Continuous Representations of Words

a.k.a. word vectors
a.k.a. word embeddings
a.k.a. projection layers

Jon Dehdari

December 7, 2015

Good Morning!

Words as Integers

- Our previous representations of words (and word classes) have been fairly flat
- For example, the word '*monkey*' can be represented as an integer, such as '7'

Words as Integers

- Our previous representations of words (and word classes) have been fairly flat
- For example, the word '*monkey*' can be represented as an integer, such as '7'
- **One-hot encoding** represents that as:

0	0	0	0	0	0	1	0	0	0	...	0

Words as Integers

- Our previous representations of words (and word classes) have been fairly flat
- For example, the word '*monkey*' can be represented as an integer, such as '7'
- **One-hot encoding** represents that as:

0	0	0	0	0	0	1	0	0	0	...	0

- and the word class (eg. 2) containing '*monkey*':

0	1	0	0

Words as Integers

- Our previous representations of words (and word classes) have been fairly flat
- For example, the word '*monkey*' can be represented as an integer, such as '7'
- **One-hot encoding** represents that as:

0	0	0	0	0	0	1	0	0	0	...	0

- and the word class (eg. 2) containing '*monkey*':

0	1	0	0

- Both of these are sparse vectors of booleans, with just one entry having a 'true' value

Words as Integers

- Our previous representations of words (and word classes) have been fairly flat
- For example, the word '*monkey*' can be represented as an integer, such as '7'
- **One-hot encoding** represents that as:

0	0	0	0	0	0	1	0	0	0	...	0

- and the word class (eg. 2) containing '*monkey*':

0	1	0	0

- Both of these are sparse vectors of booleans, with just one entry having a 'true' value
- Either way, we're working with integers (..., -2, -1, 0, 1, 2, ...)

Words as $\mathbb{R}$eal Numbers

- We can do more with real numbers (eg. -1.5, 0.23, 55.01)

Words as $\mathbb{R}$eal Numbers

- We can do more with real numbers (eg. -1.5, 0.23, 55.01)
- We can represent the word '*monkey*' as a dense vector of real numbers:

0.38	-1.27	-0.55	1.44

Words as $\mathbb{R}$eal Numbers

- We can do more with real numbers (eg. -1.5, 0.23, 55.01)
- We can represent the word '*monkey*' as a dense vector of real numbers:

0.38	-1.27	-0.55	1.44

- We can have the plural form, '*monkeys*' be close in that vector space:

0.31	-1.27	**-0.61**	1.44

Words as $\mathbb{R}$eal Numbers

- We can do more with real numbers (eg. -1.5, 0.23, 55.01)
- We can represent the word '*monkey*' as a dense vector of real numbers:

0.38	-1.27	-0.55	1.44

- We can have the plural form, '*monkeys*' be close in that vector space:

0.31	-1.27	**-0.61**	1.44

- We can also have a related word, like '*ape*' be close in that vector space, *but in different dimensions*:

0.38	**-1.33**	-0.55	**1.49**

Applications of Word Vectors

- **Word distances**. For example, closest words to '*Sweden*':

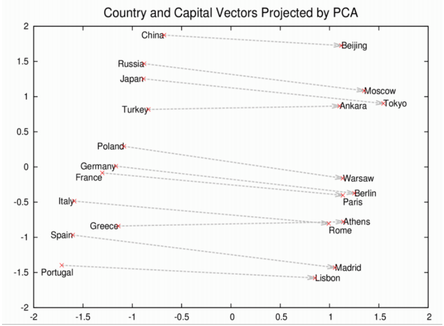
Word	Cosine Distance
Norway	0.75
Denmark	0.72
Finland	0.62
Switzerland	0.59
...	

Applications of Word Vectors

- **Word distances**. For example, closest words to '*Sweden*':

Word	Cosine Distance
Norway	0.75
Denmark	0.72
Finland	0.62
Switzerland	0.59
...	

