Word Embeddings

Pawan Goyal

CSE, IIT Kharagpur

ACM Summer School

Word Vectors

- At one level, it is simply a vector of weights.

Word Vectors

- At one level, it is simply a vector of weights.
- In a simple 1-of-N (or 'one-hot') encoding every element in the vector is associated with a word in the vocabulary.

Word Vectors

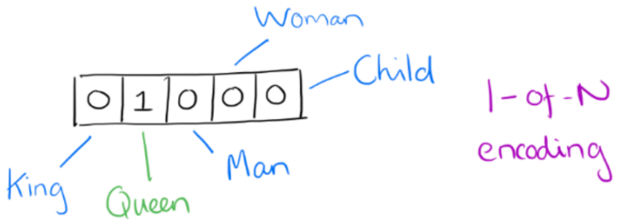
- At one level, it is simply a vector of weights.
- In a simple 1-of-N (or 'one-hot') encoding every element in the vector is associated with a word in the vocabulary.
- The encoding of a given word is simply the vector in which the corresponding element is set to one, and all other elements are zero.

One-hot representation

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Word Vectors - One-hot Encoding

- Suppose our vocabulary has only five words: King, Queen, Man, Woman, and Child.
- We could encode the word 'Queen' as:



Limitations of One-hot encoding

Limitations of One-hot encoding

Word vectors are not comparable

Using such an encoding, there is no meaningful comparison we can make between word vectors other than equality testing.

Word2Vec – A distributed representation

Distributional representation – word embedding?

Any word w_i in the corpus is given a distributional representation by an embedding

$$w_i \in R^d$$

i.e., a $d-$dimensional vector, which is mostly learnt!

Word2Vec – A distributed representation

Distributional representation – word embedding?

Any word w_i in the corpus is given a distributional representation by an embedding

$$w_i \in R^d$$

i.e., a $d-$dimensional vector, which is mostly learnt!

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Distributional Representation

- Take a vector with several hundred dimensions (say 1000).
- Each word is represented by a distribution of weights across those elements.
- So instead of a one-to-one mapping between an element in the vector and a word, the representation of a word is spread across all of the elements in the vector, and
- Each element in the vector contributes to the definition of many words.

Distributional Representation: Illustration

If we label the dimensions in a hypothetical word vector (there are no such pre-assigned labels in the algorithm of course), it might look a bit like this:

Such a vector comes to represent in some abstract way the 'meaning' of a word

Word Embeddings

- d typically in the range 50 to 1000
- Similar words should have similar embeddings

- d typically in the range 50 to 1000
- Similar words should have similar embeddings

SVD can also be thought of as an embedding method

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.
- Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship.

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.
- Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship.

Case of Singular-Plural Relations

If we denote the vector for word i as x_i, and focus on the singular/plural relation, we observe that

Reasoning with Word Vectors

- It has been found that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way.
- Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship.

Case of Singular-Plural Relations

If we denote the vector for word i as x_i, and focus on the singular/plural relation, we observe that

$$x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families} \approx x_{car} - x_{cars}$$

and so on.

Reasoning with Word Vectors

Perhaps more surprisingly, we find that this is also the case for a variety of semantic relations.

Good at answering analogy questions

a is to b, as c is to ?

man is to *woman* as *uncle* is to ? (*aunt*)

Reasoning with Word Vectors

Perhaps more surprisingly, we find that this is also the case for a variety of semantic relations.

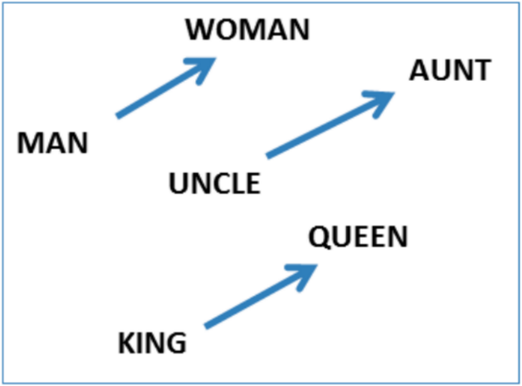
Good at answering analogy questions

a is to b, as c is to ?

