Relationship extraction using word embeddings

Matúš Pikuliak

10.11.2015

Table of content

Distributional hypothesis (Harris 1954)

Linguistic items with similar distributions have similar meanings.

What are word embeddings?

A Neural Probabilistic Language Model [Bengio 2003]
Efficient Estimation of Word Representations in Vector Space [Mikolov 2013]

- Every term has random X-dimensional vector (X = hundreds)
- We are feeding neural network N-grams from text corpus
- We are using NN to tune vectors so that:

$$NN(v_1,v_2,...,v_N) = f(w_1,w_2,...,w_N)$$

- Word w_1 has vector v_1; f() is function of frequency of given N-gram
- In the end NN can test if given N-gram is “valid”, even if given N-gram isn’t in corpus

Features of Word embeddings

- Each term is assigned vector in X-dimensional space
- X is small (hundreds) compared to previous solutions
- Components of vector don’t have explicit meaning

$$v_{word} = [1.555, 2.005, 3.225, ...]$$

Features of our space model

- Semantically similar words have similar vectors

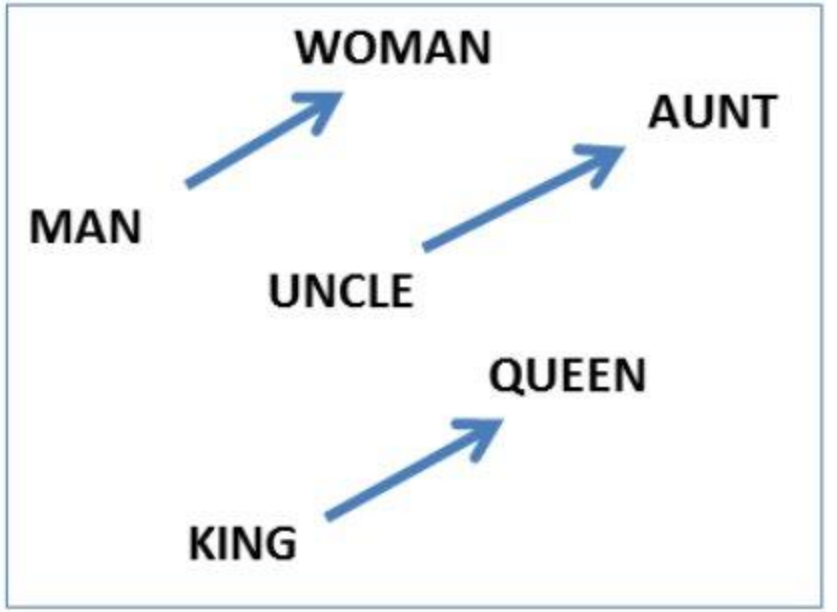
  cos(dog, cat) > cos(dog, airplane)

- Semantically similar relations between words are similar

  cos(Paris - France, Italy - Rome) is high

- these feature were found by observations - they are just byproducts of NN training
- we can use this in analogy retrieval, relation extraction, translation, other NLP tasks

Relational similarity



Our task

- We have class of term pairs with certain semantic relation, eg.

