Learning Representations of Text using Neural Networks

Speaker:

Claudio Baecchi

Authors:

Tomáš Mikolov

Joint work with Ilya Sutskever, Kai Chen, Greg Corrado, Jeff Dean, Quoc Le, Thomas Strohmann

Work presented at NIPS 2013 Deep Learning Workshop

Overview

- Distributed Representations of Text
- Efficient learning
- Linguistic regularities
- Examples
- Translation of words and phrases
- Available resources

Representations of Text

Representation of text is very important for performance of many real-world applications. The most common techniques are:

- Local representations
  - N-grams
  - Bag-of-words
  - 1-of-N coding
- Continuous representations
  - Latent Semantic Analysis
  - Latent Dirichlet Allocation
  - **Distributed Representations**

Distributed Representations

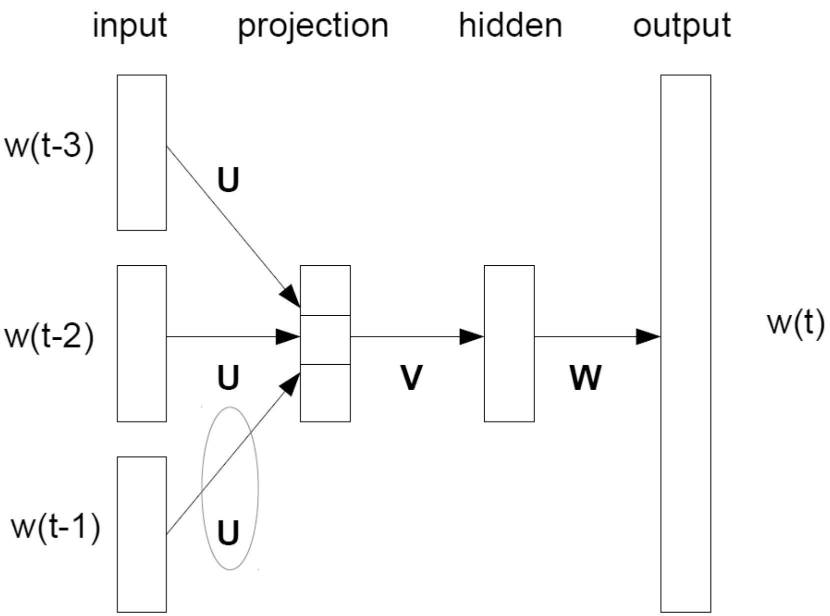
The idea behind distributed representations is to characterize an object using many features

- This reflects the way we think the brain works
- We think they can help generalizing to new objects that are similar to known ones

Distributed representations of words can be obtained from various neural network based language models:

- Feedforward neural net language model
- Recurrent neural net language model

Feedforward Neural Net Language Model

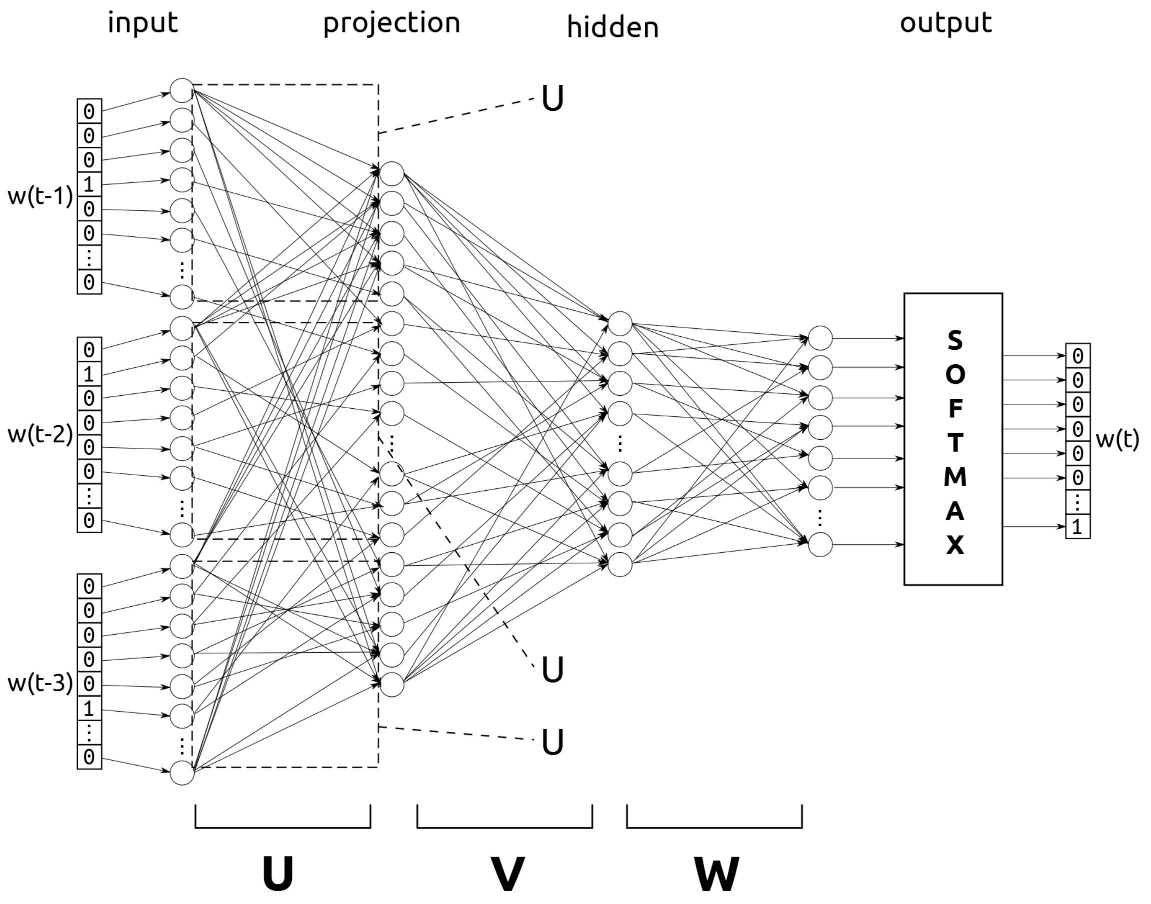


> **U**, **V** and **W** are weight matrices whose values are to be learned by the network.

> When training is complete, **U** will be used to “translate“ any word into the respective vector in the continuous space

- Four-gram neural net language model architecture (Bengio 2001)
- The training is done using stochastic gradient descent and backpropagation
- The word vectors are in matrix U

Feedforward Neural Net Language Model

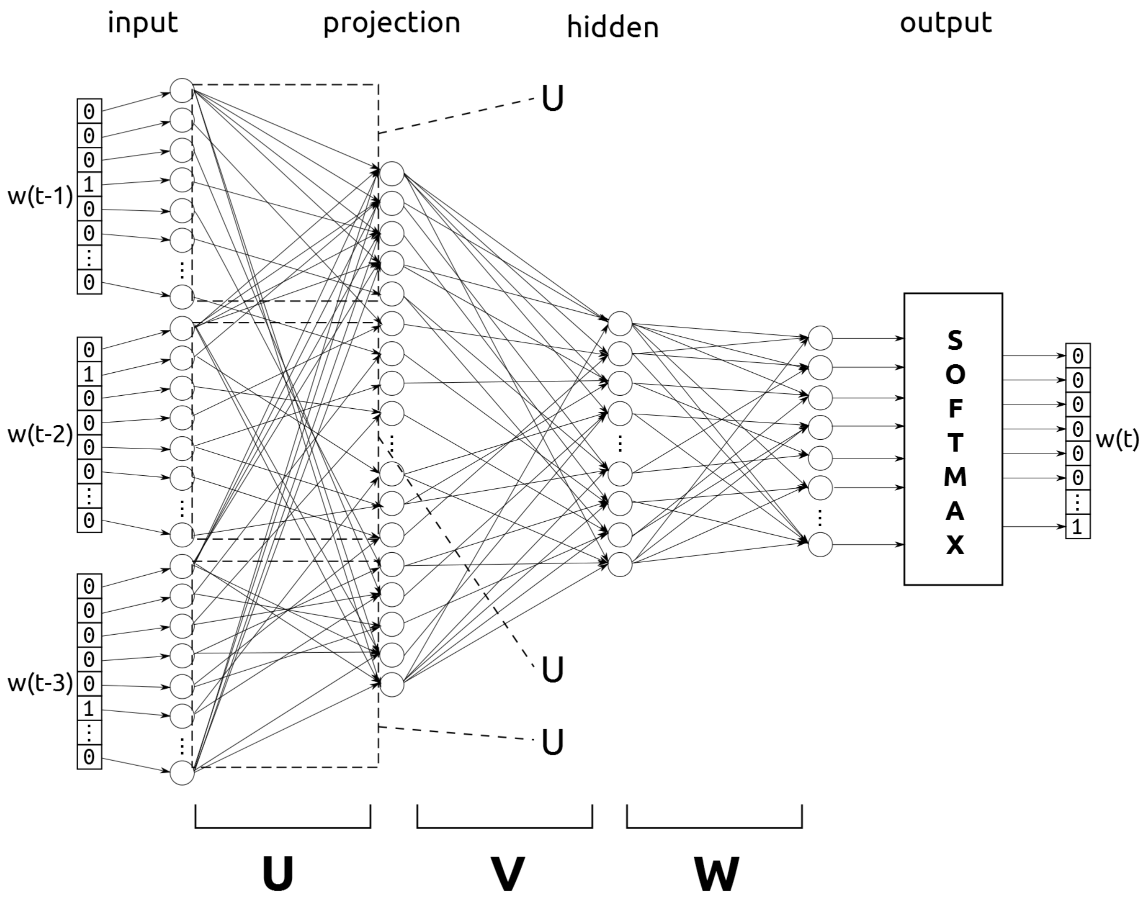


w(t-3)	*w(t-2)*	*w(t-1)*	*w(t)*
the	cat	is	black
mary	has	a	lamb
i	know	this	place

The network is trained using w(t-3), w(t-2) and w(t-1) as the input (the “context“ of w(t)) and w(t) as expected result.