- **Analogy**. E.g., *Japan* is to *Tokyo* as *Germany* is to *Berlin*

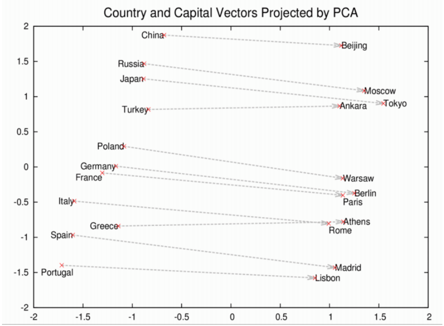


Applications of Word Vectors

- **Word distances**. For example, closest words to '*Sweden*':

Word	Cosine Distance
Norway	0.75
Denmark	0.72
Finland	0.62
Switzerland	0.59
...	

- **Analogy**. E.g., *Japan* is to *Tokyo* as *Germany* is to *Berlin*



Japan – Tokyo ≈ Germany – Berlin

Applications of Word Vectors

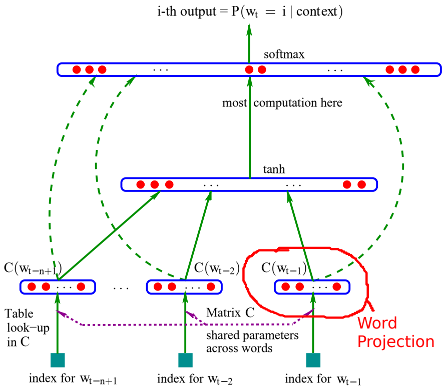
- **Sentence Completion** (actually just restricted language modeling):
- "All red-headed men who are above the age of [800 | seven | twenty-one | 1,200 | 60,000] years , are eligible."
- "That is his [generous | mother's | successful | favorite | main] fault , but on the whole he's a good worker."

Applications of Word Vectors

- **Sentence Completion** (actually just restricted language modeling):
- "All red-headed men who are above the age of [800 | seven | twenty-one | 1,200 | 60,000] years , are eligible."
- "That is his [generous | mother's | successful | favorite | main] fault , but on the whole he's a good worker."
- Mikolov et al (2013b) selected the test word that best predicted the context

Projection Layer in Neural Language Models

- **Neural Language Modeling** – this was actually one of the earliest uses of word vectors. We'll talk more about these later this semester



word2vec

- Tomáš Mikolov and colleagues found that you don't need the full neural-net language model to get useful word vectors

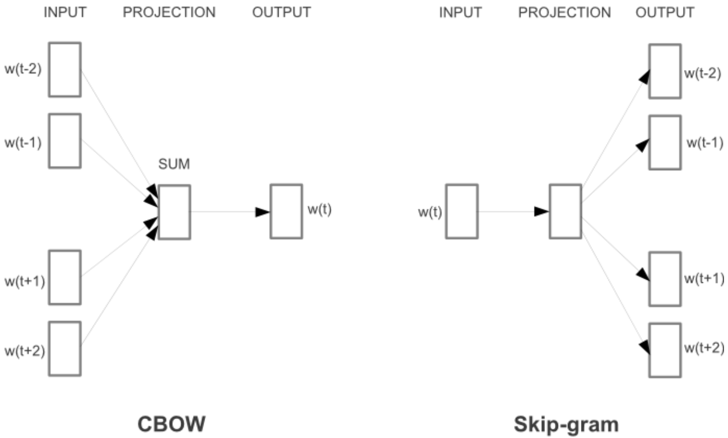
word2vec

- Tomáš Mikolov and colleagues found that you don't need the full neural-net language model to get useful word vectors
- In fact, you don't need a neural network at all. He removed the hidden layer, giving a traditional logistic regression model

word2vec

- Tomáš Mikolov and colleagues found that you don't need the full neural-net language model to get useful word vectors
- In fact, you don't need a neural network at all. He removed the hidden layer, giving a traditional logistic regression model
- He developed a simplified form of training called negative sampling (derived from earlier NCE). It's a little like a binary MaxEnt classifier

