word2vec

Tom Kenter
20142015.Information Retrieval 1
January 13 2014

word2vec

semantics

word embeddings

Word2vec: what is it (not)?

It is:

- Word embeddings = vectors representation for words
- Mapping to semantic/syntactic space
- Co-occurrence-based
- Neural network approach
- Introduced by Tomas Mikolov (then at Google)

It is ***not***:

- Deep learning

Word embeddings

Vector representation of a word

N dimensions

vector("nice") = <0.12 0.432 0.2424 … … 0.65 0.43>

vector("good") = <0.11 0.322 0.204 … … 0.53 0.393>

…

V words

…

vector("Paris") = <0.67 0.101 0.74 … … 0.303 0.112>

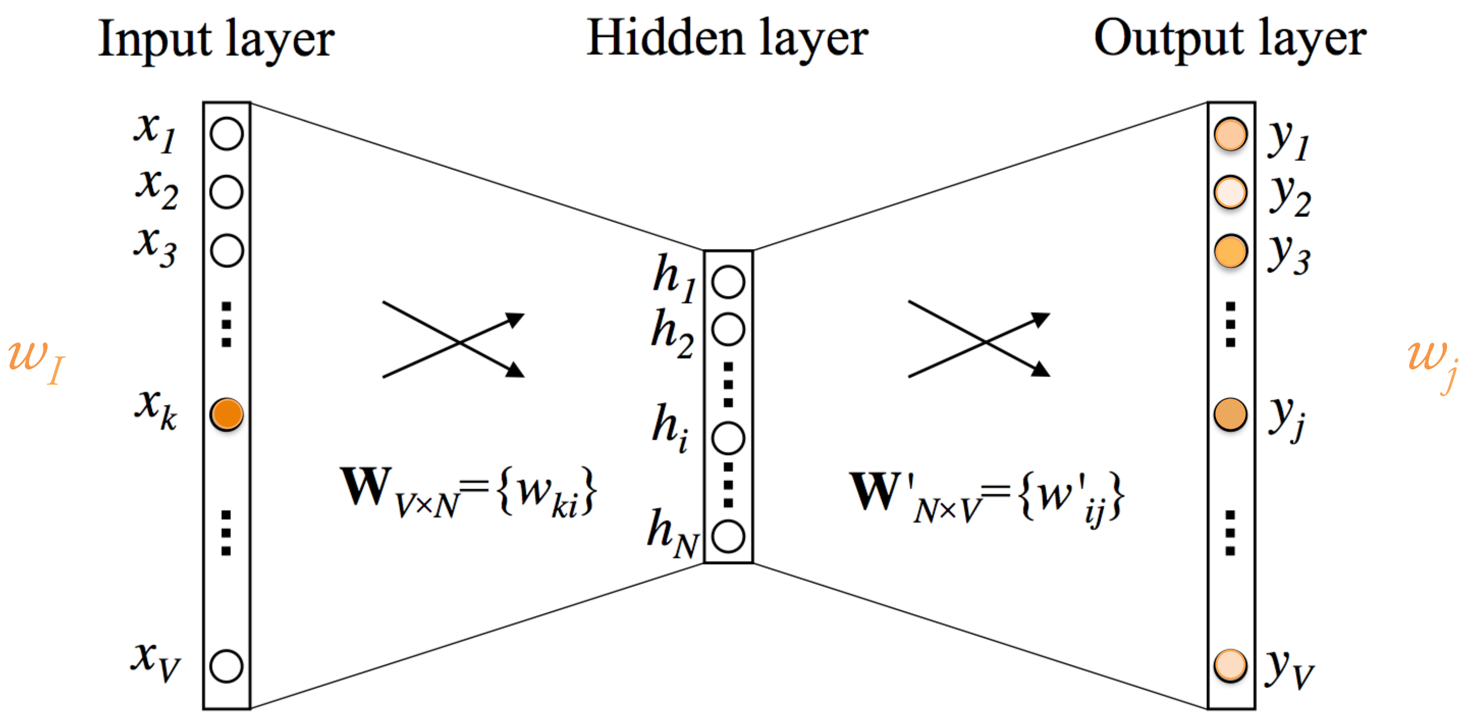
vector("France") = <0.74 0.007 0.568 … … 0.23 0.102>

