word2vec

Anđelka Zečević
andjelkaz@matf.bg.ac.rs

word2vec

Tool for embedded word representation generation according to the results of:

1) Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

2) Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.

https://code.google.com/archive/p/word2vec/

overview

- Motivation:
  - How do we handle semantics?
  - How can we represent words in order to keep the meaning?
- Neural Network Language Model
  - Continuous Bag of Words
  - Skip-gram

distributed semantics

A bottle of tesgüino is on the table.
Everybody likes tesgüino.
Tesgüino makes you drunk.
We make tesgüino out of corn.

The meaning of a word is related to the distribution of the words around it.

from **Speech and Language Processing** by Dan Jurafsky and James H. Martin.
*Tesgüino is a corn beer made by the Tarahumara Indians of Sierra Madre in Mexico.

distributed semantics

The hypothesis of linguistics by Firth (1957):

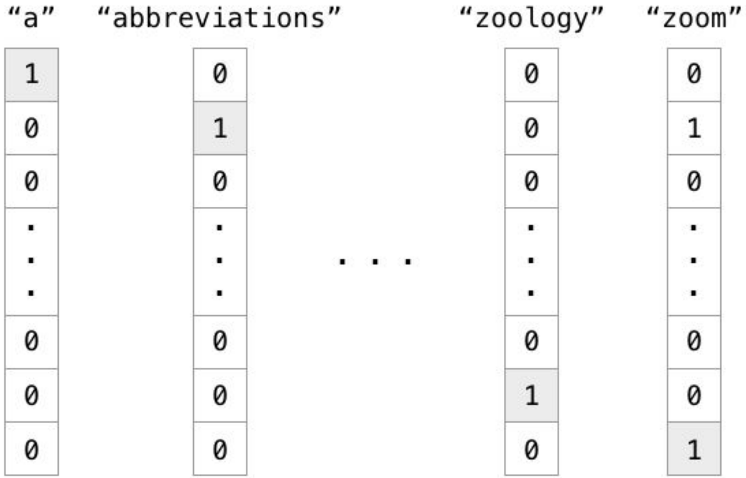
> “We shall know the word by the company it keeps.”

There can be many types of relatedness:

- synonyms: big and large
- concept categories: dog, cat → animals
- associations: bee & honey
- analogies: big and bigger as small and smaller
- ...

distributed semantics = vector semantics

Words as one-hot vectors:



No semantics!

The size of one-hot vector is equal to the vocabulary size.

distributed semantics = vector semantics

Words as rows of term-document matrix:

	D1	D2	D3	D4	D5	...
a	145	223	346	78	89	⋯
abandon	4	0	0	5	3	⋯
ability	5	10	0	4	7	⋯
able	31	35	64	3	5	⋯
about	64	68	89	24	9	⋯
above	4	5	8	0	0	⋯
abroad	0	0	1	0	0	⋯
absence	2	4	0	0	0	⋯
absent	0	0	1	0	0	⋯
absolute	3	1	5	0	1	⋯
abstract	5	1	2	1	0	⋯
abuse	0	1	0	0	0	⋯
academic	1	3	0	0	0	⋯
...	⋯	⋯	⋯	⋯	⋯	⋯

The meaning of the word is represented by documents it tends to occur in.

distributed semantics = vector semantics

Words as rows of term-term or word-word or word-context matrix: instead of documents we can define smaller contextes

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of

left context: tablespoonful of

right context: preserve or

context/window size : 2

Shorter windows can capture more syntactic connections between words while larger windows can capture more semantic information.

distributed semantics = vector semantics

Modification:

- TF-IDF: $w_{ij} = \mathrm{tf}_{ij}\mathrm{idf}_i$

  tf_{ij}: frequency of the i-th term in the j-th document

  idf_i inverse document frequency of the i-th term $\mathrm{idf}_i = \log\left(\frac{N}{df_i}\right)$

- Positive Pointwise Mutual Information:

$$\mathrm{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

distributed semantics = vector semantics

How do we quantify similarity of words?