word2vec: CBOW & Skip-gram

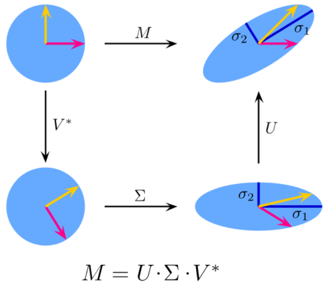


Hyperparameters

- Window size: how much surrounding context to use
- Normalization: softmax (traditional) vs. hierarchical softmax vs. negative sampling
- Vector dimensions: 100–500 common
- Number of negative samples: 3–10 common
- Number of training epochs, initial learning rate, negative sample distribution ($\alpha = 0.75$), model, . . .

Matrix Factorization of Count Co-Occurrences

- Glove and Latent Semantic Analysis (LSA) count the co-occurrences of word pairs, then use matrix factorization techniques like singular value decomposition (SVD) for dimensionality reduction of this original matrix

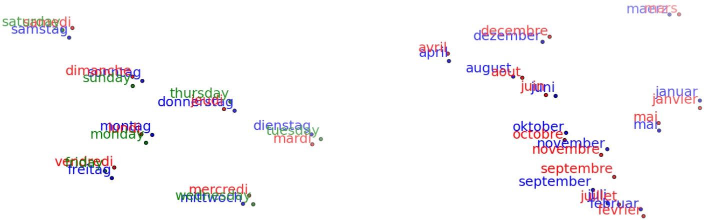


Unifying these Approaches

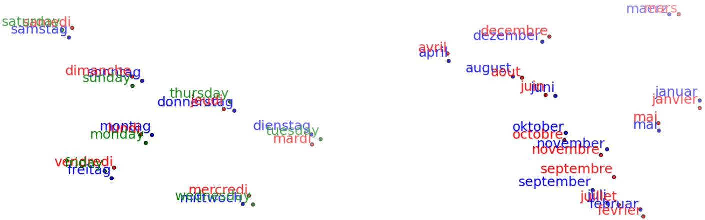
- Word2vec, Glove, and LSA all do matrix factorization (Levy & Goldberg, 2014), but the successful ones are weighted for word frequency
- Pointwise Mutual Information (PMI) is (implicitly) used by these:

$$\mathrm{PMI}(x, y) = log \frac{P(x, y)}{P(x)\, P(y)}$$

Bilingual Word Vectors



Bilingual Word Vectors



Monolingual objective: maximize likelihood of training set, where $P(w|c) = \sigma(\mathbf{w} \cdot \mathbf{c})$

Multilingual objective: maximize likelihood of both sentence-aligned training sets (s & t), based on:
$\sigma(\mathbf{w_t} \cdot \mathbf{c_t}) + \sigma(\mathbf{w_t} \cdot \mathbf{c_s}) + \sigma(\mathbf{w_s} \cdot \mathbf{c_s}) + \sigma(\mathbf{w_s} \cdot \mathbf{c_t})$

Courtesy of Hermann & Blunsom (2013)

Bilingual Word Vectors Comparison

Method	No word alignments required	No prior on the mapping between target vectors	No explicit alignments of target vectors	Computationally efficient	Can leverage monolingual corpus	Free software
Klementiev et al (2012)	✓	×	✓	×	✓	×
BiCVM	✓	✓	×	✓	×	✓
Bilingual autoencoders	✓	✓	×	×	×	✓
BilBOWA	✓	✓	×	✓	✓	✓
Trans-gram	✓	✓	✓	✓	✓	×

Courtesy of Coulmance et al. (2015)

Try Them Out!

- Original word2vec code: https://code.google.com/p/word2vec/ – includes nice illustrations
- Python version: Gensim
- Java version in DL4J
- Glove