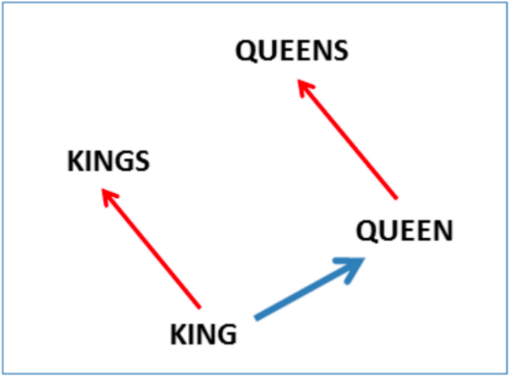
man is to *woman* as *uncle* is to ? (*aunt*)

A simple vector offset method based on cosine distance shows the relation.

Vcctor Offset for Gender Relation



Vcctor Offset for Singular-Plural Relation



Encoding Other Dimensions of Similarity

Analogy Testing

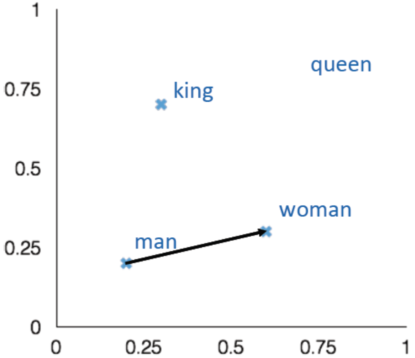
Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



Country-capital city relationships

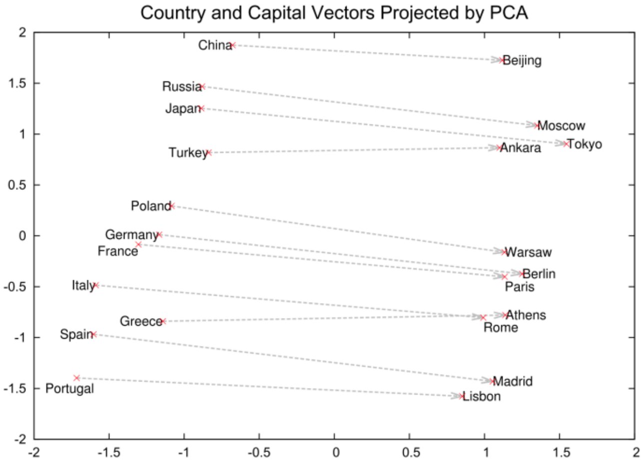


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

More Analogy Questions

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.

Element Wise Addition

We can also use element-wise addition of vector elements to ask questions such as ‘German + airlines’ and by looking at the closest tokens to the composite vector come up with impressive answers:

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

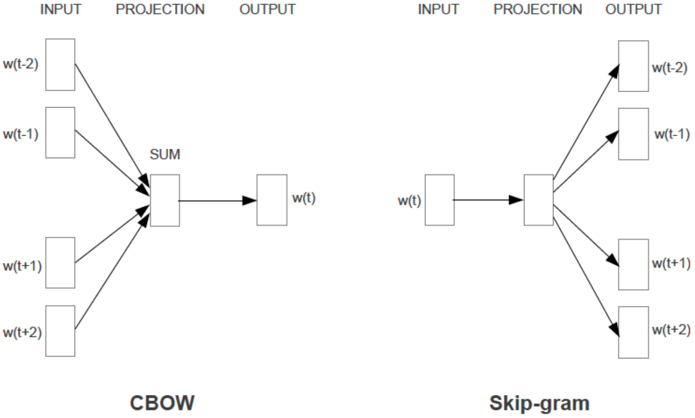
Learning Word Vectors

Basic Idea

Instead of capturing co-occurrence counts directly, predict (using) surrounding words of every word.

Code as well as word-vectors: *https://code.google.com/p/word2vec/*

Two Variations: CBOW and Skip-grams



CBOW

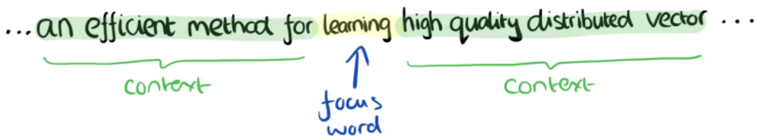
- Consider a piece of prose such as:
  "The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precises syntatic and semantic word relationships."