Each word is used as input in its one-hot form in the vocabulary. The output of the network is continuous so a softmax function must be used to assign the output to a word.

Feedforward Neural Net Language Model



Where:

U has size $|V| \times P$

V has size $(N*P) \times H$

W has size $H \times |V|$

And:

$|V|$ is the vocabulary size

P is the size of the projection space

H is the size of the hidden layer

N is the context size

Feedforward Neural Net Language Model

After the training we obtain a mapping between each word in the vocabulary (discrete form) and its continuous form.

- The mapping is performed by computing the product: $\mathbf{w}^\mathsf{T}\mathbf{U}$
- The one-hot nature of the word encoding in the vocabulary reduces the previous product to the selection of the i-th row of $\mathbf{U}$, where i is the position of the 1 in the word vector
- The mapping operation can thus be expressed as $\mathbf{U}(i)$

Efficient Learning

The training complexity of the feedforward NNLM is high:

- Propagation from projection layer to the hidden layer
- Softmax in the output layer

Using this model just for obtaining the word vectors is very inefficient

Efficient Learning

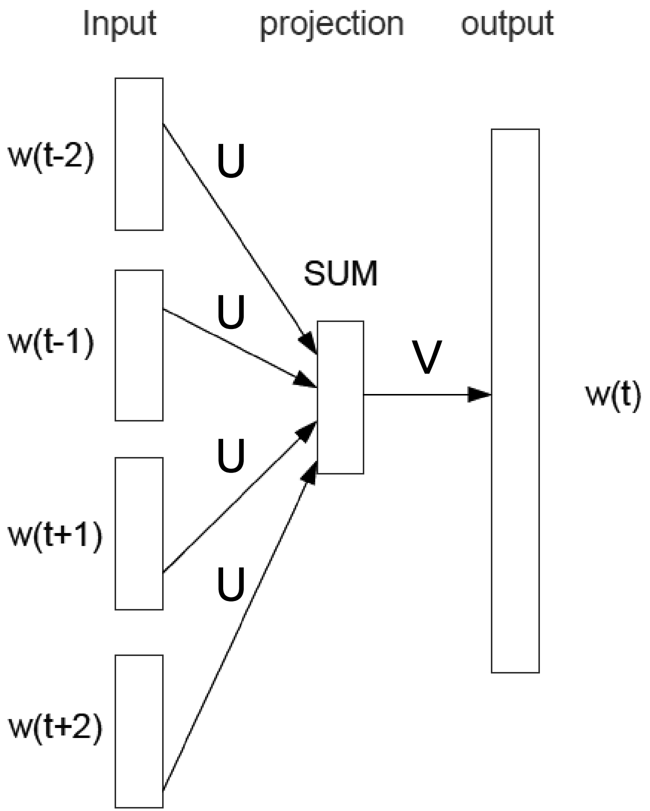
The full softmax can be replaced by:

- Hierarchical softmax (Morin and Bengio)
- Hinge loss (Collobert and Weston)
- Noise contrastive estimation (Mnih et al.)
- Negative sampling (Mikolov et al.)

We can further remove the hidden layer: for large models, this can provide additional speedup (up to 1000x)

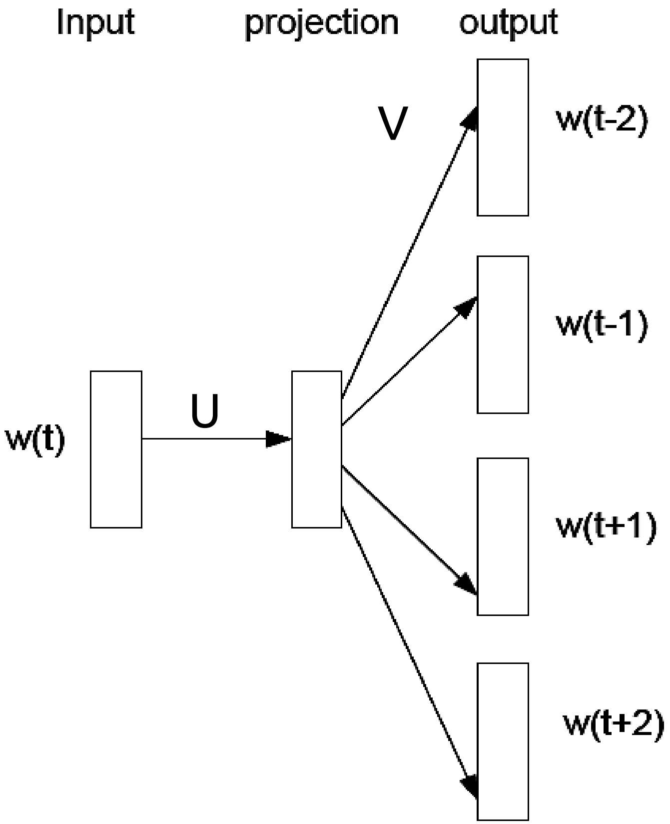
- Continuous bag-of-words model
- Continuous skip-gram model