Most commonly used metric is cosine:

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

Alternatives:

$$\begin{aligned}
\text{Jaccard}(\vec{v},\vec{w}) &= \frac{\sum_{i=1}^{N}\min(v_i,w_i)}{\sum_{i=1}^{N}\max(v_i,w_i)}\\
\text{Dice}(\vec{v},\vec{w}) &= \frac{2\times\sum_{i=1}^{N}\min(v_i,w_i)}{\sum_{i=1}^{N}(v_i+w_i)}\\
\text{JS}(\vec{v}||\vec{w}) &= D(\vec{v}|\tfrac{\vec{v}+\vec{w}}{2})+D(\vec{w}|\tfrac{\vec{v}+\vec{w}}{2})
\end{aligned}$$

distributed semantics = vector semantics

Still holds:

- the size of word vectors is equal to the vocabulary size
- word vectors are sparse

word embedding

Embedded representations: short dense vectors that keep word semantics

Embedding: the whole process

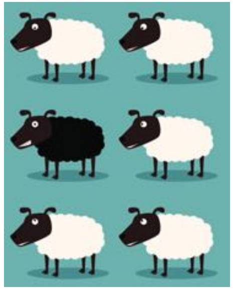
Two approaches:

- count-based methods
- predictive methods

Don't count, predict! by Marco Baroni, Georgiana Dinu and German Kruszewski, ACL 2014.

language modeling

Task: Predict a word after the sequence of *n* words.
Classical approach: maxim likelihood principle
maximize probability $P(w_n|w_{n-1}w_{n-2}\ldots w_1)$
2-gram approach:



I feel like a black ______

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

$$P(w_n w_{n-1} w_{n-2}\ldots w_1) = P(w_1)P(w_2|w_1)P(w_3|w_2w_1)\ldots P(w_n|w_{n-1}w_{n-2}\ldots w_1)$$

