Short Text Similarity with Word Embeddings

CS 6501 Advanced Topics in Information Retrieval @UVa

Tom Kenter[1], Maarten de Rijke[1]

[1]University of Amsterdam, Amsterdam, The Netherlands

Presented by Jibang Wu

Apr 19th, 2017

Outline

Outline

1. Introduction
   - Why Short Text Similarity?
   - How Traditional Approaches Fail?
   - Word Embedding
2. Methodology
   - From Word-level to Text-level Semantics
   - Saliency-weighted Semantic Similarity
   - Learning Algorithm
3. Summary
   - Experiment
   - Analysis
   - Conclusion

Why Short Text Similarity?

Example

– The procedure is generally performed in the second or third trimester.

– The technique is used during the second and, occasionally, third trimester of pregnancy.

Why Short Text Similarity?

Example

– The procedure is generally performed in the second or third trimester.

– The technique is used during the second and, occasionally, third trimester of pregnancy.

- Word-level similarity not enough
  query-query similarity, query-image caption similarity
- Cannot easily go from word-level to text-level similarity
  text structure should be taken into account

Outline

- Lexical Matching
  - Largest common substring, edit distance, lexical overlap

1. United States || United Kingdom
2. United States || USA

- Lexical Matching
  - Largest common substring, edit distance, lexical overlap

    1. United States ‖ United Kingdom
    2. United States ‖ USA

  FAILED: The second one should be better matched

- Linguistic Analysis
  - Parse tree following grammar feature

- Lexical Matching
  – Largest common substring, edit distance, lexical overlap

  1. United States || United Kingdom
  2. United States || USA

  FAILED: The second one should be better matched

- Linguistic Analysis
  – Parse tree following grammar feature

  Not all texts are necessarily parseable (e.g., tweets)
  High-quality parses usually expensive to compute at run time.

- Structured Semantic Knowledge
  – WordNet, Wikipedia

- Lexical Matching
  – Largest common substring, edit distance, lexical overlap

  1. United States || United Kingdom
  2. United States || USA

  FAILED: The second one should be better matched

- Linguistic Analysis
  – Parse tree following grammar feature

  Not all texts are necessarily parseable (e.g., tweets)
  High-quality parses usually expensive to compute at run time.

- Structured Semantic Knowledge
  – WordNet, Wikipedia

  Not available to all language, and domain-specific terms

Outline

1. Introduction
    - Why Short Text Similarity?
    - How Traditional Approaches Fail?
    - Word Embedding
2. Methodology
    - From Word-level to Text-level Semantics
    - Saliency-weighted Semantic Similarity
    - Learning Algorithm
3. Summary
    - Experiment
    - Analysis
    - Conclusion

How do we represent the meaning of a word?

Navies Approach: one-hot representation
store in a vector of vocabulary set size

Example

"hotel" $= [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0 \ldots 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$
"motel" $= [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \ldots 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

Dimensionality: 20K (speech) 500K (dictionary) 13M (Google 1T)

How do we represent the meaning of a word?

Navies Approach: one-hot representation
store in a vector of vocabulary set size

Example

"hotel" $= [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0 \ldots 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$
"motel" $= [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \ldots 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

Dimensionality: 20K (speech) 500K (dictionary) 13M (Google 1T)

Problems:

- Waste of memory
- Hard to show semantic similiarity

How do we represent the meaning of a word?

Word Embedding: distributional similarity based representations

build a dense vector for each word type, chosen so that it is good at predicting other words appearing in its context

Example

"hotel" = [0.286 0.792 -0.177 -0.107 0.109 -0.542 0.349 0.271]
"motel" = [0.280 0.772 -0.171 -0.107 0.109 -0.542 0.349 0.271]

How do we represent the meaning of a word?