CBOW

- Consider a piece of prose such as:
  "The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precises syntatic and semantic word relationships."
- Imagine a sliding window over the text, that includes the central word currently in focus, together with the four words that precede it, and the four words that follow it:

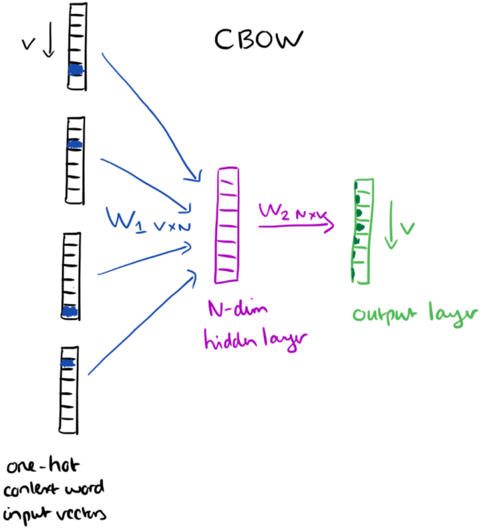
CBOW

- Consider a piece of prose such as:
  "The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precises syntatic and semantic word relationships."
- Imagine a sliding window over the text, that includes the central word currently in focus, together with the four words that precede it, and the four words that follow it:



CBOW

The context words form the input layer. Each word is encoded in one-hot form. A single hidden and output layer.



CBOW: Training Objective

- The training objective is to maximize the conditional probability of observing the actual output word (the focus word) given the input context words, with regard to the weights.
- In our example, given the input (“an”, “efficient”, “method”, “for”, “high”, “quality”, “distributed”, “vector”), we want to maximize the probability of getting “learning” as the output.

Since our input vectors are one-hot, multiplying an input vector by the weight matrix W_1 amounts to simply selecting a row from W_1.

$$\begin{array}{cccc} \text{input} & W_1 & & \text{hidden} \\ 1\times V & V\times N & & 1\times N \\ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} & \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} & = & \begin{bmatrix} e & f & g & h \end{bmatrix} \\ & W_1 & & \end{array}$$

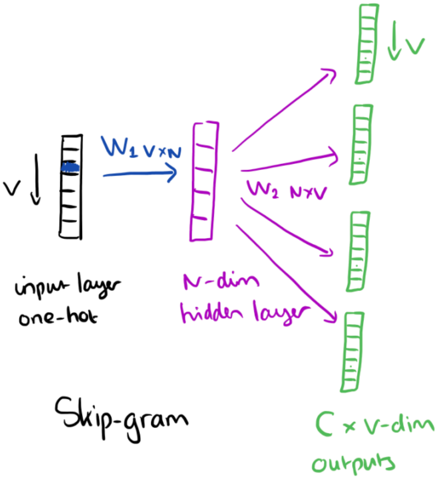
Given C input word vectors, the activation function for the hidden layer h amounts to simply summing the corresponding 'hot' rows in W_1, and dividing by C to take their average.

CBOW: Hidden to Output Layer

From the hidden layer to the output layer, the second weight matrix W_2 can be used to compute a score for each word in the vocabulary, and softmax can be used to obtain the posterior distribution of words.

Skip-gram Model

The skip-gram model is the opposite of the CBOW model. It is constructed with the focus word as the single input vector, and the target context words are now at the output layer:



Skip-gram Model: Training

- The activation function for the hidden layer simply amounts to copying the corresponding row from the weights matrix W_1 (linear) as we saw before.
- At the output layer, we now output C multinomial distributions instead of just one.
- The training objective is to mimimize the summed prediction error across all context words in the output layer. In our example, the input would be "learning", and we hope to see ("an", "efficient", "method", "for", "high", "quality", "distributed", "vector") at the output layer.

Skip-gram Model

Details

Predict surrounding words in a window of length c of each word

Skip-gram Model

Details

Predict surrounding words in a window of length c of each word

Objective Function: Maximize the log probablility of any context word given the current center word:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} log\ p(w_{t+j}|w_t)$$

Word Vectors

For $p(w_{t+j}|w_t)$ the simplest first formulation is

$$p(w_O|w_I) = \frac{exp({v'_{wO}}^T v_{WI})}{\sum_{w=1}^{W} exp({v'_w}^T v_{WI})}$$

Word Vectors

For $p(w_{t+j}|w_t)$ the simplest first formulation is

$$p(w_O|w_I) = \frac{exp({v'_{wO}}^T v_{WI})}{\sum_{w=1}^{W} exp({v'_w}^T v_{WI})}$$

where v and v' are "input" and "output" vector representations of w (so every word has two vectors)

Parameters θ

With $d-$dimensional words and V many words:

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ v'_{aardvark} \\ v'_a \\ \vdots \\ v'_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

Gradient Descent for Parameter Updates

$$\theta_j{}^{new} = \theta_j{}^{old} - \alpha \frac{\partial}{\partial \theta_j{}^{old}} J(\theta)$$

Implementation Tricks

Batch update would take a very long time

Instead, parameters are updated after each window t.