$$\sim P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_n|w_{n-1})$$

Issues: out-of-vocabulary words, smoothing

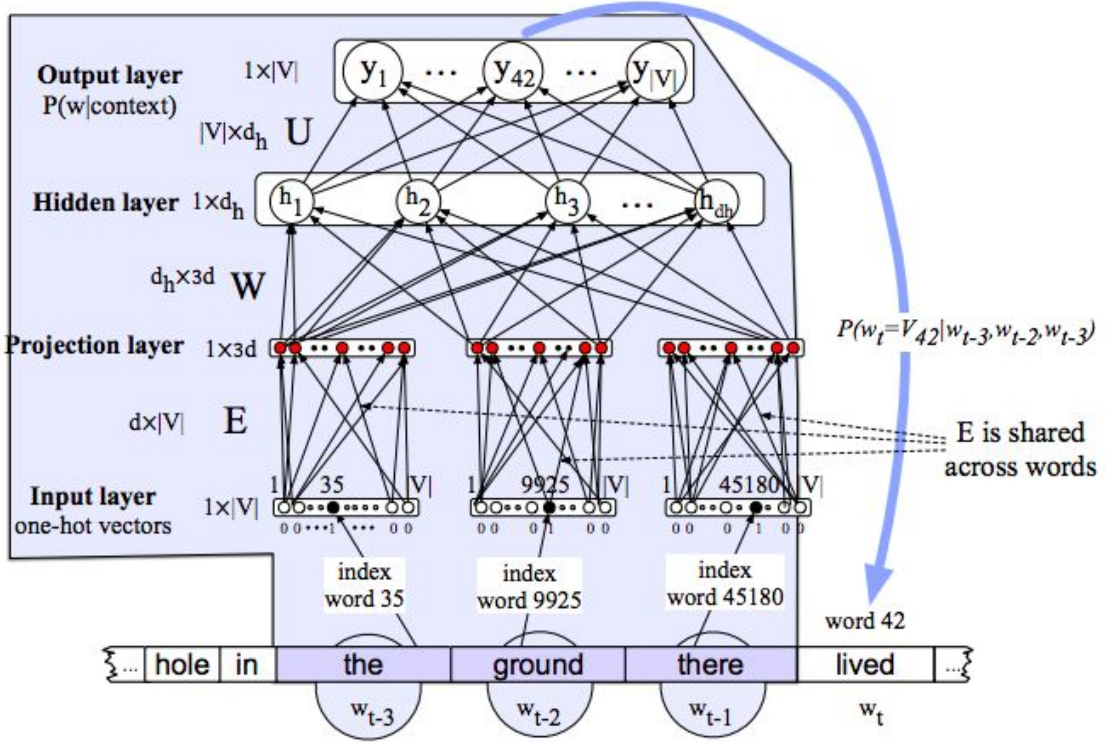
neural network language model

Task: predict w_t after a sequence of words w_{t-3} w_{t-2} w_{t-1}

Feedforward NN classifier

Input: word indexes
Output: index of the next word

V: vocabulary size
d: embedded word size



neural network language model

Network:

1) transform word w_i to one hot representation x_i
2) $e = (Ex_1, Ex_2, \ldots, Ex_k)$ ← learned **embedding** matrix
3) h = activation(We + b)
4) z = Uh
5) y = softmax(z)

softmax calculates probability distribution

$$softmax(x) = \left(\frac{e^{x_1}}{\sum_{i=1}^{C} e^{x_i}}, \ldots, \frac{e^{x_C}}{\sum_{i=1}^{C} e^{x_i}} \right)$$

Training: backpropagation

Loss: categorical cross entropy

Optimisation algorithm: stochastic gradient descent

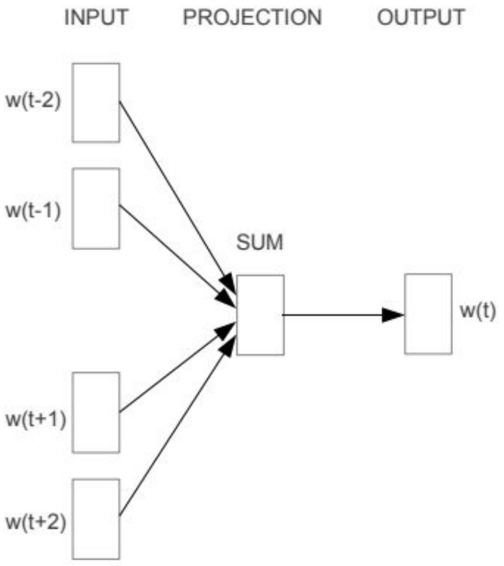
A Neural Probabilistic Language Model. Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, JMLR, **2003**.

word2vec approach

- **Inspection:** most of the complexity comes from the connection of the projection layer and the hidden layer as projections are dense
- feedforward neural networks without hidden layer:
  - input layer
  - projection layer
  - output layer
- increase the amount of training data
- the plan is to use the **embedding matrix**, not to predict words as it might be expected

Continuous Bag of Words (CBOW)

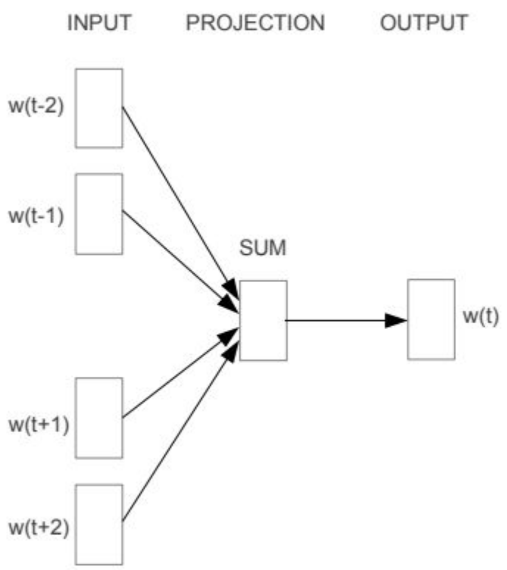
Task:
build a log-linear classifier that can correctly classify middle word for the given context



