track).

Introducing the task

TL:DR

- This is our experience of participating in RUSSE'18 shared task (knowledge-free track).
- Our system ranked 2nd for the *wiki-wiki* dataset...

Introducing the task

TL:DR

- This is our experience of participating in RUSSE'18 shared task (knowledge-free track).
- Our system ranked 2nd for the *wiki-wiki* dataset...
- and 3rd for the *bts-rnc* and *active-dict* datasets.

Introducing the task

TL:DR

- This is our experience of participating in RUSSE'18 shared task (knowledge-free track).
- Our system ranked 2nd for the *wiki-wiki* dataset...
- and 3rd for the *bts-rnc* and *active-dict* datasets.
- Method: naive clustering of contexts represented with averaged word embeddings.

Introducing the task

TL:DR

- This is our experience of participating in RUSSE'18 shared task (knowledge-free track).
- Our system ranked 2nd for the *wiki-wiki* dataset...
- and 3rd for the *bts-rnc* and *active-dict* datasets.
- Method: naive clustering of contexts represented with averaged word embeddings.
- Takeaway message: small but balanced corpora are superior again.

Introducing the task

Contributions

- Can Russian Word Sense Induction (WSI) task be solved using only already available algorithms and off-the-shelf models?

Introducing the task

Contributions

- Can Russian Word Sense Induction (WSI) task be solved using only already available algorithms and off-the-shelf models?
- It can, especially for the *wiki-wiki* dataset.

Introducing the task

Contributions

- Can Russian Word Sense Induction (WSI) task be solved using only already available algorithms and off-the-shelf models?
- It can, especially for the *wiki-wiki* dataset.
- WSI system for Russian is described and published.

Introducing the task

Contributions

- Can Russian Word Sense Induction (WSI) task be solved using only already available algorithms and off-the-shelf models?
- It can, especially for the *wiki-wiki* dataset.
- WSI system for Russian is described and published.
- It successfully extracts word senses for homonyms and is based exclusively on off-the-shelf tools and models.

Introducing the task

Contributions

- Can Russian Word Sense Induction (WSI) task be solved using only already available algorithms and off-the-shelf models?
- It can, especially for the *wiki-wiki* dataset.
- WSI system for Russian is described and published.
- It successfully extracts word senses for homonyms and is based exclusively on off-the-shelf tools and models.
- Training corpus balance is very important for word embedding models in intrinsic evaluation...
- this holds for extrinsic evaluation setting as well (WSI in this case).

Contents

What we all know about WSI

- Human language is ambiguous on all tiers.

What we all know about WSI

- Human language is ambiguous on all tiers.
- Syntactic ambiguity is solved by PoS taggers and dependency parsers.

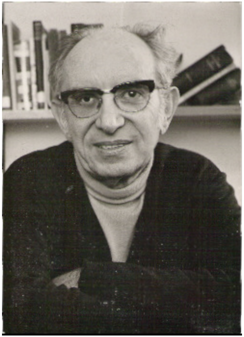
What we all know about WSI

- Human language is ambiguous on all tiers.
- Syntactic ambiguity is solved by PoS taggers and dependency parsers.
- ...but words can possess different senses/meanings.

What we all know about WSI

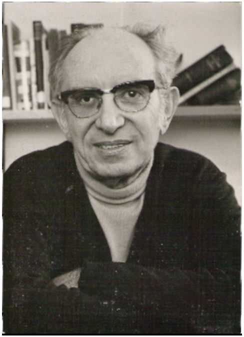
- Human language is ambiguous on all tiers.
- Syntactic ambiguity is solved by PoS taggers and dependency parsers.
- ...but words can possess different senses/meanings.
- All that happens with semantics, happens at the level of word senses, not words.

What we all know about WSI

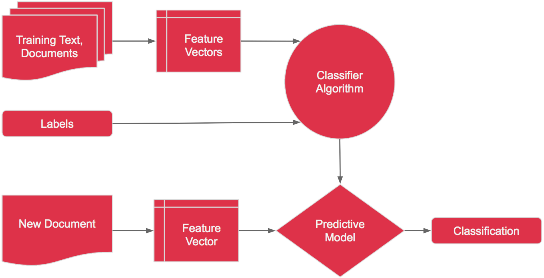


- Even word sense disambiguation is difficult for computers.
- Yehoshua Bar-Hillel:
  - '*sense ambiguity could not be resolved by electronic computer either current or imaginable*' [Bar-Hillel, 1964].

