How much meaning can you pack into a real-valued vector? Semantic similarity measuring using recursive auto-encoders

Wojciech Walczak

Samsung R&D Institute Poland, 2016

SAMSUNG

Agenda

- Why are we here?
- Why is paraphrase detection important?
- Why is NLP hard?
- Can word embeddings save the day?
- How to aggregate word embeddings?
- SemEval contest

Why are we here?

- Our team won the Semantic Textual Similarity task within the SemEval 2016 contest.

  The aim: recognizing paraphrases among pairs of sentences.

- Not only pictures!

  Most PyData Warsaw's talks focus on processing images.
  How about processing some textual data?

Why is paraphrase detection important?

Lots of practical, industrial applications:

- **Question answering** (questions matching in customer service)
- **Plagiarism detection** (are certain paragraphs too similar?)
- **Information retrieval** (is this query similar to other queries?)

...and many more!

Why is NLP hard?

- **Language is ambiguous:**
  - Syntactic ambiguity (e.g. *John saw the man on the mountain with a telescope*)
  - Polysemous words (e.g. *mouse* – an animal or a device?)
- **Single sentences can be complex** (hard to parse).
- **Multiple sentences are even more complex** (boundary detection):

  ...in the US. Govt. ...
- **Named entities may be hard to recognize:**

  Washington – place or person?

  May – person or month?

Why is NLP hard? #2

- **Context is important:**
  - Local standards, e.g. *How far is it?* (miles, km?)
  - Social context: *That was bad!* (reprimand to a kid? kudos to a friend?)
- **Spelling mistakes:**

  TOP 10 word embeddings close to *galaxy* are: *galexy*, *galxy*, *galazy*, *glaxy*, *gallaxy*, *galasy*, *sg*, *galaxys*, *glalaxy*, *gal*
- **Slang, jargon, abbreviations:**

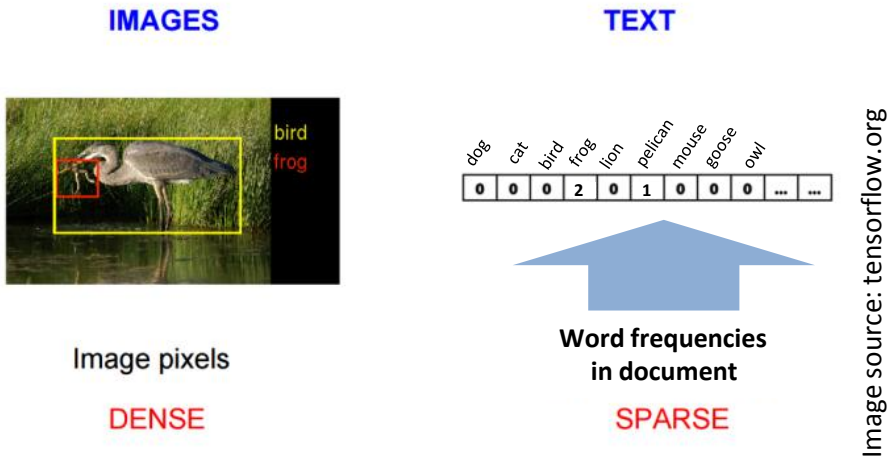
  Example user question: *was link up lte but i cnt use d internet in the least!!!!!!!*

Most of these issues come up when detecting paraphrases.

Can embeddings save the day?

- Image processing: rich, high-dimensional data encoded as vectors.
- Language processing: sparse data, words as discrete atomic symbols.

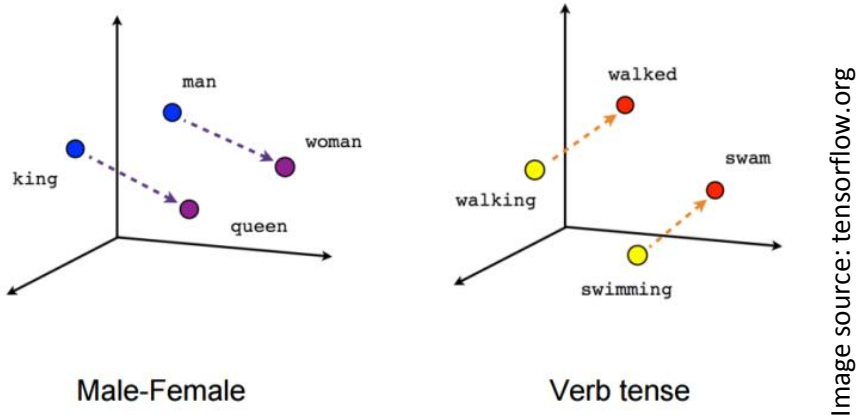
  No information regarding relationships between words.