Continuous Bag-of-words Architecture



- Predicts the current word given the context

Skip-gram Architecture



- Predicts the surrounding words given the current word

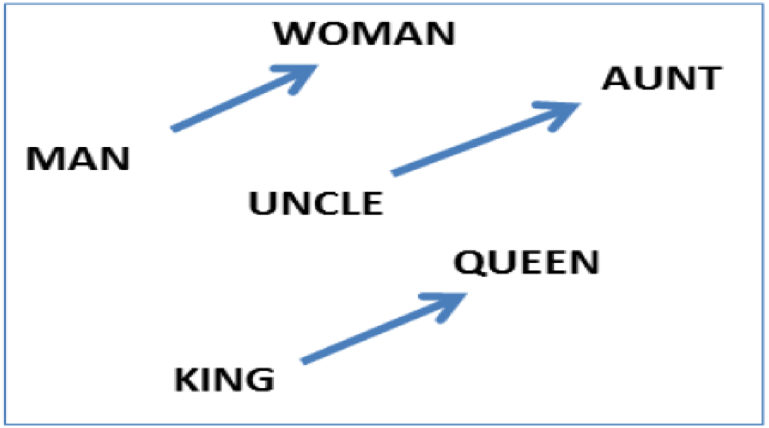
Efficient Learning - Summary

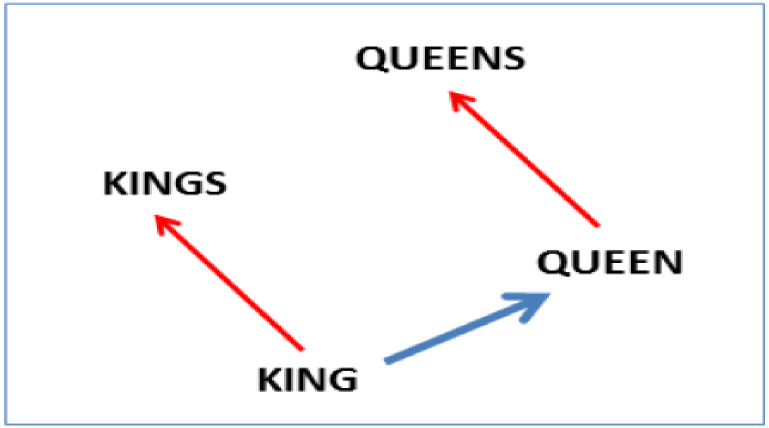
Efficient multi-threaded implementation of the new models greatly reduces the training complexity

The training speed is in order of 100k - 5M words per second

Quality of word representations improves significantly with more training data

Linguistic Regularities in Word Vector Space





- The word vector space implicitly encodes many regularities among words

Linguistic Regularities in Word Vector Space

The resulting distributed representations of words contain surprisingly a lot of syntactic and semantic information

There are multiple degrees of similarity among words:

- **KING** is similar to **QUEEN** as **MAN** is similar to **WOMAN**
- **KING** is similar to **KINGS** as **MAN** is similar to **MEN**

Simple vector operations with the word vectors provide very intuitive results

Linguistic Regularities - Results

- Regularity of the learned word vector space is evaluated using test set with about 20k questions
- The test set contains both syntactic and semantic questions
- We measure TOP1 accuracy (input words are removed during search)
- We compare our models to previously published word vectors

Linguistic Regularities - Results

Model	*Vector Dimensionality*	*Training Words*	*Training Time*	*Accuracy [%]*
Collobert NNLM	50	660M	2 months	11
Turian NNLM	200	37M	few weeks	2
Mnih NNLM	100	37M	7 days	9
Mikolov RNNLM	640	320M	weeks	25
Huang NNLM	50	990M	weeks	13
Our NNLM	100	6B	2.5 days	51
Skip-gram (hier.s.)	1000	6B	Hours	66
CBOW (negative)	300	1.5B	**minutes**	**72**

Linguistic Regularities - Results

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

Performance on Rare Words

- Word vectors from neural networks were previously criticized for their poor performance on rare words
- Scaling up training data set size helps to improve performance on rare words
- For evaluation of progress, we have used data set from Luong et al.: *Better word representations with recursive neural networks for morphology*, CoNLL 2013