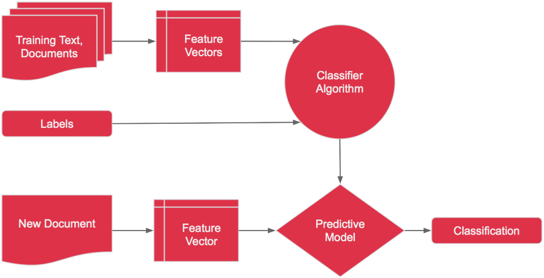
What we all know about WSI



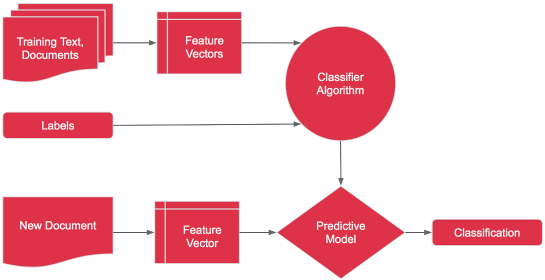
- Even word sense disambiguation is difficult for computers.
- Yehoshua Bar-Hillel:
  - '*sense ambiguity could not be resolved by electronic computer either current or imaginable*' [Bar-Hillel, 1964].
- But people learned how to disambiguate word senses in a supervised setup...
- using manually annotated semantic concordances and lexical databases.



- Supervised or corpus-based WSD makes use of machine learning:



- Supervised or corpus-based WSD makes use of machine learning:
  1. annotate text with word senses and train classifiers on this data;



- Supervised or corpus-based WSD makes use of machine learning:
  1. annotate text with word senses and train classifiers on this data;
  2. at test time collect features and predict the correct sense with the classifier.

What we all know about WSI

Problem: knowledge acquisition bottleneck

- Sense inventory is needed for each word.

What we all know about WSI

Problem: knowledge acquisition bottleneck

- Sense inventory is needed for each word.
- Manually annotated resources quickly get outdated.

What we all know about WSI

Problem: knowledge acquisition bottleneck

- Sense inventory is needed for each word.
- Manually annotated resources quickly get outdated.
- They don't keep up with the changes in language.

What we all know about WSI

Problem: knowledge acquisition bottleneck

- Sense inventory is needed for each word.
- Manually annotated resources quickly get outdated.
- They don't keep up with the changes in language.
- Humans simply can't annotate that fast.

What we all know about WSI

- However, it is relatively easy to compile large up-to-date corpora of unannotated text.

What we all know about WSI

- However, it is relatively easy to compile large up-to-date corpora of unannotated text.
- We can try to infer sense inventories from these corpora automatically.

What we all know about WSI

- However, it is relatively easy to compile large up-to-date corpora of unannotated text.
- We can try to infer sense inventories from these corpora automatically.
- No pre-defined sense inventories.

What we all know about WSI

- However, it is relatively easy to compile large up-to-date corpora of unannotated text.
- We can try to infer sense inventories from these corpora automatically.
- No pre-defined sense inventories.
- This is called word sense induction (WSI).

What we all know about WSI

- However, it is relatively easy to compile large up-to-date corpora of unannotated text.
- We can try to infer sense inventories from these corpora automatically.
- No pre-defined sense inventories.
- This is called word sense induction (WSI).
- One can call it ‘unsupervised WSD’.
- INPUT is corpus, OUTPUT is sense sets for each content word in the corpus.

What we all know about WSI

- However, it is relatively easy to compile large up-to-date corpora of unannotated text.
- We can try to infer sense inventories from these corpora automatically.
- No pre-defined sense inventories.
- This is called word sense induction (WSI).
- One can call it ‘unsupervised WSD’.
- INPUT is corpus, OUTPUT is sense sets for each content word in the corpus.