Vector similarity

Cosine similarity between vectors A and B

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Neural network language models



$$\boxed{p(w_j|w_I)} = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V}\exp(u_{j'})}$$

Soft max

$$u_j = {\mathbf{v}'_{w_j}}^T \cdot \mathbf{h}$$

Figure from [6]

Continuous Bag Of Word architecture

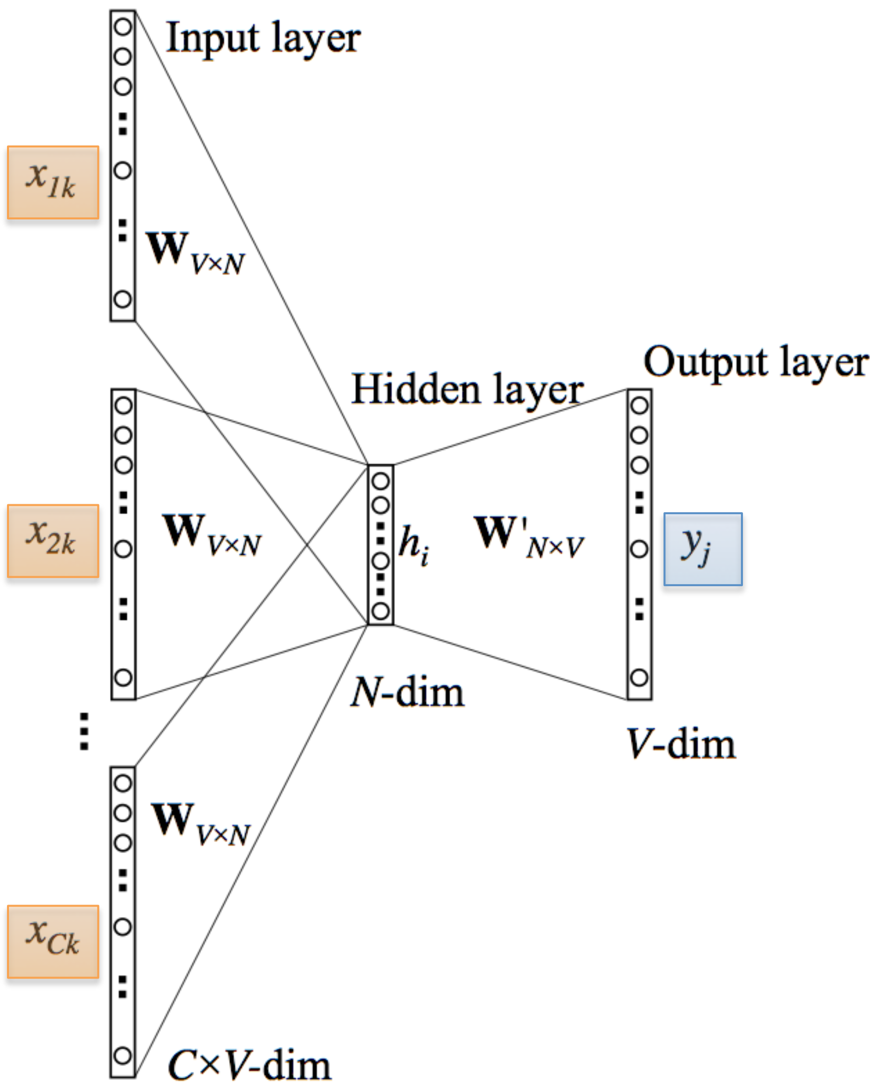
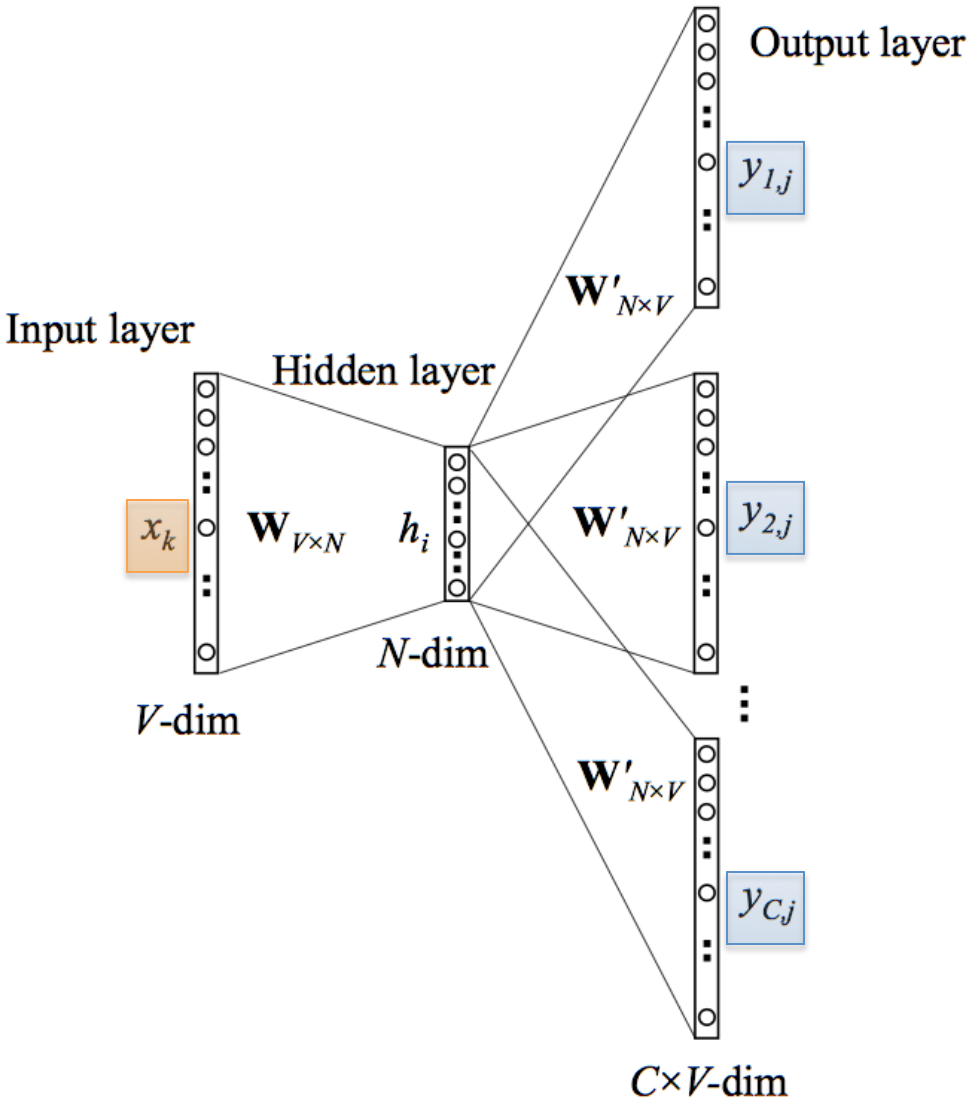


Figure from [6]

Skip-gram architecture



- Hierarchical softmax
- Negative sampling

Figure from [6]

In short: two architectures

CBOW Skip-gram

- **CBOW**
  Continuous Bag Of Words
  Predict a missing word given a window of context words