Word Embedding: distributional similarity based representations

build a dense vector for each word type, chosen so that it is good at predicting other words appearing in its context

Example

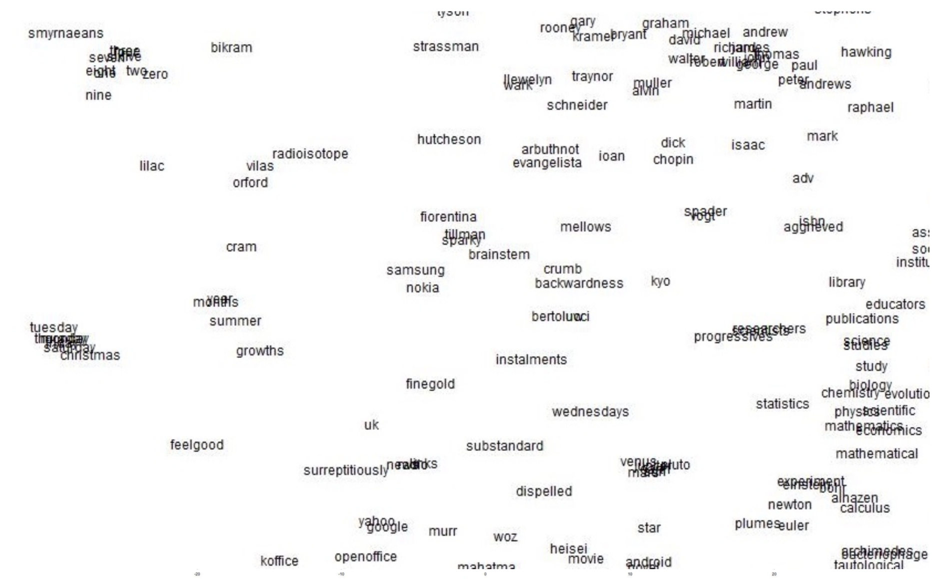
"hotel" = [0.286 0.792 -0.177 -0.107 0.109 -0.542 0.349 0.271]
"motel" = [0.280 0.772 -0.171 -0.107 0.109 -0.542 0.349 0.271]

Dimensionality: 300-500 (Word2vec) 300 (GloVe)
Neural network trained from extensive unlabeled context. [more details]

Advantage:

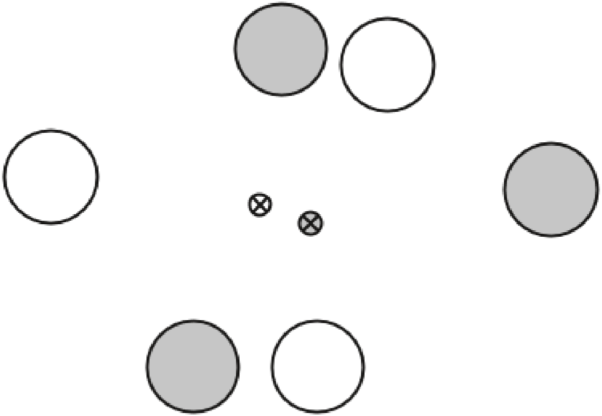
- Efficient in memory and computation
- Easy to show semantic similarity

Intuitions


smyrnaeans
eight
two
zero
seven
nine
bikram
lilac
radioisotope
vilas
orford
cram
summer
tuesday
christmas
growths
feelgood
strassman
hutcheson
fiorentina
brainstem
samsung
nokia
finegold
uk
surreptitiously
substandard
dispelled
yahoo
google
murr
woz
koffice
openoffice
mahatma
heisei
movie
star
android
rooney
gary
kramer
bryant
graham
michael
david
andrew
walter
george
paul
peter
andrews
hawking
llewelyn
traynor
muller
alvin
schneider
martin
raphael
arbuthnot
evangelista
ioan
dick
chopin
isaac
mark
adv
mellows
spader
aggrieved
crumb
backwardness
kyo
library
educators
publications
instalments
progressives
researchers
science
studies
study
biology
chemistry
physics
scientific
mathematics
economics
statistics
wednesdays
mathematical
venus
pluto
experiment
newton
alhazen
calculus
plumes
euler
archimedes
tautological