Performance on Rare Words - Results

Model	*Correlation with Human Ratings (Spearman's rank correlation)*
Collobert NNLM	0.28
Collobert NNLM + Morphology features	0.34
CBOW (100B)	**0.50**

Rare Words - Examples of Nearest Neighbours

	Redmond	Havel	graffiti	capitulate
Collobert NNLM	conyers lubbock keene	plauen dzerzhinsky osterreich	cheesecake gossip dioramas	abdicate accede rearm
Turian NNLM	McCarthy Alston Cousins	Jewell Arzu Ovitz	gunfire emotion impunity	- - -
Mnih NNLM	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-gram (phrases)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	spray paint grafitti taggers	capitulation capitulated capitulating

From Words to Phrases and Beyond

Often we want to represent more than just individual words: phrases, queries, sentences

The vector representation of a query can be obtained by:

- Forming the phrases
- Adding the vectors together

From Words to Phrases and Beyond

- Example query:
  restaurants in mountain view that are not very good
- Forming the phrases:
  restaurants in (mountain view) that are (not very good)
- Adding the vectors:
  restaurants + in + (mountain view) + that + are + (not very good)
- Very simple and efficient
- Will not work well for long sentences or documents

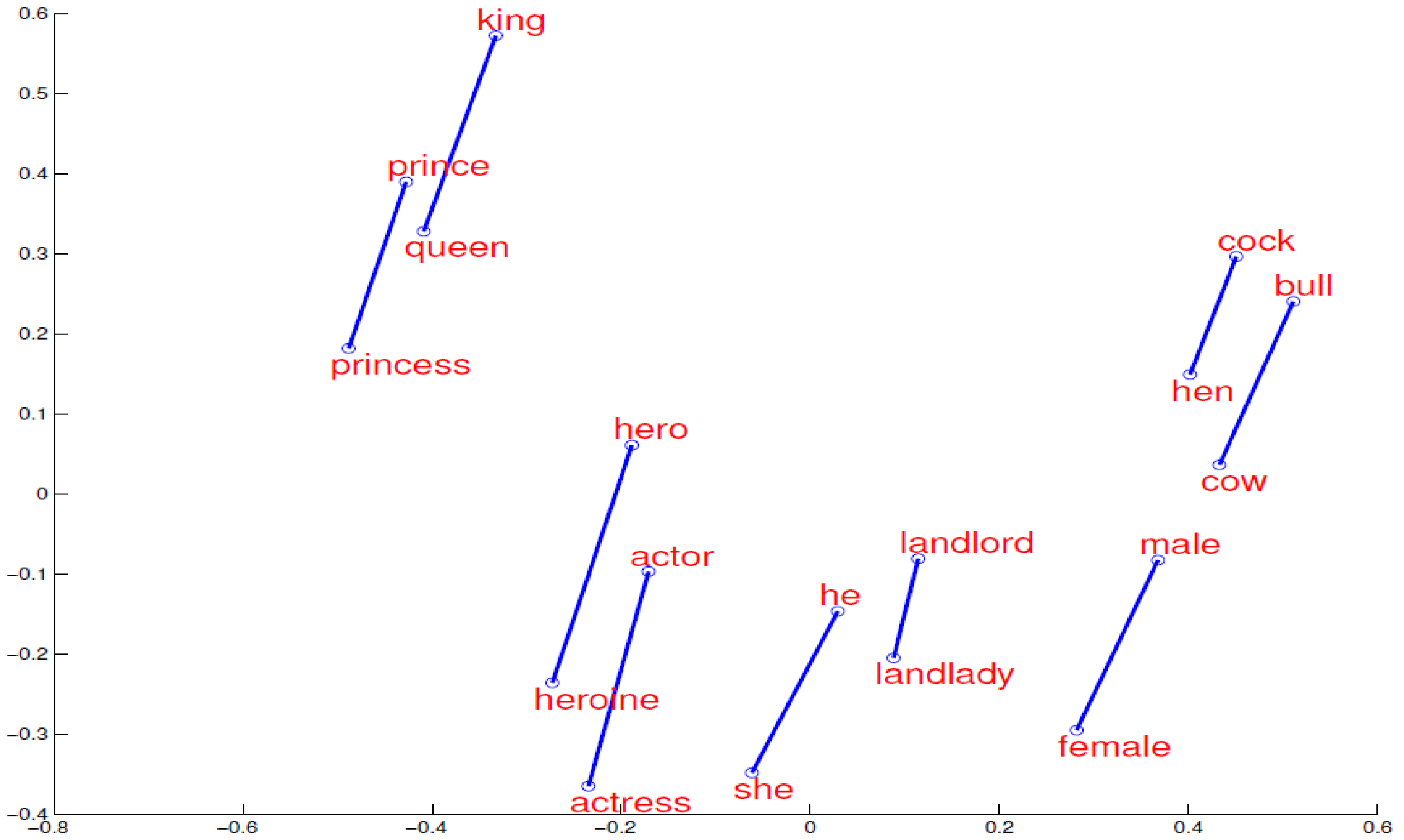
Compositionality by Vector Addition

Expression	*Nearest tokens*
Czech + currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa, flag carrier Lufthansa
Russian + river	Moscow, Volga River, upriver, Russia
French + actress	Juliette Binoche, Vanessa Paradis, Charlotte Gainsbourg