‘*Word senses are abstractions from clusters of corpus citations*’
[Kilgarriff, 1997]

What we all know about WSI

Foundations for clustering-based WSI were laid in [Jones, 1964] and [Schutze, 1998].

What we all know about WSI

Foundations for clustering-based WSI were laid in [Jones, 1964] and [Schutze, 1998].
Very straightforward approach based on word distributions:

What we all know about WSI

Foundations for clustering-based WSI were laid in [Jones, 1964] and [Schutze, 1998].

Very straightforward approach based on word distributions:

1. Represent each ambiguous word with a list of its context vectors;

What we all know about WSI

Foundations for clustering-based WSI were laid in [Jones, 1964] and [Schutze, 1998].

Very straightforward approach based on word distributions:

1. Represent each ambiguous word with a list of its context vectors;
2. context vector contains identifiers of context words in a particular context

What we all know about WSI

Foundations for clustering-based WSI were laid in [Jones, 1964] and [Schutze, 1998].

Very straightforward approach based on word distributions:

1. Represent each ambiguous word with a list of its context vectors;
2. context vector contains identifiers of context words in a particular context
3. For each word, cluster its lists into a (predefined) number of groups;

What we all know about WSI

Foundations for clustering-based WSI were laid in [Jones, 1964] and [Schutze, 1998].

Very straightforward approach based on word distributions:

1. Represent each ambiguous word with a list of its context vectors;
2. context vector contains identifiers of context words in a particular context
3. For each word, cluster its lists into a (predefined) number of groups;
4. For each cluster, find its centroid;

What we all know about WSI

Foundations for clustering-based WSI were laid in [Jones, 1964] and [Schutze, 1998].

Very straightforward approach based on word distributions:

1. Represent each ambiguous word with a list of its context vectors;
2. context vector contains identifiers of context words in a particular context
3. For each word, cluster its lists into a (predefined) number of groups;
4. For each cluster, find its centroid;
5. Centroids serve as sense vectors for WSD.

What we all know about WSI

At test time:

What we all know about WSI

At test time:

1. We are given a new context (e.g. sentence) with an ambiguous input word;
2. compute current context vector (just list context words);

What we all know about WSI

At test time:

1. We are given a new context (e.g. sentence) with an ambiguous input word;
2. compute current context vector (just list context words);
3. choose the sense with the vector most similar to the current context vector.

What we all know about WSI

At test time:

1. We are given a new context (e.g. sentence) with an ambiguous input word;
2. compute current context vector (just list context words);
3. choose the sense with the vector most similar to the current context vector.

- NB: senses are 'coarse', nameless and often not directly interpretable.

What we all know about WSI

At test time:

1. We are given a new context (e.g. sentence) with an ambiguous input word;
2. compute current context vector (just list context words);
3. choose the sense with the vector most similar to the current context vector.

- NB: senses are 'coarse', nameless and often not directly interpretable.
- The approach can be enriched with additional techniques like lexical substitution [Alagić et al., 2018]

What we all know about WSI

Word embeddings

- Instead of one-hot word vectors, one can use distributional information about word meanings.

What we all know about WSI

Word embeddings

- Instead of one-hot word vectors, one can use distributional information about word meanings.
- To this end, we employ prediction-based word embedding models:
  - *Continuous Skipgram* [Mikolov et al., 2013]
  - *fastText* [Bojanowski et al., 2017]

Contents

Datasets and models overview

RUSSE'18 offered three datasets:

1. *wiki-wiki*: sense inventories and contexts from the Russian Wikipedia articles;
2. *bts-rnc*: sense inventories from 'Bolshoi Tolkovii Slovar' dictionary (BTS), contexts from the Russian National Corpus;
3. *active-dict*: sense inventories from the Active Dictionary of the Russian Language, contexts from the examples in the same dictionary.