Outline

Semantic Space



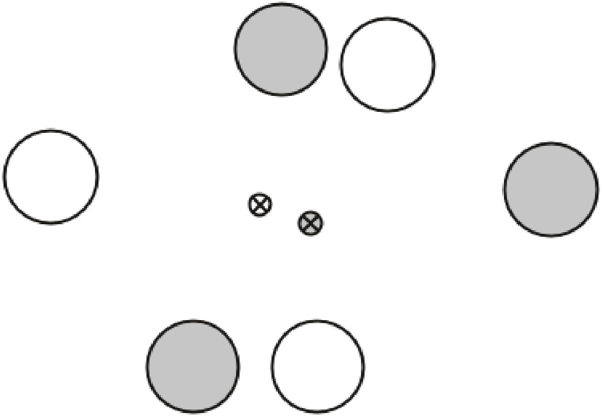
$\bigcirc : w \in S_1$

$\bullet : w \in S_2$

$\otimes : S_1 = \sum_{w \in S_1} w$

$\otimes : S_2 = \sum_{w \in S_2} w$

Semantic Space



○ : $w \in S_1$

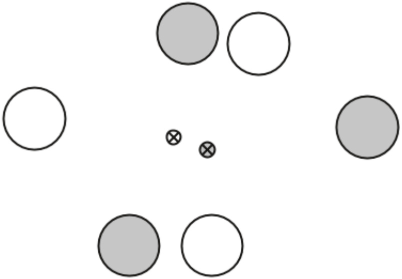
● : $w \in S_2$

$\otimes$: $S_1 = \sum_{w \in S_1} w$

$\otimes$: $S_2 = \sum_{w \in S_2} w$

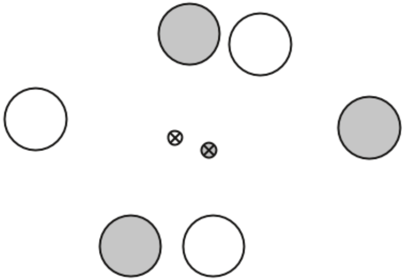
Average sum ?= Sentence similarity

Unweighted Semantic Similarity



1. For each pair of terms (w_1, w_2) in S_1 and S_2, compute the cosine similarities
2. Fully connected, unweighted, bipartite graph
3. Maximum Bipartite Matching
4. Separate the word pairs into bins of different similarity level

Unweighted Semantic Similarity



1. For each pair of terms (w_1, w_2) in S_1 and S_2, compute the cosine similarities
2. Fully connected, unweighted, bipartite graph
3. Maximum Bipartite Matching
4. Separate the word pairs into bins of different similarity level

- Not all terms are equally important
- Longer text has more probability to hit

Outline

1. Introduction
   - Why Short Text Similarity?
   - How Traditional Approaches Fail?
   - Word Embedding
2. Methodology
   - From Word-level to Text-level Semantics
   - Saliency-weighted Semantic Similarity
   - Learning Algorithm
3. Summary
   - Experiment
   - Analysis
   - Conclusion

From BM25

$$r(q, d) = \sum_{w \in q \cap d} IDF(w) \cdot \frac{c(w, d) \cdot (k_1 + 1)}{c(w, d) + k_1 \cdot (1 - b + b \cdot \frac{n}{n_{avg}})}$$

- $c(w, d)$ literal match of words

From BM25

$$f_{sts}(s_l, s_s) = \sum_{w \in s_l} IDF(w) \cdot \frac{sem(w, s_s) \cdot (k_1 + 1)}{sem(w, s_s) + k_1 \cdot (1 - b + b \cdot \frac{|s_s|}{avg_{s_l}})}$$

$$sem(w, s_s) = \max_{w' \in s} f_{sem}(w, w')$$

- $f_{sem}(w, w')$ returns semantic match score from word embedding
- Common words has smaller IDF(w) than rare words.
- Bin summands of different range of score together