Can embeddings save the day? #2

- Word embedding models represent words in a continuous vector space.
- Semantically similar words are mapped to nearby points.

  The words are **embedded** nearby each other.
  Examples: word2vec, GloVe.



Can embeddings save the day? #3

```
>>> from gensim.models.word2vec import Word2Vec

>>> embeddings = Word2Vec.load_word2vec_format('word_vectors.txt', binary=False)

>>> embeddings['vehicle'][:10] # 10 values of a vector of dimensionality 50
array([-0.756091. -1.01268494,  2.04105091,  2.43842196,  2.95695996,
       -0.33063, -1.34891498, -0.251019,  2.78287601,  0.55933303], dtype=float32)

>>> embeddings.similarity('vehicle', 'car') # cosine similarity between two vectors
0.787731
```

Auto-encoders

- Can we aggregate collections of word embeddings?
- We can encode word embeddings into a single vector using auto-encoders.

  Unsupervised (self-supervised) learning algorithm.

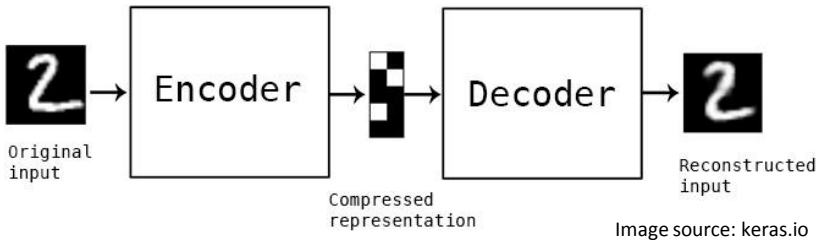
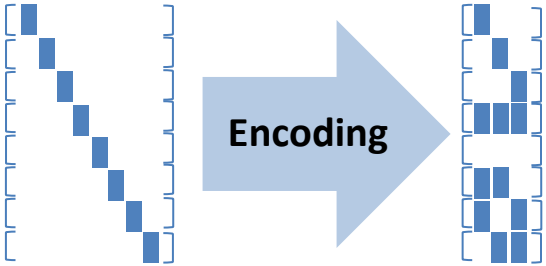
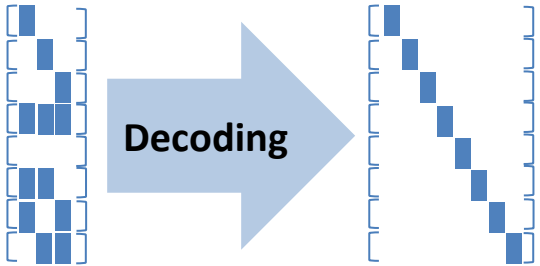


Image source: keras.io

Auto-encoders: simple example





Input:
1-hot vectors
of length 8

Learned representation:
int-valued vectors of
length 3

Input:
int-valued vectors of
length 3

Output:
1-hot vectors
of length 8

Auto-encoders: network

```
input_size, hidden_size = 8, 3

X = tf.placeholder(tf.float32, [8, input_size])
```

X is a placeholder for the input data (here: 8x8 matrix of 1-hot vectors)

```
W_input_to_hidden = tf.Variable(tf.truncated_normal([input_size, hidden_size]))

bias_hidden = tf.Variable(tf.truncated_normal([hidden_size]))
```

Weights and bias for hidden layer

```
W_hidden_to_output = tf.Variable(tf.truncated_normal([hidden_size, input_size]))

bias_output = tf.Variable(tf.truncated_normal([input_size]))
```

Weights and bias for output layer

```
hidden = tf.nn.sigmoid(tf.nn.xw_plus_b(X, W_input_to_hidden, b_hidden))

output = tf.nn.softmax(tf.nn.xw_plus_b(hidden, W_hidden_to_output, b_output))
```

Input to hidden + sigmoid
Hidden to output + softmax

```
error = tf.sqrt(tf.reduce_mean(tf.square(X - output)))

train_op = tf.train.AdamOptimizer(learning_rate=0.01).minimize(error)
```