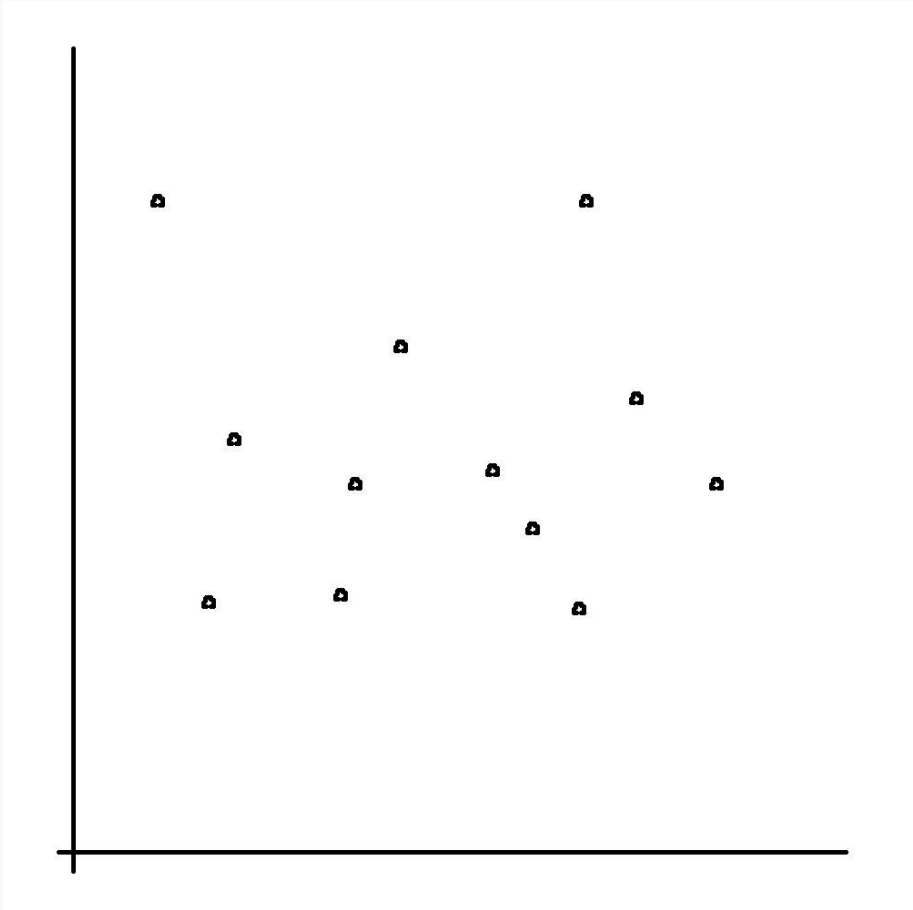
  man-woman, king-queen, uncle-aunt, etc.

- Find new pairs of terms with this relation in text corpus using word embeddings
- We are concerned with precision, not so much with recall