Datasets and models overview

wiki-wiki dataset is substantially different from the other two:

Training dataset	**wiki-wiki**	**bts-rnc**	**active-dict**
Average number of senses	2	3.2	3.7
Maximum number of senses	2	8	17

Datasets and models overview

Different nature of these senses

- wiki-wiki mostly contains homonyms: word senses are unrelated:

Datasets and models overview

Different nature of these senses

- wiki-wiki mostly contains homonyms: word senses are unrelated:
  - бор^{1} (pinewood) and бор^{2} (Boron)

Datasets and models overview

Different nature of these senses

- wiki-wiki mostly contains homonyms: word senses are unrelated:
  - бор^1 (pinewood) and бор^2 (Boron)
- bts-rnc and active-dict contain polysemous words: senses are related:

Different nature of these senses

- wiki-wiki mostly contains homonyms: word senses are unrelated:
  - бор^1 (pinewood) and бор^2 (Boron)
- bts-rnc and active-dict contain polysemous words: senses are related:
  - обед^1 (lunch) and обед^2 (lunchtime)

Different nature of these senses

- wiki-wiki mostly contains homonyms: word senses are unrelated:
  - бор^1 (pinewood) and бор^2 (Boron)
- bts-rnc and active-dict contain polysemous words: senses are related:
  - обед^1 (lunch) and обед^2 (lunchtime)
  - дерево^1 (tree) and дерево^2 (wood)

Word senses represent some sort of a continuum

- There is no distinct boundary between homonymy and polysemy.

Word senses represent some sort of a continuum

- There is no distinct boundary between homonymy and polysemy.
- Often difficult to tell how many senses does a word really have.

Datasets and models overview

Word senses represent some sort of a continuum

- There is no distinct boundary between homonymy and polysemy.
- Often difficult to tell how many senses does a word really have.
- But still:
- Inducing meanings of homonyms is a different and easier task than inducing different sense of polysemous words.

Datasets and models overview

Pre-trained word embedding models from the RusVectōrēs web service [Kutuzov and Kuzmenko, 2016]

Model id	corpus	corpus size, words	algorithm
ruscorpora_upos_skipgram_300_5_2018	Russian National Corpus (RNC)	250M	word2vec skipgram
ruwikiruscorpora_upos_skipgram_300_2_2018	RNC + Wikipedia	600M	word2vec skipgram
news_upos_cbow_600_2_2018	News corpus	5000M	word2vec CBOW
araneum_upos_skipgram_300_2_2018	Araneum Russicum Maximum	10000M	word2vec skipgram
araneum_none_fasttextskipgram_300_5_2018	Araneum Russicum Maximum	10000M	fastText (char 3-grams)

Prior to training, all the corpora were tokenized, split into sentences, lemmatized and PoS-tagged using *UDPipe* [Straka and Straková, 2017]. All the models use vector size 300.

Contents

Averaging embeddings to get senses

Our approach consisted of the following steps:

1. Lemmatize and PoS-tag contexts;

Averaging embeddings to get senses

Our approach consisted of the following steps:

1. Lemmatize and PoS-tag contexts;
2. Represent each context as a fixed-length vector manifesting its semantics;

Averaging embeddings to get senses

Our approach consisted of the following steps:

1. Lemmatize and PoS-tag contexts;
2. Represent each context as a fixed-length vector manifesting its semantics;
3. Determine the number of clusters in the set of contexts, using the *Affinity Propagation* algorithm;
4. Group the contexts into clusters representing word senses, using either *Affinity Propagation* or other clustering algorithm.

Contexts representation

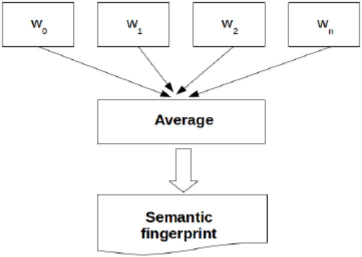
1. Each word in the context is mapped to its vector in the model;

Contexts representation

1. Each word in the context is mapped to its vector in the model;
2. Ambiguous query word itself is removed;

Contexts representation

1. Each word in the context is mapped to its vector in the model;
2. Ambiguous query word itself is removed;
3. For each context utterance, a 'semantic fingerprint' [Kutuzov et al., 2016] is created:
4. Weighted average of all words' vectors.
5. This dense vector is our semantic representation of the context utterance.