Define mean squared error
Optimize the error

Auto-encoders: training and usage

```
eye = np.eye(8, dtype=np.float32)
```

8x8 1-hot matrix

```
[ 1.  0.  0.  0.  0.  0.  0.  0.]
[ 0.  1.  0.  0.  0.  0.  0.  0.]
[ 0.  0.  1.  0.  0.  0.  0.  0.]
[ 0.  0.  0.  1.  0.  0.  0.  0.]
[ 0.  0.  0.  0.  1.  0.  0.  0.]
[ 0.  0.  0.  0.  0.  1.  0.  0.]
[ 0.  0.  0.  0.  0.  0.  1.  0.]
[ 0.  0.  0.  0.  0.  0.  0.  1.]
```

```
with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    for i in range(50000):
        cur_eye = sorted(eye, key=lambda k: random.random())
        sess.run([train_op], feed_dict={X: cur_eye})
```

TRAINING

```
inputs = sess.run([hidden], feed_dict={X: eye})[0]
for orig, encoded in zip(eye, inputs):
    print('{} => {}'.format(orig, encoded))
```

ENCODE

```
[ 1.  0.  0.  0.  0.  0.  0.  0.] => [ 1.  1.  1.]
[ 0.  1.  0.  0.  0.  0.  0.  0.] => [ 0.  1.  0.]
[ 0.  0.  1.  0.  0.  0.  0.  0.] => [ 1.  1.  0.]
[ 0.  0.  0.  1.  0.  0.  0.  0.] => [ 1.  0.  0.]
[ 0.  0.  0.  0.  1.  0.  0.  0.] => [ 0.  0.  1.]
[ 0.  0.  0.  0.  0.  1.  0.  0.] => [ 1.  0.  1.]
[ 0.  0.  0.  0.  0.  0.  1.  0.] => [ 0.  0.  0.]
[ 0.  0.  0.  0.  0.  0.  0.  1.] => [ 0.  1.  1.]
```

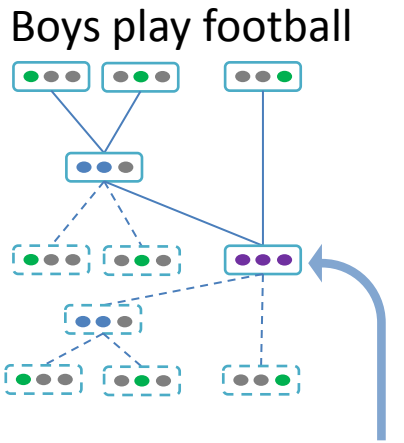
```
outputs = sess.run([output], feed_dict={hidden: np.array(inputs)})[0]
for encoded, decoded in zip(inputs, outputs):
    print('{} => {}'.format(encoded, decoded))
```

DECODE

```
[ 1.  1.  1.] => [ 1.  0.  0.  0.  0.  0.  0.  0.]
[ 0.  1.  0.] => [ 0.  1.  0.  0.  0.  0.  0.  0.]
[ 1.  1.  0.] => [ 0.  0.  1.  0.  0.  0.  0.  0.]
[ 1.  0.  0.] => [ 0.  0.  0.  1.  0.  0.  0.  0.]
[ 0.  0.  1.] => [ 0.  0.  0.  0.  1.  0.  0.  0.]
[ 1.  0.  1.] => [ 0.  0.  0.  0.  0.  1.  0.  0.]
[ 0.  0.  0.] => [ 0.  0.  0.  0.  0.  0.  1.  0.]
[ 0.  1.  1.] => [ 0.  0.  0.  0.  0.  0.  0.  1.]
```

Recursive Auto-Encoders (RAE)

- Real-life scenarios aren't that easy.
- Instead of simple 1-hot vectors, we work with parse trees and word embeddings.
- Tree structures can be encoded using Recursive Auto-Encoders.



The final vector represents the encoded sentence

The boxes represent word embeddings (vectors). The dimensionality is usually 50 or more

The word vectors are recursively encoded in order resembling the parse tree

The dashed boxes represent decoded word vectors used to count reconstruction error during training

The intermediate vectors are unfolded until word vectors are decoded (it helps avoid propagating errors of intermediate nodes)

