Word vectors of various kinds

Jon Dehdari and **Asad Sayeed**

December 14, 2016

Jorge Luis Borges

An Argentinian philosopher and fiction writer. One of his stories mentions 'a certain Chinese Encyclopedia', the *Celestial Emporium of Benevolent knowledge.* It contains a classifcation of animals.

- those that belong to the emperor
- embalmed ones
- those that are trained
- suckling pigs
- mermaids
- fabulous ones
- stray dogs

Jorge Luis Borges

. . . actually, it goes on.

- those that are included in the present classification
- those that tremble as if they are mad
- innumerable ones
- those drawn with a very fine camelhair brush
- others
- those that have just broken a flower vase
- those that from a long way off look like flies

What words are

So far we've talked about words in order. But words have a relationship to each other.

- We use dictionaries in real life for a reason.
- We need to make fine-grained distinctions, draw connections, and so on.
- Humans make judgements about similarities.
    - You know that "motorcycle" can be used in most, but not all contexts that "car" can be used.
    - English-German bilinguals know that "pride" and "Stolz" are quite similar.

Define "chair"

From dictionary.com (just the noun version):

Define “chair”

From dictionary.com (just the noun version):

- A seat, especially for one person, usually having four legs for support and a rest for the back and often having rests for the arms.
- Something that serves as a chair or supports like a chair: “two men clasped hands to make a chair for their injured companion”.
- A position of authority, as of a judge, professor, etc.
- The person occupying a seat of office, especially the chairperson of a meeting: “the speaker addressed the chair”
- (in an orchestra) the position of a player, assigned by rank; desk: “first clarinet chair”.
- “the chair”, Informal. electric chair.

Words in terms of other words

That doesn't seem very helpful, but it gives us a place to start.
Define "chair" in terms of features:

- +one-person, +four-legs, +support, +backrest, +armrest
- +authority
- +occupies-chair
- +orchestra
- +execution

Words in terms of other words

OK, that gives us the definition of a chair in terms of (rather specific) features.

Define the noun "cockpit". Let's go to dictionary.com again. I get as features:

- +enclosed, +airplane, +controls, +panel, +seats
- +instrumentation, +automobile
- +pit, +cockfights
- +conflict

Very little overlaps.

So can we compare them?

Encode features as 1 or 0

	chair	cockpit
one-person	1	0
backrest	1	0?
four-legs	1	0
support	1	0?
armrest	1	0?
authority	1	0?
enclosed	0	1
airplane	0	1
seats	0?	1
...		

Similarity

- What we've just defined is a vector space.

Similarity

- What we've just defined is a vector space.
- Dimension = feature. So far it's a low-dimensional space.

Similarity

- What we've just defined is a vector space.
- Dimension = feature. So far it's a low-dimensional space.
- How can we measure the similarity between them? Common answer: cosine similarity.

Similarity

- What we've just defined is a vector space.
- Dimension = feature. So far it's a low-dimensional space.
- How can we measure the similarity between them? Common answer: cosine similarity.
- So what would the similarity of "chair" and "cockpit" be in our space? Probably zero!

Words in terms of other words

We need a new data source. Collect it from a real corpus. Let's try Google.

Words in terms of other words

We need a new data source. Collect it from a real corpus. Let's try Google.

	chair	cockpit
one-person		
backrest		
four-legs		
support		
armrest		
authority		
enclosed		
airplane		
seats		
...		

Words in terms of other words

We need a new data source. Collect it from a real corpus. Let's try Google.

	chair	cockpit
one-person		
backrest		
four-legs		
support		
armrest		
authority		
enclosed		
airplane		
seats		
...		

Now it's not so bad: we can get a non-zero similarity. Yay?

Words in terms of other words

- In fact, rather than using dictionary definitions of explicit features, cut out the middle man.

Words in terms of other words

- In fact, rather than using dictionary definitions of explicit features, cut out the middle man.
- "Learn" a vector for each word by counting corpus context. Ways of learning:
  - Simple co-occurrence counts based on a window.
    - The vocabulary basically becomes the feature space.

Words in terms of other words

- In fact, rather than using dictionary definitions of explicit features, cut out the middle man.
- "Learn" a vector for each word by counting corpus context. Ways of learning:
  - Simple co-occurrence counts based on a window.
    - The vocabulary basically becomes the feature space.
  - More complex counts, such as POS tags, bits of parse trees.

Words in terms of other words

- In fact, rather than using dictionary definitions of explicit features, cut out the middle man.
- "Learn" a vector for each word by counting corpus context. Ways of learning:
  - Simple co-occurrence counts based on a window.
    - The vocabulary basically becomes the feature space.
  - More complex counts, such as POS tags, bits of parse trees.
- Sometimes raw counts aren't what you need: smoothing, reweighting.

Words in terms of other words

These are "count" vectors. What are the problems with doing it this way?

Words in terms of other words

These are "count" vectors. What are the problems with doing it this way?

- Sparsity: many words just never appear with other words.

Words in terms of other words

These are "count" vectors. What are the problems with doing it this way?

- Sparsity: many words just never appear with other words.
- Dimensionality: especially if you use fancy features (syntax, etc), you get million dimensional spaces.

Words in terms of other words

These are "count" vectors. What are the problems with doing it this way?

- Sparsity: many words just never appear with other words.
- Dimensionality: especially if you use fancy features (syntax, etc), you get million dimensional spaces.

What we need? Dimensionality reduction, or some other way to start from a compressed space.

Words in terms of other words

These are "count" vectors. What are the problems with doing it this way?

- Sparsity: many words just never appear with other words.
- Dimensionality: especially if you use fancy features (syntax, etc), you get million dimensional spaces.