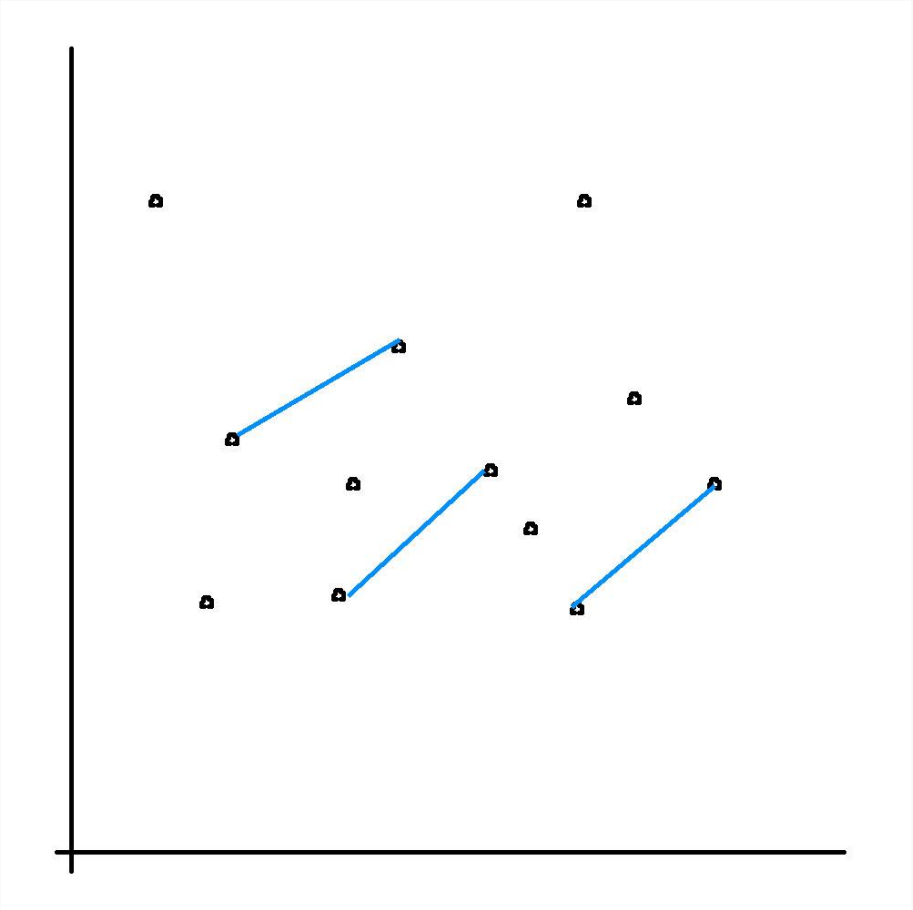
Solution

1. Create word embeddings from text corpus
2. Find vectors of terms in given class
3. Find suitable candidates for classifying
4. Train one-class classifier on given class
5. Classify candidates