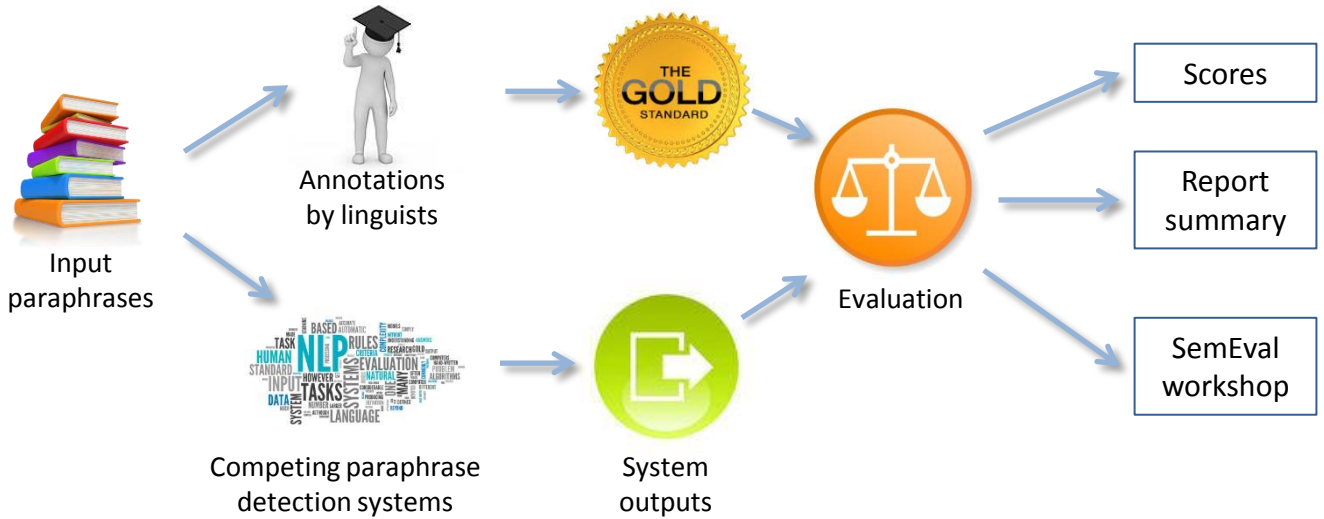
What is SemEval?

- SemEval (Semantic Evaluation) is an ongoing series of evaluations of computational semantic analysis systems.
- Umbrella organization: SIGLEX, a „Special Interest Group on the Lexicon” of the Association for Computational Linguistics.

Competition's tasks

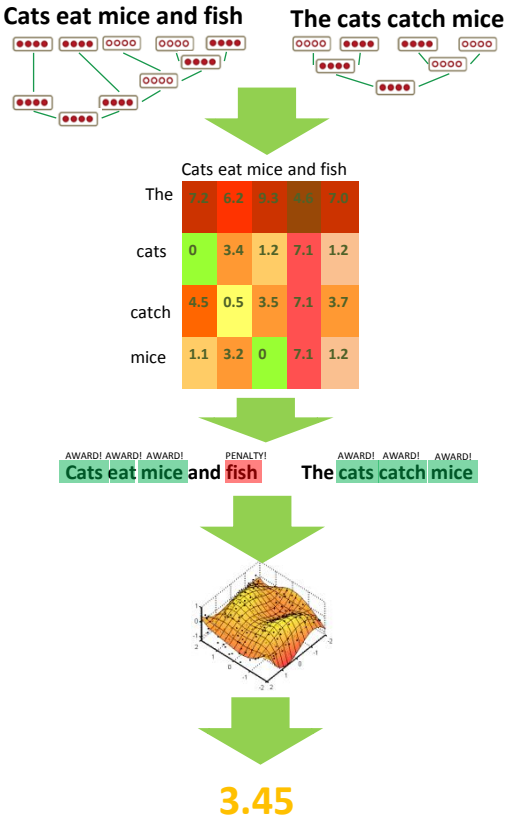
Track I. Textual Similarity and Question Answering Track
Task 1: Semantic Textual Similarity: A Unified Framework for Semantic Processing and Evaluation
Task 2: Interpretable Semantic Textual Similarity
Task 3: Community Question Answering

Track II. Sentiment Analysis Track
...
Track III. Semantic Parsing Track
...
Track IV. Semantic Analysis Track
...
Track V. Semantic Taxonomy Track
...



SemEval: our solution (basic)

Are two sentences similar?



A working evaluation tool must be able to detect whether two sentences have the same meaning.

The Recursive Auto Encoder encodes word embeddings into aggregated vectors.

A distance matrix is computed to generate similarity scores for two sentences.
The similarity scores are also counted for subtrees (not shown on the slide).

The WordNet-based module makes adjustments to the distances between words:
- awarding pairs of words with positive semantic similarity;
- penalizing out-of-context words and disjoint similar concepts.

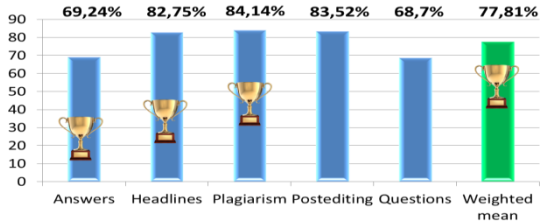
The WordNet-adjusted similarity matrices are converted to a matrix suitable for the Linear Support Vector Regression. The SVR model generates the final result.