CBOW

Continuous Bag of Words (CBOW)

- input: context words
  - one-hot representations
- output: hierarchical softmax
  - vocabulary is presented as Huffman binary tree
- projection layer:
  - it is shared as well as projection matrix - projected values are averaged
  - the order of words in the history does not influence the projection



CBOW

complexity per training example: $N \times D + D \times \log_2(V)$

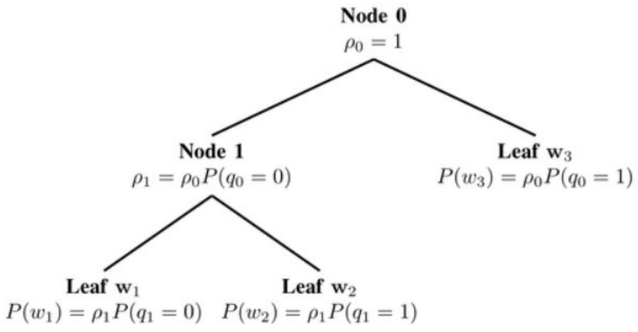
N - context size, D - embeddings dimensions, V - vocabulary size

hierarchical softmax

- Softmax:

$$softmax(x) = \left(\frac{e^{x_1}}{\sum_{i=1}^{C} e^{x_i}}, \dots, \frac{e^{x_C}}{\sum_{i=1}^{C} e^{x_i}} \right)$$

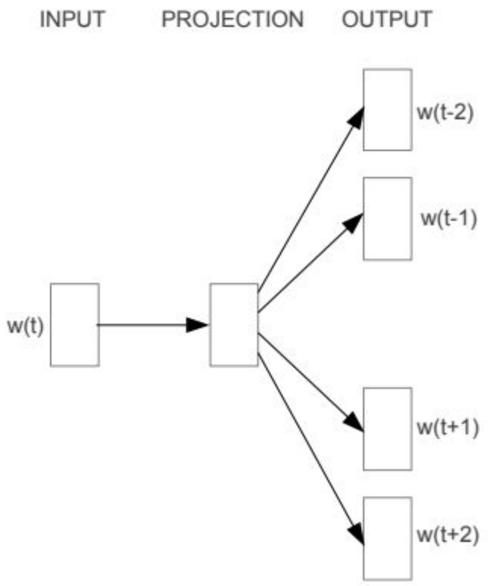
- Computation complexity is O(V) ⟵ C=V
- Vocabulary is presented as a Huffman binary tree
- Hierarchical Softmax:
  decompose calculating the probability of one word into a sequence of probability calculations
- Computational complexity $O(\log_2(V))$



Continuous skip-gram model

- **Task:**
  build a log-linear classifier that can correctly classify neighbour words for the given center word

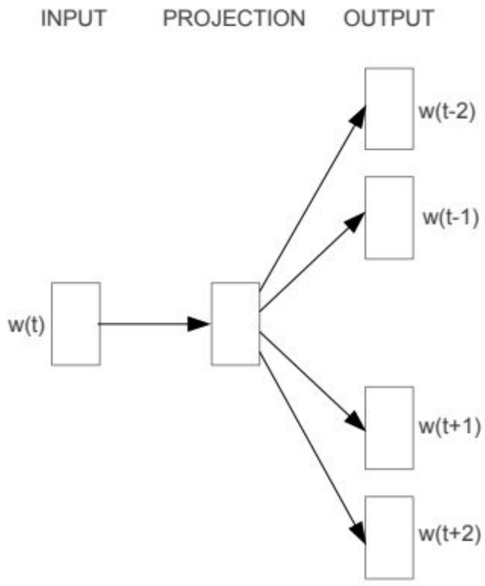
- Precisely:
  for the center word and a given new word network will give us the probability of “a new word is a neighbour word” property



Skip-gram

Continuous skip-gram model

- input: word
  - one-hot representations
- output: hierarchical softmax
  - vocabulary is presented as Huffman binary tree
- projection layer:
  - used for word embeddings