What we need? Dimensionality reduction, or some other way to start from a compressed space.

- Sharing dimensions helps generalization.

Words in terms of other words

These are "count" vectors. What are the problems with doing it this way?

- Sparsity: many words just never appear with other words.
- Dimensionality: especially if you use fancy features (syntax, etc), you get million dimensional spaces.

What we need? Dimensionality reduction, or some other way to start from a compressed space.

- Sharing dimensions helps generalization.
- Nevertheless, there's value in count vectors (for things that require explicit linguistic knowledge)

Words in terms of other words

These are "count" vectors. What are the problems with doing it this way?

- Sparsity: many words just never appear with other words.
- Dimensionality: especially if you use fancy features (syntax, etc), you get million dimensional spaces.

What we need? Dimensionality reduction, or some other way to start from a compressed space.

- Sharing dimensions helps generalization.
- Nevertheless, there's value in count vectors (for things that require explicit linguistic knowledge)

So now... "predict" vectors...

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*':

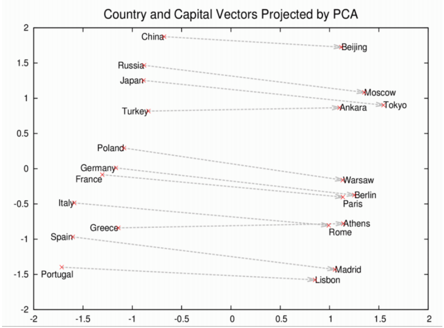
Word	Cosine Distance
Norway	0.75
Denmark	0.72
Finland	0.62
Switzerland	0.59
...	

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*



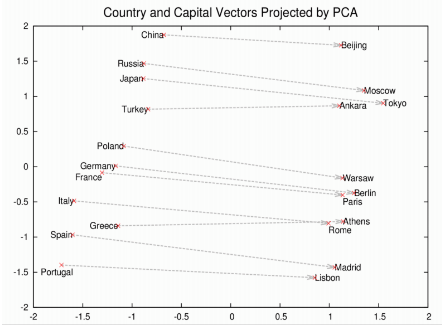
Applications of Word Vectors

- **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

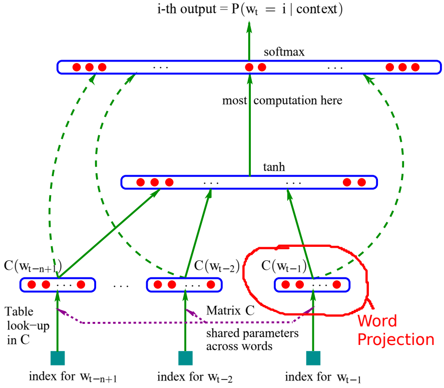
- **Sentence Completion** (actually just restricted language modeling):
- "All red-headed men who are above the age of [800 | seven | twenty-one | 1,200 | 60,000] years , are eligible."
- "That is his [generous | mother's | successful | favorite | main] fault , but on the whole he's a good worker."

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

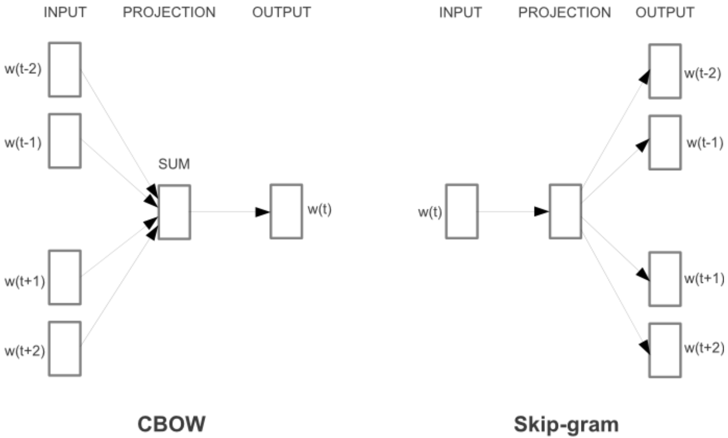
word2vec

- 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 log-linear 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 log-linear 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

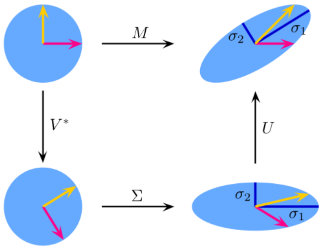


Hyperparameters

- 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



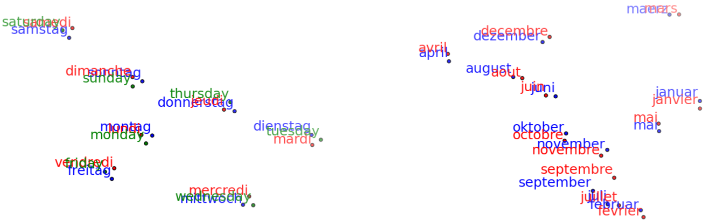
$$M = U \cdot \Sigma \cdot V^*$$

Unifying these Approaches

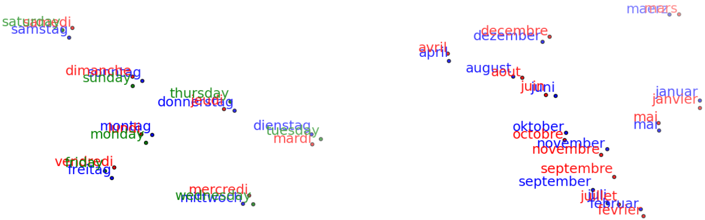
- Word2vec, Glove, and LSA all do matrix factorization (Levy & Goldberg, 2014), but the successful ones are weighted for word frequency
- Pointwise Mutual Information (PMI) is (implicitly) used by these:

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

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