The STS competition aimed at evaluating the sentences on the scale of 0 to 5, where 5 means perfect paraphrase. The score of 3.45 means that the sentences have a lot in common, but aren't an exact match.

SemEval results

Companies:

- Toyota Technological Institute
- RICOH
- Mayo Clinic
- IHS Markit



Public research institutions:

- German Research Center for Artificial Intelligence, Germany
- National Centre for Text Mining, UK
- Institute of Software, Chinese Academy of Sciences, China

Universities:

- University of Colorado Boulder, USA
- University of Texas, Arlington, USA
- University of Sheffield, UK
- University of Sussex, UK
- Universität des Saarlandes, Germany
- Heinrich Heine University Düsseldorf, Germany
- University of Madrid, Spain
- Dublin City University, Ireland
- Beijing Institute of Technology, China

...and others!

Top 10 results during SemEval 2016

Place	Team	Overall mean
1	**Samsung R&D Poland: ensemble 1**	77.8%
2	University of West Bohemia, Czech Republic	75.7%
3	Mayo Clinic, USA	75.6%
4	**Samsung R&D Poland: ensemble 2**	75.4%
5	East China Normal University, China	75.1%
6	The National Centre for Text Mining, UK	74.8%
7	Univeristy of Maryland, USA Toyota Technological Institute, USA University of Waterloo, Canada	74.2%
8	University of Massachusetts Lowell, USA	73.8%
9	Mayo Clinic, USA	73.569%
10	**Samsung R&D Poland: basic solution**	73.566%

...total of **40** teams and **113** runs

More on our solution

- **”Samsung Poland NLP Team at SemEval-2016 Task 1: Necessity for diversity; combining recursive autoencoders, WordNet and ensemble methods to measure semantic similarity”,** Barbara Rychalska, Katarzyna Pakulska, Krystyna Chodorowska, Wojciech Walczak and Piotr Andruszkiewicz

- **”Paraphrase Detection Ensemble – SemEval 2016 winner”**, Katarzyna Pakulska, Barbara Rychalska, Krystyna Chodorowska, Wojciech Walczak, Piotr Andruszkiewicz, IPI PAN seminar (10 October 2016), PDF available at: http://zil.ipipan.waw.pl/seminar

Thank you!

Questions?

Appendix

Further reading

1. **”Samsung Poland NLP Team at SemEval-2016 Task 1: Necessity for diversity; combining recursive autoencoders, WordNet and ensemble methods to measure semantic similarity”,** Barbara Rychalska, Katarzyna Pakulska, Krystyna Chodorowska, Wojciech Walczak and Piotr Andruszkiewicz
2. **”Paraphrase Detection Ensemble – SemEval 2016 winner”**, Katarzyna Pakulska, Barbara Rychalska, Krystyna Chodorowska, Wojciech Walczak, Piotr Andruszkiewicz, IPI PAN seminar (10 October 2016), PDF available at: http://zil.ipipan.waw.pl/seminar
3. **”Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection”** Richard Socher and Eric H. Huang and Jeffrey Pennington and Andrew Y. Ng and Christopher D. Manning
4. **”Grounded Compositional Semantics For Finding And Describing Images With Sentences”**, Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, Andrew Y. Ng.
5. **”Semantic Textual Similarity Systems”**, Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield and Jonathan Weese

Further reading

6. **”WordNet: Similarity - Measuring the Relatedness of Concepts”**, Ted Pedersen, Siddharth Patwardhan, Jason Michelizzi
7. **”WordNet: A Lexical Database for English”**, George A. Miller (1995). Communications of the ACM Vol. 38, No. 11: 39-41.
8. **”WordNet: An Electronic Lexical Database”**, Christiane Fellbaum (1998, ed.) . Cambridge, MA: MIT Press.
9. **”Samsung: Align-and-Differentiate Approach to Semantic Textual Similarity”**, Lushan Han, Justin Martineau, Doreen Cheng, Christopher Thomas
8. **”DLS@CU: Sentence Similarity from Word Alignment”** Arafat Sultan, Steven Bethard, Tamara Sumner
9. **”Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks”** Kai Sheng Tai, Richard Socher, Christopher D. Manning
10. **”ExB Themis: Extensive Feature Extraction from Word Alignments for Semantic Textual Similarity”** Christian Hanig, Robert Remus, Xose De La Puente