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J_t(\theta)$$

Implementation Tricks

Batch update would take a very long time

Instead, parameters are updated after each window t.

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J_t(\theta)$$

Computing denominator in $p(w_O|w_I)$ is too computionally expensive

Negative Sampling

$$\log \sigma \left(v_{wI}^T v'_{wO} \right) + \sum_{i \sim P_n(w)} \log \sigma \left(-v_{wI}^T v'_{wi} \right)$$

Two sets of vectors

Best solution is to sum these up

$$L_{final} = L + L'$$

Two sets of vectors

Best solution is to sum these up

$$L_{final} = L + L'$$

A good tutorial to understand parameter learning:

https://arxiv.org/pdf/1411.2738.pdf

Two sets of vectors

Best solution is to sum these up

$$L_{final} = L + L'$$

A good tutorial to understand parameter learning:

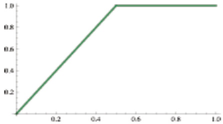
https://arxiv.org/pdf/1411.2738.pdf

An interactive Demo

https://ronxin.github.io/wevi/

Glove

$$J = \frac{1}{2}\sum_{ij} f(P_{ij})\left(w_i \cdot \tilde{w}_j - \log P_{ij}\right)^2 \qquad f \sim$$

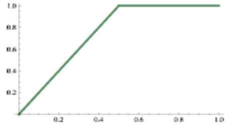


Combine the best of both worlds – count based methods as well as direct prediction methods

Glove

$$J = \frac{1}{2} \sum_{ij} f(P_{ij}) \left(w_i \cdot \tilde{w}_j - \log P_{ij} \right)^2$$

$f \sim$

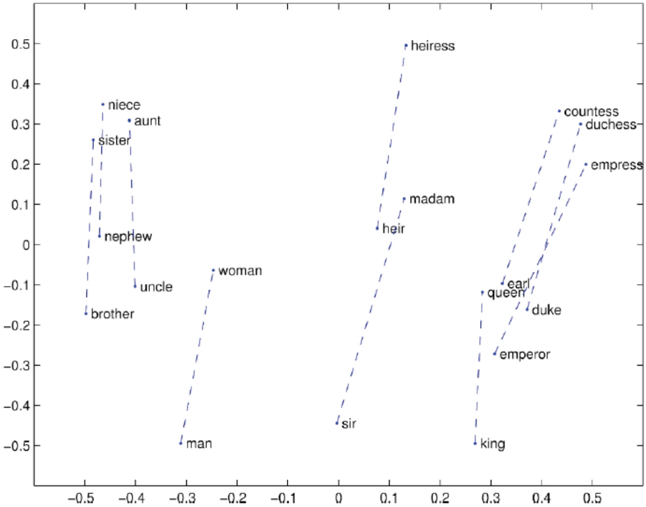


Combine the best of both worlds – count based methods as well as direct prediction methods

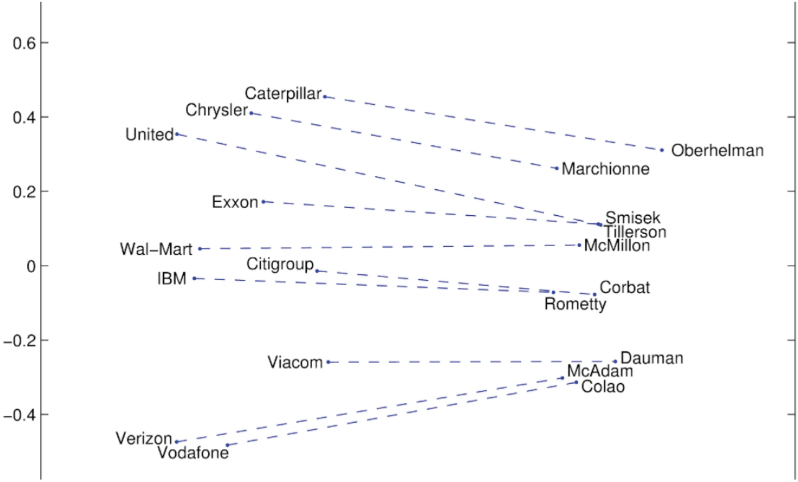
- Fast training
- Scalable to huge corpora
- Good performance even with small corpus, and small vectors