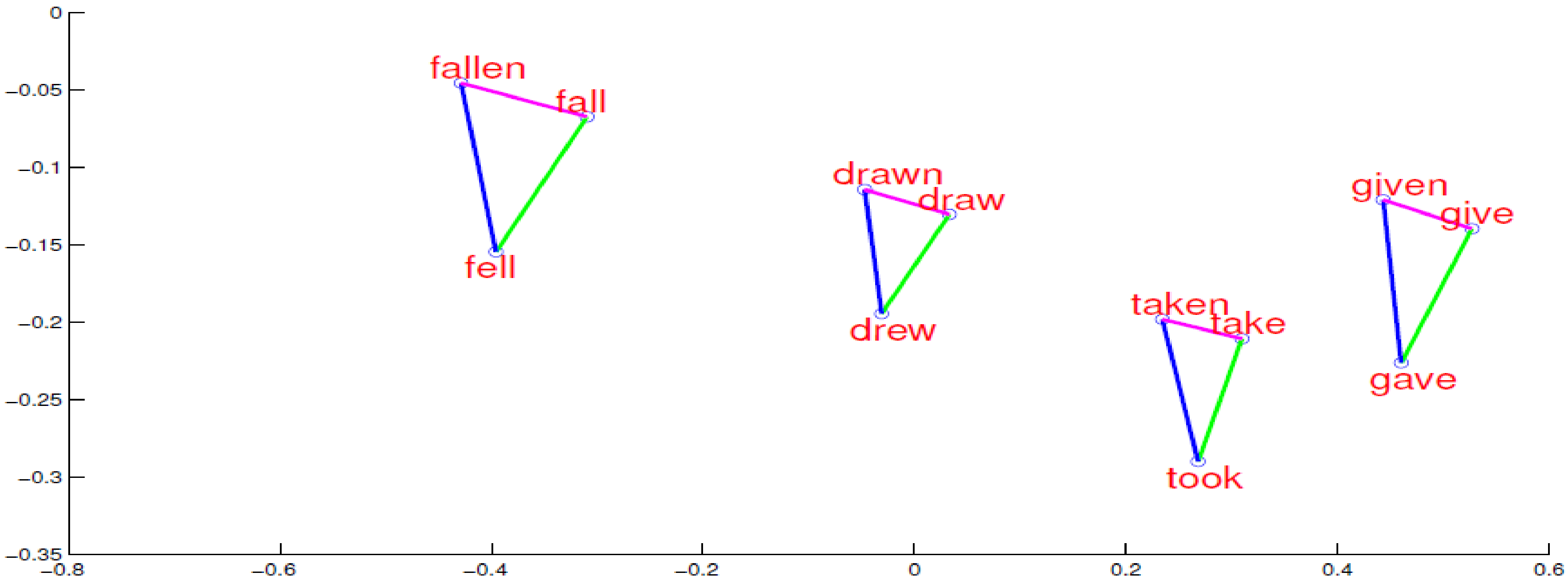
Visualization of Regularities in Word Vector Space

- We can visualize the word vectors by projecting them to 2D space
- PCA can be used for dimensionality reduction
- Although a lot of information is lost, the regular structure is often visible