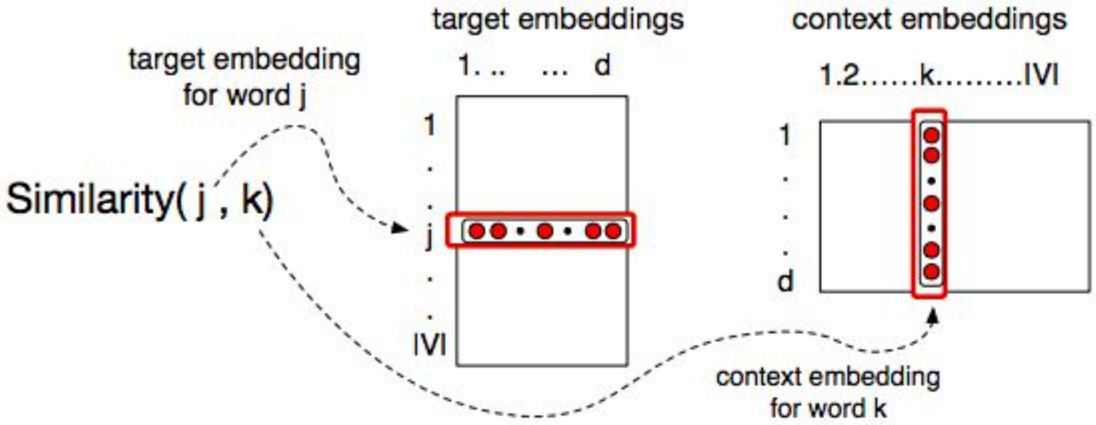
Skip-gram

complexity per training example: $C \times (D + D \times \log_2(V))$

C - maximum distance of the words, D - embeddings dimensions, V - vocabulary size

Continuous skip-gram model

This model learn two matrices: **embedding** matrix and **context** matrix



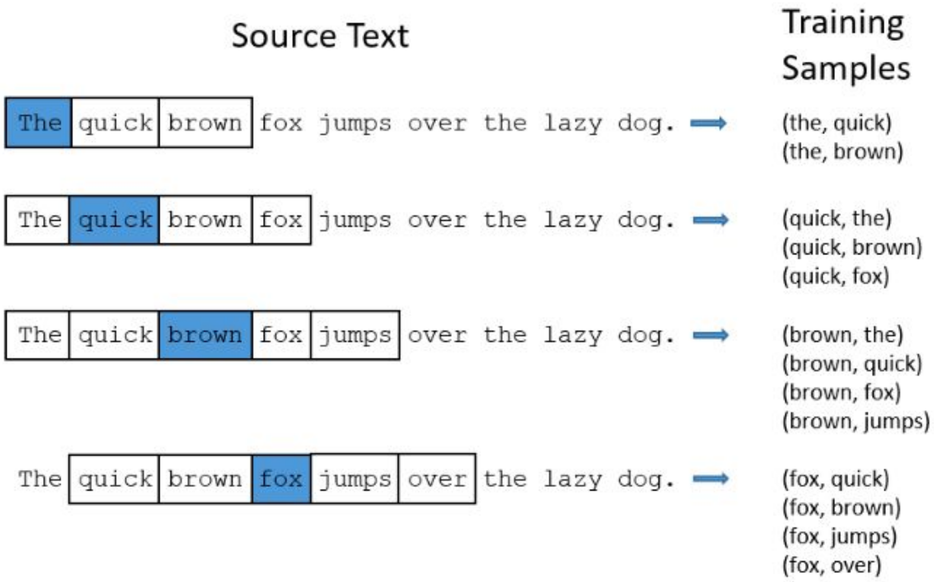
For center word w_j we compute:

$$p(w_k|w_j) = \frac{exp(c_k \cdot v_j)}{\sum_{i \in |V|} exp(c_i \cdot v_j)}$$

neural network training

- For every positive sample,
  we use some number of
  “negative” samples: samples
  we would like the network to predict
  value 0
  for example: quick, sheep

- Some recommended values
  for the number of negative samples
  5 to 20
- “unigram” table with frequencies



neural network training

- Backpropagation
- Stochastic gradient descent
  - start learning rate 0.025 and decrease it lineary
- Large scale parallel training of models is done by distributed framework called **DistBelief**

evaluation of word embeddings quality

Question:

”What is the word that is similar to X in the same sense as Y is similar to Z?”