Contexts representation

A few modifications:

1. Multiple occurrences of the same lemma counted as one occurrence:

Contexts representation

A few modifications:

1. Multiple occurrences of the same lemma counted as one occurrence:
   - binary bag-of-words, to discard local word frequencies in the contexts;

Contexts representation

A few modifications:

1. Multiple occurrences of the same lemma counted as one occurrence:
   - binary bag-of-words, to discard local word frequencies in the contexts;
2. The average was weighted by word frequencies in the training corpus of the embedding model used:
   - globally frequent words get less influence,
   - globally rare words are more influential.

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.
- But the number of senses (clusters) for each query word is unknown.

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.
- But the number of senses (clusters) for each query word is unknown.
- Thus, the algorithm should be able to induce it from the data.

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.
- But the number of senses (clusters) for each query word is unknown.
- Thus, the algorithm should be able to induce it from the data.
- We employed *Affinity Propagation*.

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.
- But the number of senses (clusters) for each query word is unknown.
- Thus, the algorithm should be able to induce it from the data.
- We employed *Affinity Propagation*.
- It detects the supposed number of clusters and provides the clustering itself.

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.
- But the number of senses (clusters) for each query word is unknown.
- Thus, the algorithm should be able to induce it from the data.
- We employed *Affinity Propagation*.
- It detects the supposed number of clusters and provides the clustering itself.
- Once again: datasets are different!

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.
- But the number of senses (clusters) for each query word is unknown.
- Thus, the algorithm should be able to induce it from the data.
- We employed *Affinity Propagation*.
- It detects the supposed number of clusters and provides the clustering itself.
- Once again: datasets are different!
  - For *wiki-wiki* we simply used this clustering.

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.
- But the number of senses (clusters) for each query word is unknown.
- Thus, the algorithm should be able to induce it from the data.
- We employed *Affinity Propagation*.
- It detects the supposed number of clusters and provides the clustering itself.
- Once again: datasets are different!
  - For *wiki-wiki* we simply used this clustering.
  - For *bts-rnc* and *active-dict* we employed *K-Means* or *Spectral Clustering* to separate the contexts into the number of clusters detected by *Affinity Propagation*.

Contexts clustering

- Any clustering algorithm can be used to group contexts into sense clusters.
- But the number of senses (clusters) for each query word is unknown.
- Thus, the algorithm should be able to induce it from the data.
- We employed *Affinity Propagation*.
- It detects the supposed number of clusters and provides the clustering itself.
- Once again: datasets are different!
  - For *wiki-wiki* we simply used this clustering.
  - For *bts-rnc* and *active-dict* we employed *K-Means* or *Spectral Clustering* to separate the contexts into the number of clusters detected by *Affinity Propagation*.
  - This gave the best performance.

Contexts clustering

A simple system, but reasonable clusterings:

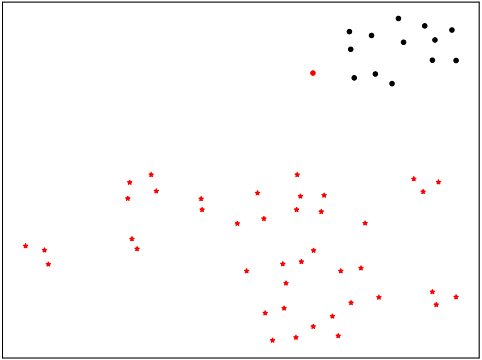


Figure: Clustering of the бор contexts (‘pine wood’ and ‘Boron’). Each point is a context, 2-dimensional t-SNE projection. Colors are clusters assigned by the system, shapes are gold clusters.

Contents

And what? The results

ARI on the training data, dependent on the word embedding model

Model id	wiki-wiki	bts-rnc	active-dict
ruscorpora_upos_skipgram_300_5_2018	**0.772**	**0.176**	**0.260**
ruwikiruscorpora_upos_skipgram_300_2_2018	0.669	0.162	0.210
news_upos_cbow_600_2_2018	0.653	0.174	0.143
araneum_upos_skipgram_300_2_2018	0.492	0.162	0.197
araneum_none_fasttextskipgram_300_5_2018	0.695	0.171	0.178

And what? The results

ARI on the training data, dependent on the word embedding model

Model id	wiki-wiki	bts-rnc	active-dict
ruscorpora_upos_skipgram_300_5_2018	**0.772**	**0.176**	**0.260**
ruwikiruscorpora_upos_skipgram_300_2_2018	0.669	0.162	0.210
news_upos_cbow_600_2_2018	0.653	0.174	0.143
araneum_upos_skipgram_300_2_2018	0.492	0.162	0.197
araneum_none_fasttextskipgram_300_5_2018	0.695	0.171	0.178

The RNC wins, despite being significantly smaller.
Properly compiling and balancing the training corpora for word embedding models is extremely important: even in an extrinsic evaluation setting like WSI.