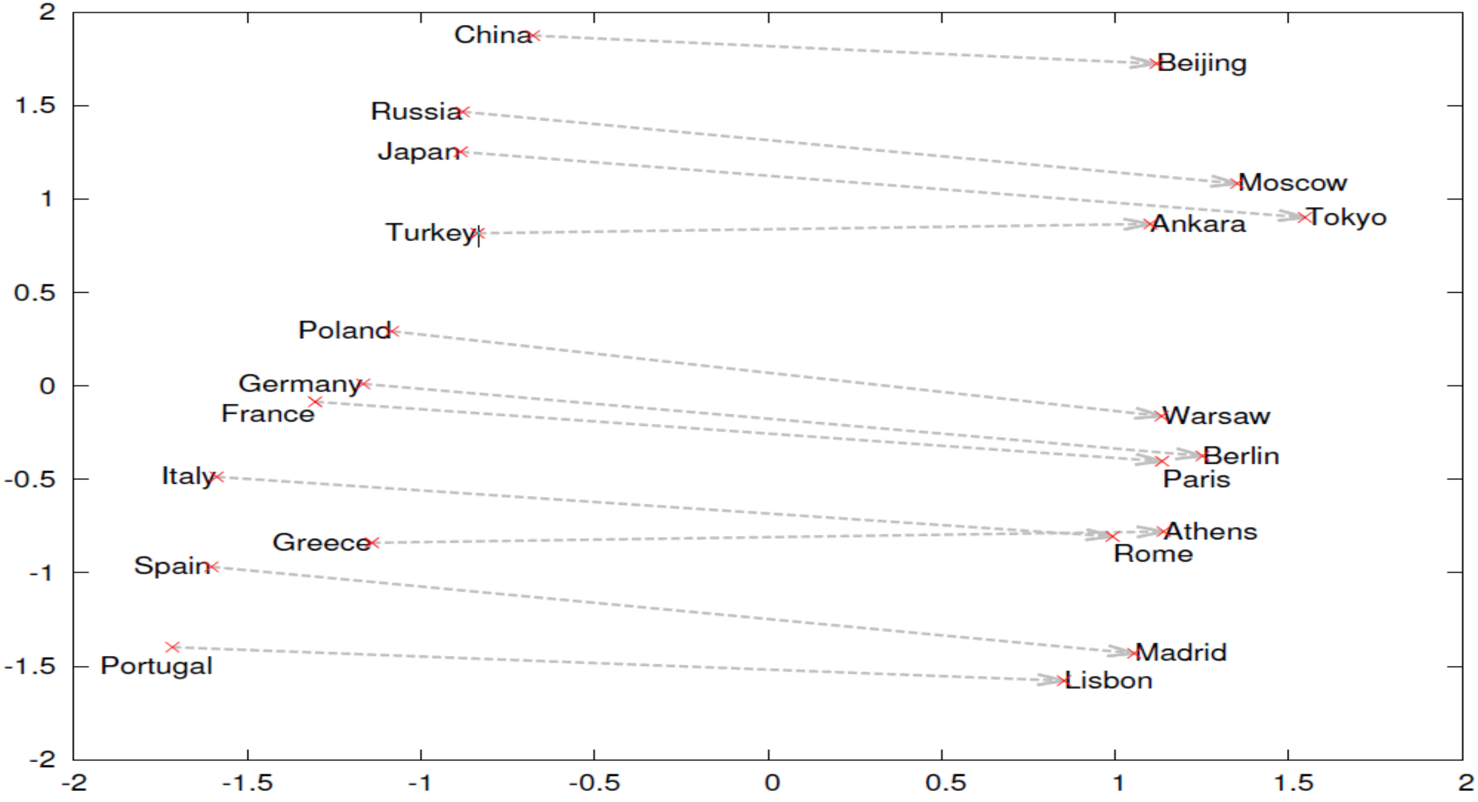
Visualization of Regularities in Word Vector Space



Visualization of Regularities in Word Vector Space



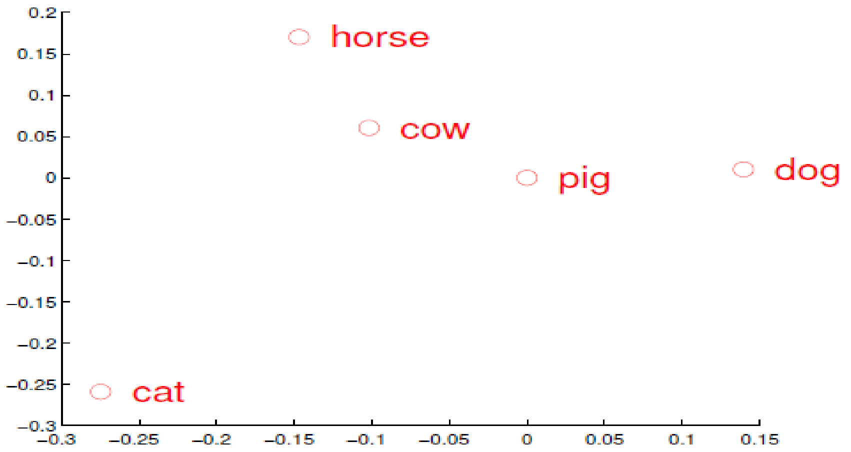
Visualization of Regularities in Word Vector Space

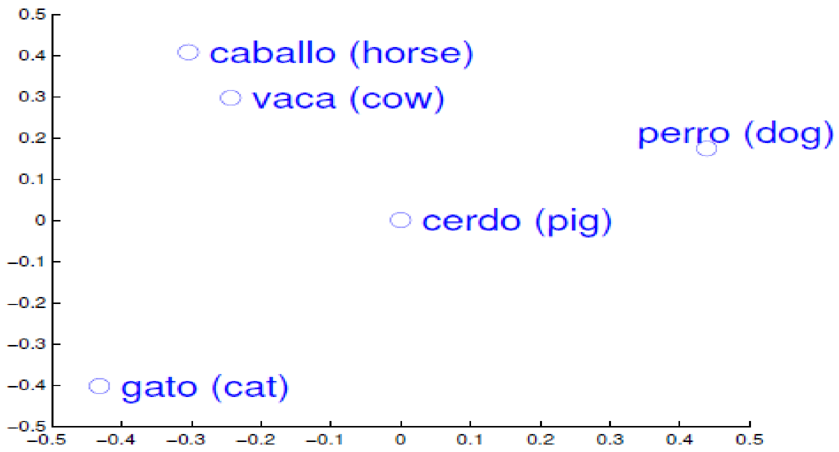


Machine Translation

- Word vectors should have similar structure when trained on comparable corpora
- This should hold even for corpora in different languages