Outline

Word Embedding Models

- Pre-trained Out-of-the-Box word embeddings
  - Word2vec 300-dimensions by Mikolov et al.
  - Word2vec 400-dimensions by Baroni et al.
  - GloVe 300-dimensional trained on 840 billion token corpus
  - GloVe 300-dimensional trained on 42 billion token corpus

- Auxiliary word embeddings

  trained on INEX with 1.2 billion tokens

  based either on Word2vec or GloVe Algorithm

  to optimize parameter setting for predicting short text similarity

Binary Classifier from Supervised Learning

Input : List of sentence pairs $((s_{1,1}, s_{1,2}), (s_{2,1}, s_{2,2}), \ldots, (s_{n,1}, s_{n,2}))$

Input : List of associated labels $L = [l_1, l_2, l_3, \ldots, l_n]$

Required: Sets of word embeddings $[WE_1, WE_2, ..., WE_m]$

Required: Multiple feature extractors $[fe_1, fe_2, \ldots, fe_l]$

Output : A trained prediction model M

F = empty feature matrix;

for $i \leftarrow 1$ **to** n **do**

  $\vec{f} = \langle\rangle$;

  for $j \leftarrow 1$ **to** m **do**

    for $k \leftarrow 1$ **to** l **do**

      $\vec{f} = \texttt{concat}(\vec{f}, fe_k((s_{i,1}, s_{i,2}), WE_j))$;

    end

  end

  $F[i] \leftarrow \vec{f}$;

end

$M = trainModel(F, L)$;

Outline

Experiment Setup

- Dataset: Microsoft Research Paraphrase(MSR) Corpus
  5801 sentence pairs annotated with binary labels
  divided into training set of 4076, and testing set of 1725
- Handle Out-of-vocabulary word
  ignore in training, map randomly in runtime
- Paremeter settings
  for f_{sts}, $k_1 = 1.2$, $b = 0.75$, IDF calculated from INEX data

  Three bin threshold:

Similarity level	Highly	Medium	Unlikely
Saliency-weighted Semantic Network	$0 - 0.15$	$0.15 - 0.4$	$0.4 - \infty$
Unweighted Semantic Network	$0 - 0.45$	$0.45 - 0.8$	$0.8 - \infty$

Experiment Results

Baseline methods		Acc.	p	r	F_1
Convolutional NNs [20]		.699	–	–	.809
VSM [38]		.710	.710	.954	.814
Corpus-based PMI [21]		.726	.747	.891	.813
Our method	**Features**	**Acc.**	**p**	**r**	**F_1**
OoB	unwghtd	.746	.768	.882	.822
OoB	unwghtd + swsn	.751	.768	.896	.827
OoB + aux w2v	unwghtd	.754	.770	.897	.829
OoB + aux w2v	unwghtd + swsn	.757	.775	.894	.830
OoB + aux Glv	unwghtd	.756	.774	.894	.830
OoB + aux Glv	unwghtd + swsn	.758	.771	.907	.833
OoB + both aux	unwghtd	.762†	.780†	.893†	.833†
OoB + both aux	unwghtd + swsn	.766†	.781†	.906†	.839†

OoB: out-of-the-box vectors
aux: auxiliary vectors
w2v: Word2vec
glv: GloVe

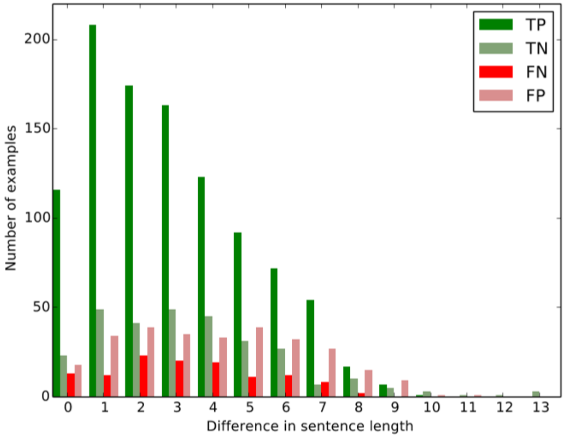
unwghtd: unweighted semantic feature

swsn: saliency-weighted semantic feature

Best model uses all features and word embedding models
The method overall outperform previous approaches