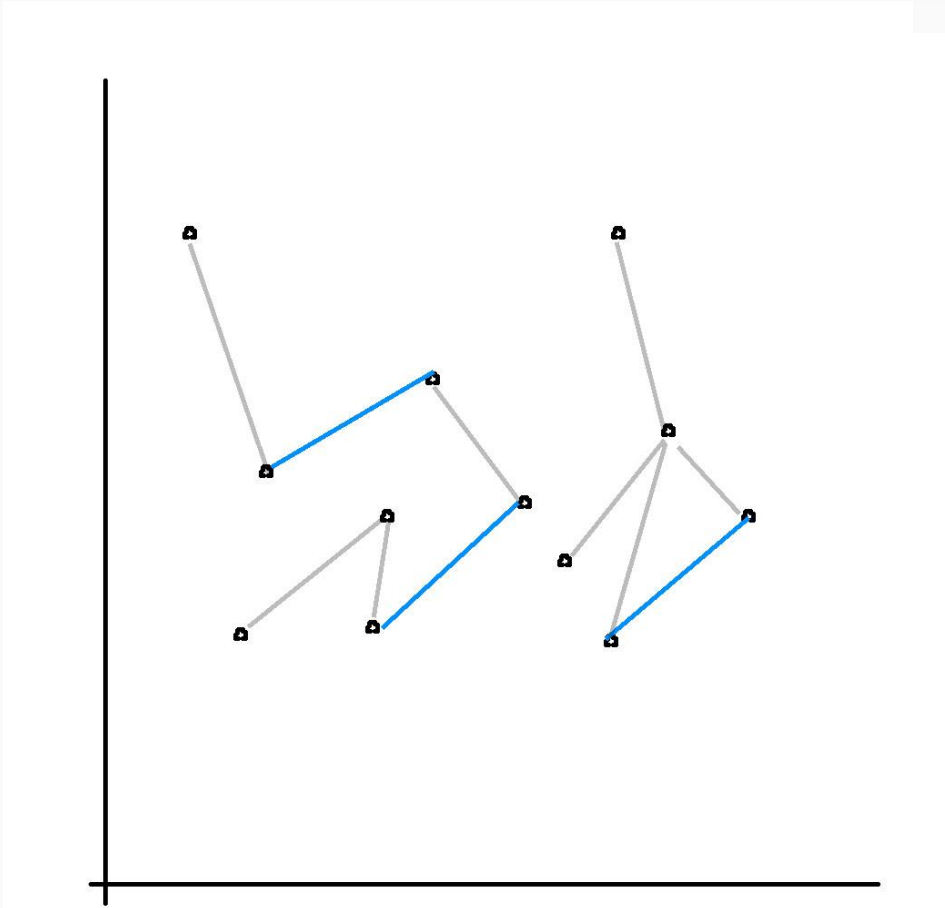
1. Create word embeddings



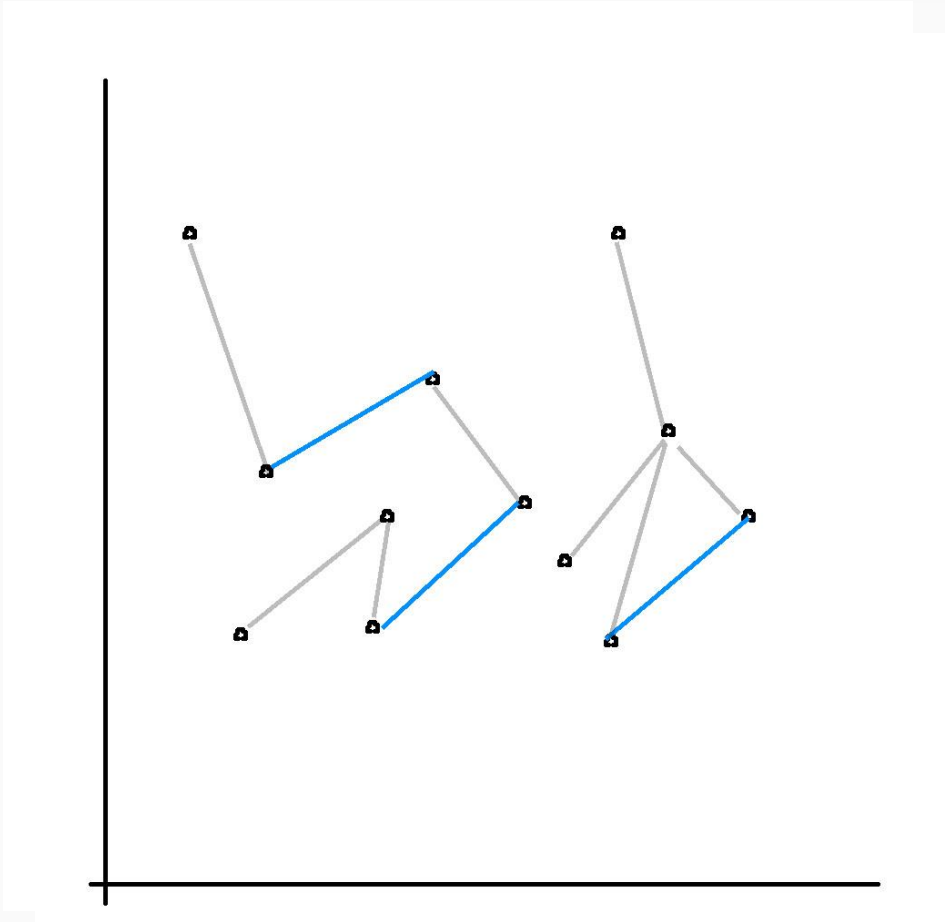
2. Find vectors from given class



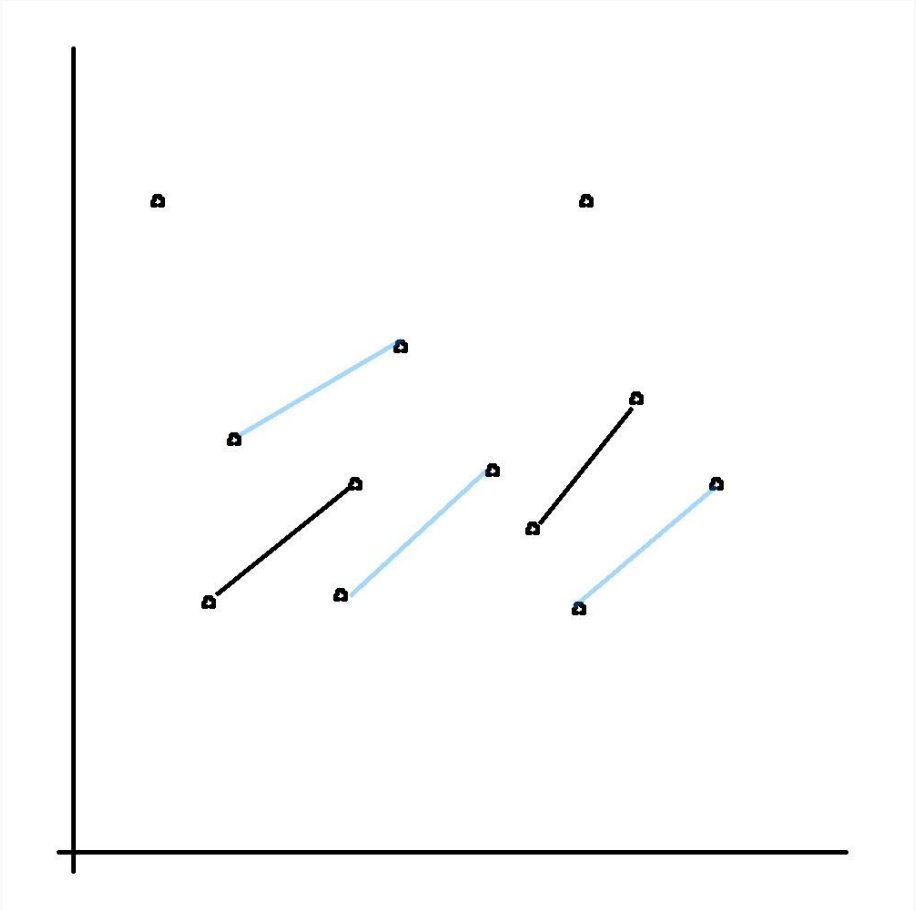
3. Find suitable candidates



4. Train classifier



5. Classify



Solution

1. **Create word embeddings from text corpus**
2. Find vectors of terms in given class
3. Find suitable candidates for classifying
4. Train one-class classifier on given class
5. Classify candidates

1. Create word embeddings from text

- there are more algorithms for creating word embeddings
- all are based on NN, but the NN can vary
- some of them use additional techniques to improve output

We have tested several of resultant embeddings from different algorithms, all trained on the same text corpus. We have find that the original word2spec implementation of Mikolov on BoW5 setting is best performing. So far.

Solution

1. Create word embeddings from text corpus
2. Find vectors of terms in given class
3. **Find suitable candidates for classifying**
4. Train one-class classifier on given class
5. Classify candidates

3. Find suitable candidates

- We need to find pairs of terms, that can have given semantic relation
- We expect that the terms in same position in pairs will be similar

  class: Italy-Rome, France-Paris, England-London

  position 1: Italy, France, England are similar

  position 2: Rome, Paris, London are similar

3. Find suitable candidates

- We can sweep our space for similar terms using simple cosine similarity
- This way we can find suitable terms, eg.

  position 1 - Germany, Spain, …

  position 2 - Madrid, Berlin, …

- Then we can pair them AxB and we will get possible candidates:

  [Germany,Madrid],[Germany,Berlin],[Spain,Madrid],[Spain,Berlin],...

Solution

1. Create word embeddings from text corpus
2. Find vectors of terms in given class
3. Find suitable candidates for classifying
4. **Train one-class classifier on given class**
5. Classify candidates

4. Train one-class classifier

- We have only positive examples
- It is hard to find proper negative set and not just noise
- We are currently testing several algorithms for one-class classification:

  OOSVM, Similarity based classification

- PU Learning - classification with positive and unlabeled data:

  also SVM extension

4. Train one-class classifier

- What should we classify?
  1. relation as vector difference
  2. relation as matrix A where A.a = b
  3. metrics from similarity to training class

Solution

1. Create word embeddings from text corpus
2. Find vectors of terms in given class
3. Find suitable candidates for classifying
4. Train one-class classifier on given class
5. **Classify candidates**

5. Classify candidates

- Precision over recall
- Optimally we would like continuous scale

Outlook

We can find new instances of relations in corpus.