Machine Tanslation - English to Spanish



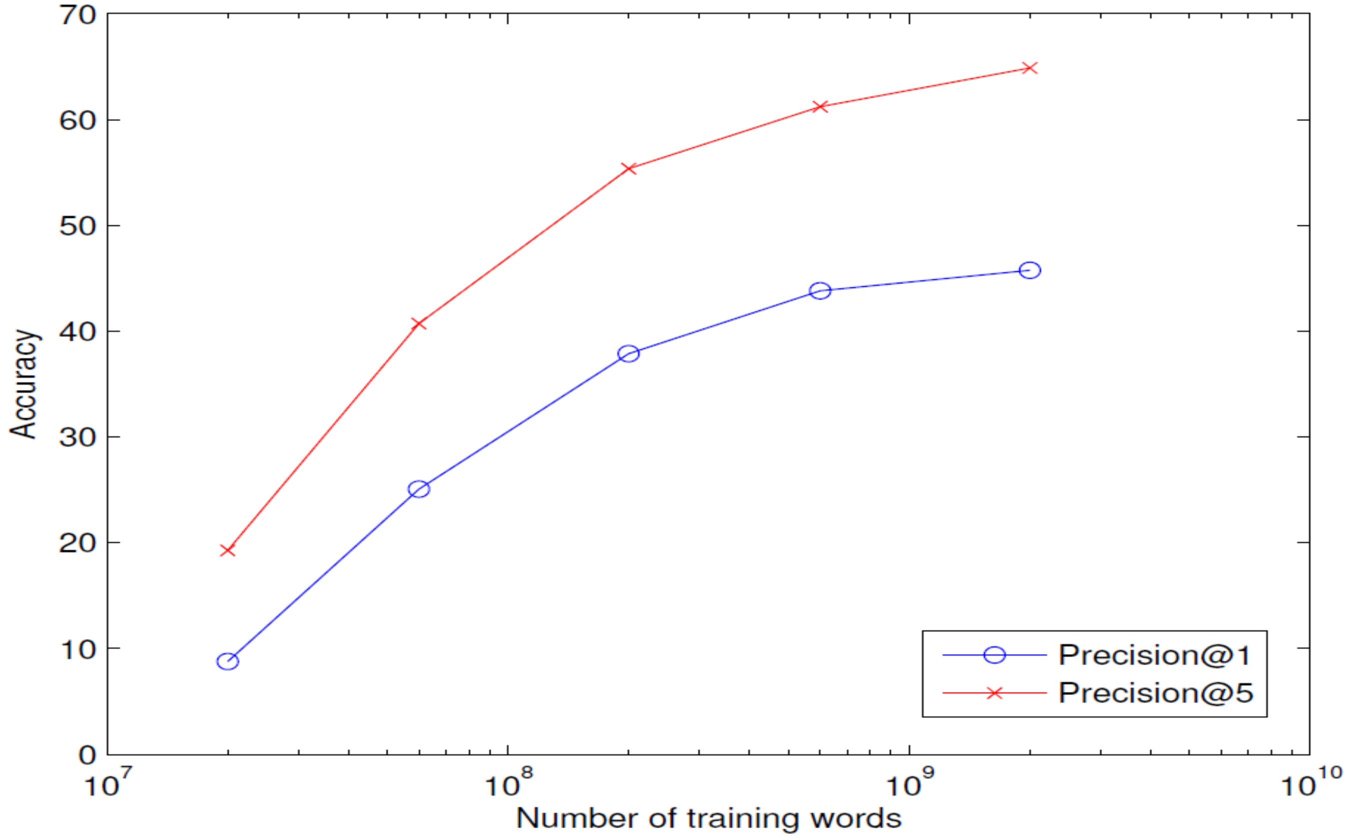


- The figures were manually rotated and scaled

Machine Translation

- For translation from one vector space to another, we need to learn a linear projection (will perform rotation and scaling)
- Small starting dictionary can be used to train the linear projection
- Then, we can translate any word that was seen in the monolingual data

MT - Accuracy of English to Spanish translation



Machine Translation

- When applied to English to Spanish word translation, the accuracy is above 90% for the most confident translations
- Can work for any language pair (we tried English to Vietnamese)
- More details in paper: *Exploiting similarities among languages for machine translation*

Available Resources

The project webpage is `code.google.com/p/word2vec`

- open-source code
- pretrained word vectors (model for common words and phrases will be uploaded soon)
- links to the papers