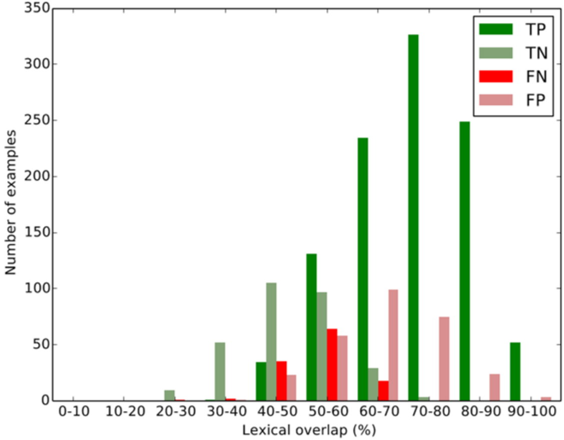
Outline

Performance Across Sentence Length



- Perform better on sentences that are alike in length
- Tend to predict dissimilarity when texts substantially differ in length

Performance Across Levels of Lexical Overlap



- At low lexical overlap level, the algorithm shows the benefit of semantic matching over lexical matching

Outline

Advantages:

- Word embedding based unsupervised learning
- Substitute methods based on external semantic knowledge
- Crucial application in search, query suggestion

Limitations:

- The order of words is not taken into account
- Context awareness is important in real applications

Citation I

Kenter, Tom, and Maarten de Rijke
Short Text Similarity with Word Embeddings.
Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM, 2015.

Mikolov, Tomas, et al.
Distributed representations of words and phrases and their compositionality.
Advances in neural information processing systems. 2013.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning.
Glove: Global Vectors for Word Representation.s.
EMNLP. Vol. 14. 2014.

Outline

4. More
   - Word Embedding

Mainstream Algorithms

- Word2Vec
  predict surrounding words in a window of radius m of every word
  - Continuous bag-of-words (CBOW)
    predicting the word given its context
    several times faster to train than the skip-gram
    slightly better accuracy for the frequent words
  - Skip-gram
    predicting the context given a word
    works well with small amount of the training data
    represents well even rare words or phrases
- Global Vectors for Word Representation (GloVe)
  combines the advantages of global matrix factorization and local context window methods

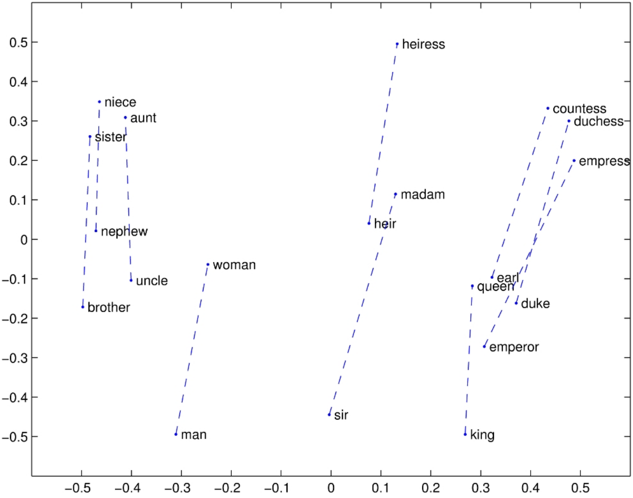
Window based co-occurrence matrix

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Slide from Stanford CS224n

Feature Highlights



$$W(\text{"woman"}) - W(\text{"man"}) \simeq W(\text{"aunt"}) - W(\text{"uncle"})$$