And what? The results

ARI on the training data, dependent on the parameters of word vector averaging

Dataset	Original	+binary BOW	+weights
wiki-wiki	0.579	0.717	**0.772**
bts-rnc	0.169	0.167	0.176
active-dict	0.250	0.254	0.260

And what? The results

ARI on the training data, dependent on the parameters of word vector averaging

Dataset	Original	+binary BOW	+weights
wiki-wiki	0.579	0.717	**0.772**
bts-rnc	0.169	0.167	0.176
active-dict	0.250	0.254	0.260

The effect of binary bag-of-words and weights is most visible on the *wiki-wiki* dataset.

And what? The results

ARI on the test data

Dataset	Our ARI	Rank (of 17)	The best ARI
wiki-wiki	0.7096	2	0.9625
bts-rnc	0.2415	3	0.3384
active-dict	0.2144	3	0.2477

And what? The results

ARI on the test data

Dataset	Our ARI	Rank (of 17)	The best ARI
wiki-wiki	0.7096	2	0.9625
bts-rnc	0.2415	3	0.3384
active-dict	0.2144	3	0.2477

- No one achieved decent scores on *bts-rnc* and *active-dict*.

And what? The results

ARI on the test data

Dataset	Our ARI	Rank (of 17)	The best ARI
wiki-wiki	0.7096	2	0.9625
bts-rnc	0.2415	3	0.3384
active-dict	0.2144	3	0.2477

- No one achieved decent scores on *bts-rnc* and *active-dict*.
- Arguably, because of flaws in the gold data:

And what? The results

ARI on the test data

Dataset	Our ARI	Rank (of 17)	The best ARI
wiki-wiki	0.7096	2	0.9625
bts-rnc	0.2415	3	0.3384
active-dict	0.2144	3	0.2477

- No one achieved decent scores on *bts-rnc* and *active-dict*.
- Arguably, because of flaws in the gold data:
  - would be interesting to measure human performance and inter-rater reliability on these datasets;

And what? The results

ARI on the test data

Dataset	Our ARI	Rank (of 17)	The best ARI
wiki-wiki	0.7096	2	0.9625
bts-rnc	0.2415	3	0.3384
active-dict	0.2144	3	0.2477

- No one achieved decent scores on *bts-rnc* and *active-dict*.
- Arguably, because of flaws in the gold data:
  - would be interesting to measure human performance and inter-rater reliability on these datasets;
  - will humans be better than machines?

And what? The results

ARI on the test data

Dataset	Our ARI	Rank (of 17)	The best ARI
wiki-wiki	0.7096	2	0.9625
bts-rnc	0.2415	3	0.3384
active-dict	0.2144	3	0.2477

- No one achieved decent scores on *bts-rnc* and *active-dict*.
- Arguably, because of flaws in the gold data:
  - would be interesting to measure human performance and inter-rater reliability on these datasets;
  - will humans be better than machines?
- Best results for *wiki-wiki* and *bts-rnc* outperform SOTA for English:
  - ARI 0.215-0.286 [Navigli and Vannella, 2013, Bartunov et al., 2016]

And what? The results

ARI on the test data

Dataset	Our ARI	Rank (of 17)	The best ARI
wiki-wiki	0.7096	2	0.9625
bts-rnc	0.2415	3	0.3384
active-dict	0.2144	3	0.2477

- No one achieved decent scores on *bts-rnc* and *active-dict*.
- Arguably, because of flaws in the gold data:
  - would be interesting to measure human performance and inter-rater reliability on these datasets;
  - will humans be better than machines?
- Best results for *wiki-wiki* and *bts-rnc* outperform SOTA for English:
  - ARI 0.215-0.286 [Navigli and Vannella, 2013, Bartunov et al., 2016]
- Probably, RUSSE'18, SemEval-2013, and WWSI datasets are different, but still interesting.