How can we use this method?

- Use ontologies as a source of training data.
- We are planning to use YAGO ontology.
- **Automatically enrich ontologies with new relations.**

Experiments

- We’ve tried simple similarity based one class classification
- We calculate mean similarity between instances of class
- We calculate similarity of classified pair to instances of class
- We compare them, if new pair is above average similarity, it is considered positive instance.

Experiments features

- We are testing on random data (can we filter noise?)
- Recall is quite bad - ~50% of instances are below average similarity
- We are focusing on precision
- We have prepared relations examples (semeval, mikolov dataset)

eg: capital cities, opposites, taxonomy, meronymy, etc

What was tested? Contexts

Bag of words #5 > Bag of words #2

Dependency based are generally worse than BOW5, but in some cases outperforms them

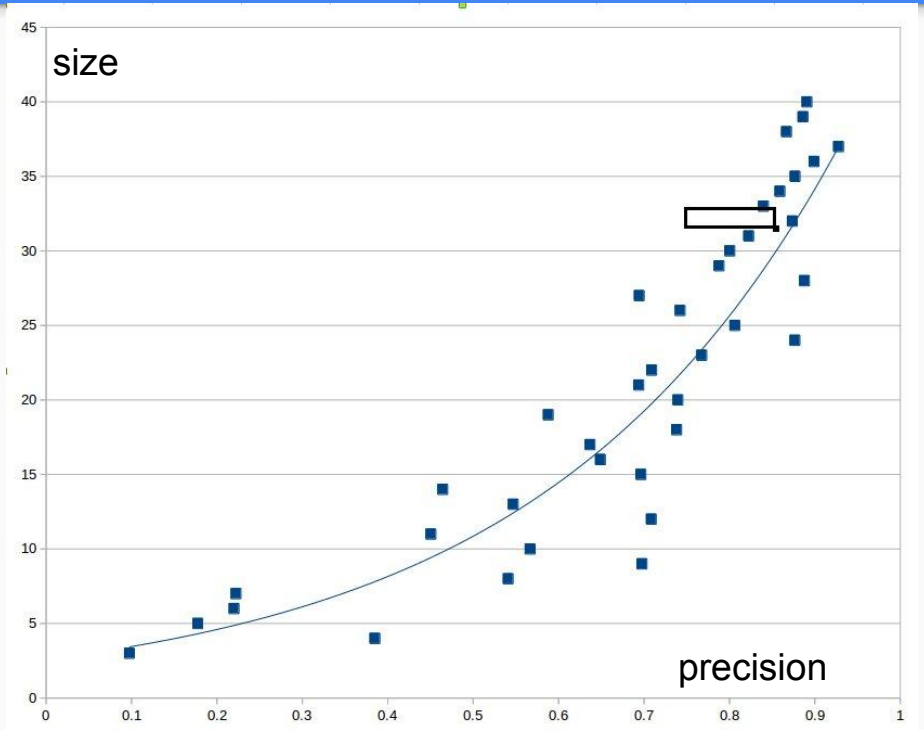
What was tested? Similarity measures

- pairs [*a*,*b*], [*c*,*d*]

- alt 1 - cos(b-a,d-c)
- alt 2 - cos(b-a,d-c).cos(b,d).cos(a,c)

- alt 2 significantly outperforms alt 1

What was tested? Size of exemplary sets



Data

Embeddings:

https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

https://code.google.com/p/word2vec/

Data

Relations classes:

https://github.com/matus-pikuliak/word-embeddings/tree/master/relations

> SemEval 2012 Task 8 Dataset

> Mikolov Analogy Retrieval Dataset