Code and vectors: http://nlp.stanford.edu/projects/glove/

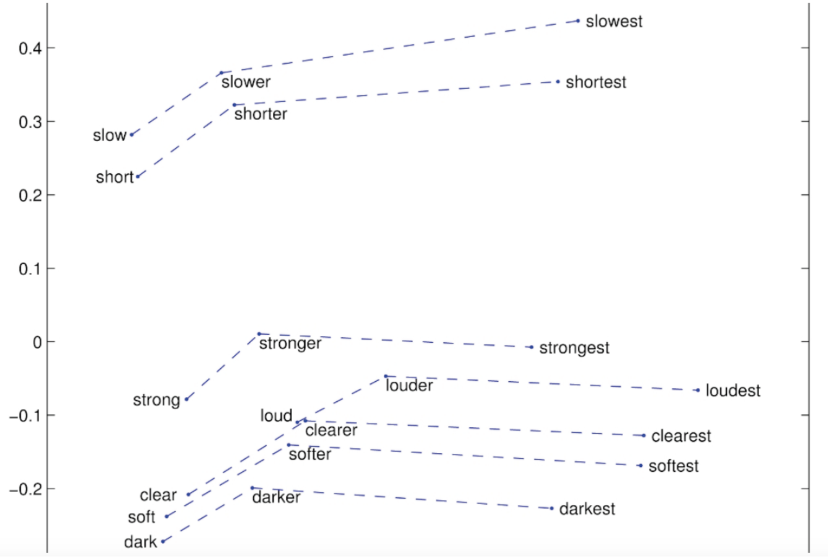
Glove Visualisations



Glove Visualisations



Glove Visualisations



Intrinsic Evaluation

- Word vector distances and their correlation with human judgements
- Example dataset: WordSim353

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer		internet 7.58
plane	car	5.77
professor		doctor 6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Hyperparameters

Skip-gram: using evaluation on analogy testing

- **Dimensions:** 300 dimensions work the best
- **Window size:** 8 words around each center word works well.
- **More training time and data helps!!**

Handling Polysemy

Problem with word vectors

Multiple senses of a given word get the same representation!!

Handling Polysemy

Problem with word vectors

Multiple senses of a given word get the same representation!!

Huang et al., "Improving Word Representations via Global Context and Multiple Word Prototypes", ACL 2012.

Basic Idea

Cluster words windows around words, retrain with each word assigned to multiple different clusters, e.g., bank_1, bank_2 etc.

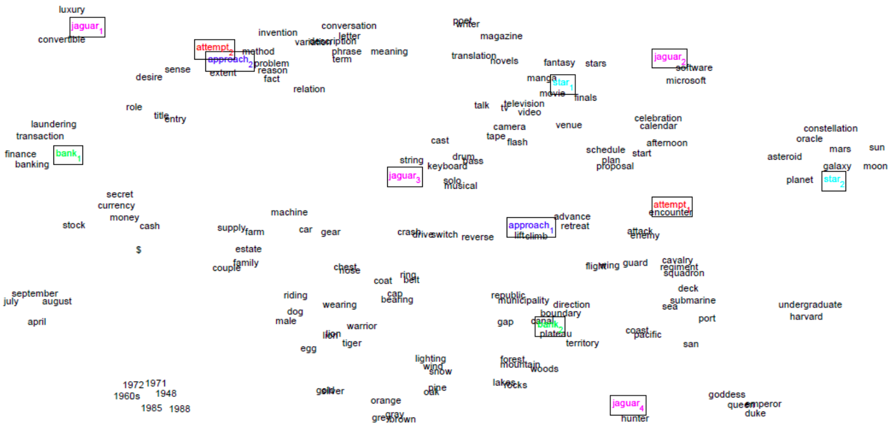
Multiple Word Prototypes: Nearest Neighbors