Contents

Summary

- Very naive WSI system making use of pre-trained word embedding models and standard clustering algorithms.

Summary

- Very naive WSI system making use of pre-trained word embedding models and standard clustering algorithms.
- Quite successful for *wiki-wiki*.

Summary

- Very naive WSI system making use of pre-trained word embedding models and standard clustering algorithms.
- Quite successful for *wiki-wiki*.
- Less successful for *bts-rnc* and *active-dict*
  - May be, because of polysemous words with highly inter-related senses.

Summary

- Very naive WSI system making use of pre-trained word embedding models and standard clustering algorithms.
- Quite successful for *wiki-wiki*.
- Less successful for *bts-rnc* and *active-dict*
  - May be, because of polysemous words with highly inter-related senses.
- Word embedding models trained on well-balanced and clean corpora (like RNC) are superior in the extrinsic WSI task to the models trained on large but noisy and unbalanced web or news corpora.

Summary

- Very naive WSI system making use of pre-trained word embedding models and standard clustering algorithms.
- Quite successful for *wiki-wiki*.
- Less successful for *bts-rnc* and *active-dict*
  - May be, because of polysemous words with highly inter-related senses.
- Word embedding models trained on well-balanced and clean corpora (like RNC) are superior in the extrinsic WSI task to the models trained on large but noisy and unbalanced web or news corpora.
- Python source code of the system is available, results are reproducible: `https://github.com/akutuzov/russian_wsi`

Summary

- Very naive WSI system making use of pre-trained word embedding models and standard clustering algorithms.
- Quite successful for *wiki-wiki*.
- Less successful for *bts-rnc* and *active-dict*
  - May be, because of polysemous words with highly inter-related senses.
- Word embedding models trained on well-balanced and clean corpora (like RNC) are superior in the extrinsic WSI task to the models trained on large but noisy and unbalanced web or news corpora.
- Python source code of the system is available, results are reproducible: `https://github.com/akutuzov/russian_wsi`

Thanks to the RUSSE'18 organizers! Questions?

References I

Alagić, D., Šnajder, J., and Padó, S. (2018).
Leveraging lexical substitutes for unsupervised word sense induction.
In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Bar-Hillel, Y. (1964).
Language and information; selected essays on their theory and application.
Addison-Wesley.

Bartunov, S., Kondrashkin, D., Osokin, A., and Vetrov, D. (2016).
Breaking sticks and ambiguities with adaptive skip-gram.
In *Artificial Intelligence and Statistics*, pages 130–138.

References II

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017).
Enriching word vectors with subword information.
Transactions of the Association for Computational Linguistics, 5:135–146.

Jones, K. S. (1964).
Synonymy and semantic classification.
Edinburgh University Press.

Kilgarriff, A. (1997).
I don't believe in word senses.
Computers and the Humanities, 31(2):91–113.

Kutuzov, A., Kopotev, M., Sviridenko, T., and Ivanova, L. (2016).
Clustering comparable corpora of Russian and Ukrainian academic texts: Word embeddings and semantic fingerprints.
In *Ninth Workshop on Building and Using Comparable Corpora*, page 3.

References III

Kutuzov, A. and Kuzmenko, E. (2016).
Webvectors: a toolkit for building web interfaces for vector semantic models.
In *International Conference on Analysis of Images, Social Networks and Texts*, pages 155–161. Springer.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013).
Distributed representations of words and phrases and their compositionality.
Advances in Neural Information Processing Systems 26, pages 3111–3119.

Navigli, R. and Vannella, D. (2013).
Semeval-2013 task 11: Word sense induction and disambiguation within an end-user application.
In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 193–201. Association for Computational Linguistics.

Schutze, H. (1998).
Automatic word sense discrimination.
Computational Linguistics Special-Issue-on-Word Sense Disambiguation, 24(1).

References V

Straka, M. and Straková, J. (2017).
Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe.
In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.