Result is obtained by simple algebraic operations:

vector closest to the vector(Z)−vector(Y) + vector(X)

For example:

X = small, Y = big, Z = bigger

r = vector(”bigger”)−vector(”big”) + vector(”small”)

search for embedding that is closest (cosine metric) to r gives “smaller”

evaluation of word embeddings quality

- 5 types of semantic questions
- 9 types of syntactic questions

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

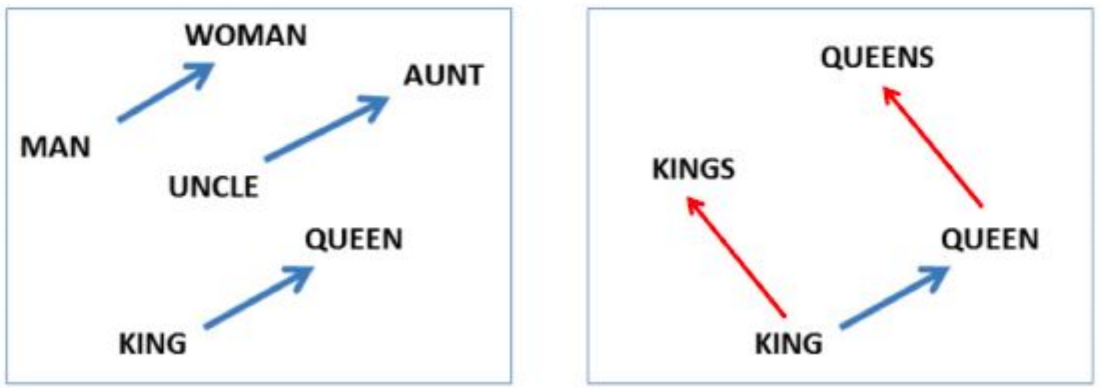
In total: 8869 semantic and 10675 syntactic questions

evaluation of word embeddings quality

Vectors of various sizes on various dataset are learned and evaluated.

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

properties of word embeddings



embedding(‘kings’)-embedding(‘king’)+embedding(‘queen’) → embedding(‘queens’)

properties of word embeddings

Examples of learned relationships:

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

improvements

- **phrases** are included:
  - New York, Montreal Canadiens, …
  - In total: 3 million of new words
- data-driven approach is used for phrases extraction
  - (pab - min_count) / (pa * pb) where pa, pb, and pab are the number of occurrences of words a, b, and their combination
  - min_count is a predefined value used for elimination of very infrequent phrases
- phrases are treated as individual tokens
  - New_York, Montreal_Canadiens, ...

improvements

- subsampling of frequent words
  - some words appear more often than other words but do not contribute to semantic the, a, in, …
  - exclusion of these word will improve the balance of rare and frequent word as well as speed up the training (according to results from 2x to 10x)
- $z(w_i)$ is the fraction of total words in the corpus that are w_i
  for example, if *peanut* occurs 1,000 times in a 1 billion word corpus, then z('peanut') = 1E-6
- The probability of keeping the word w_i

$$P(w_i) = (\sqrt{\frac{z(w_i)}{0.001}} + 1) \cdot \frac{0.001}{z(w_i)}$$

- $P(w_i) = 1$ when $z(w_i) <= 0.0026$ → words which represent more than 0.26% of the total words will be subsampled.

comparison

- The size of embeddings: 300
- NEG-k: negative sampling with k negative samples per positive sample
- NSE: Noise Contrastive Estimation
- HS-Huffman: Hierarchical Softmax

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

Thank you!