Center Word	Nearest Neighbors
bank_1	corporation, insurance, company
bank_2	shore, coast, direction
star_1	movie, film, radio
star_2	galaxy, planet, moon
cell_1	telephone, smart, phone
cell_2	pathology, molecular, physiology
left_1	close, leave, live
left_2	top, round, right

Code and dataset:

http://www.socher.org/index.php/Main/ImprovingWordRepresentationsViaGlobalC

Multiple Word Prototypes: Visualization



Cross-lingual applications: Word Embeddings

Cross-lingual information retrieval task

Query is in one language and documents in another language

Cross-lingual applications: Word Embeddings

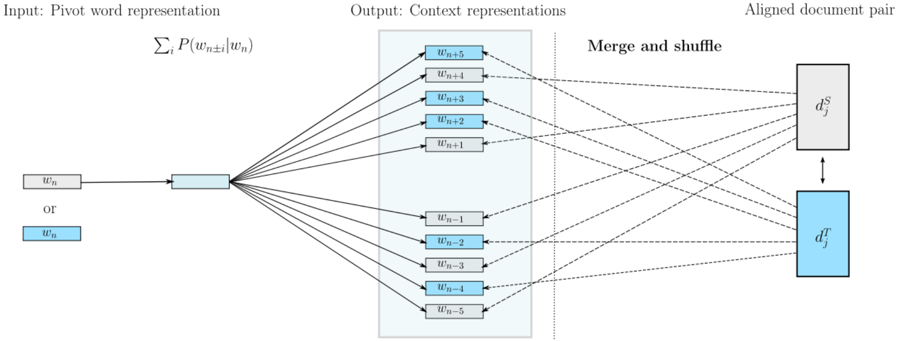
Cross-lingual information retrieval task

Query is in one language and documents in another language

Obtaining common representations for words in multiple languages

So that you can visualize words in multiple languages in the same space

Bilingual Word Embeddings: Merge and Shuffle

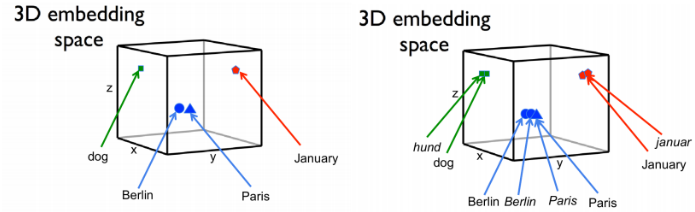


Vulić, Ivan, and Marie-Francine Moens. "Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings." SIGIR 2015.

Other Approaches

- The previous approach requires a comparable corpora. What if you do not have such corpora?
- Suppose you have a dictionary to start with, what would be an approach?

Applications: Bilingual Word Embeddings



1. Align pretrained monolingual embedding spaces **(offline)** *using dictionaries* [Mikolov et al., arXiv 2013; Lazaridou et al., ACL 2015]
2. Jointly learn and align embeddings **(online)** using *parallel-only data* [Hermann and Blunsom, ACL 2014; Chandar et al., NIPS 2014]
3. Jointly learn and align embeddings **(online)** using *mono* **and** *parallel data* [Gouws et al., ICML 2015; Soyer et al., ICLR 2015, Shi et al., ACL 2015]

Applications: Sentiment Specific Embeddings

Basic Idea

Introduce an additional objective: The word vectors of the model should predict the sentiment label using some appropriate predictor.

$$\hat{s} = f(\phi_w)$$

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

Andrew Mass' Model

	Our model Sentiment + Semantic	Our model Semantic only	LSA
melancholy	bittersweet heartbreaking happiness tenderness compassionate	thoughtful warmth layer gentle loneliness	poetic lyrical poetry profound vivid
ghastly	embarrassingly trite laughably atrocious appalling	predators hideous tube baffled smack	hideous inept severely grotesque unsuspecting
lackluster	lame laughable unimaginative uninspired awful	passable unconvincing amateurish clichéd insipid	uninspired flat bland forgettable mediocre
romantic	romance love sweet beautiful relationship	romance charming delightful sweet chemistry	romance screwball grant comedies comedy

Table 1: Similarity of learned word vectors. Each target word is given with its five most similar words using cosine similarity of the vectors determined by each model. The full version of our model (left) captures both lexical similarity as well as similarity of sentiment strength and orientation. Our unsupervised semantic component (center) and LSA (right) capture semantic relations.