- **Skip-gram**
  Predict context words given a word

Two architectures

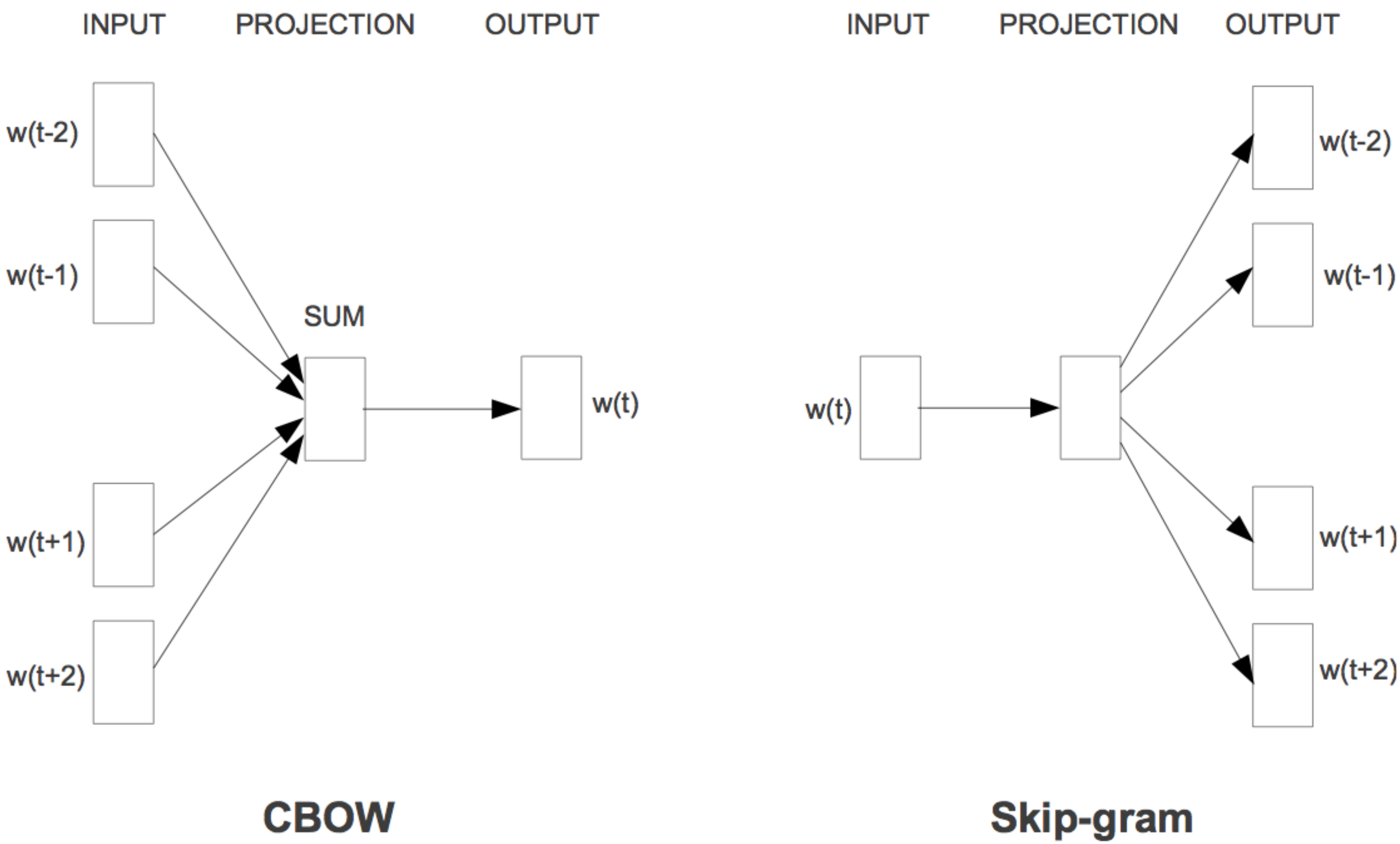


Figure from [4]

Where are the word embeddings?

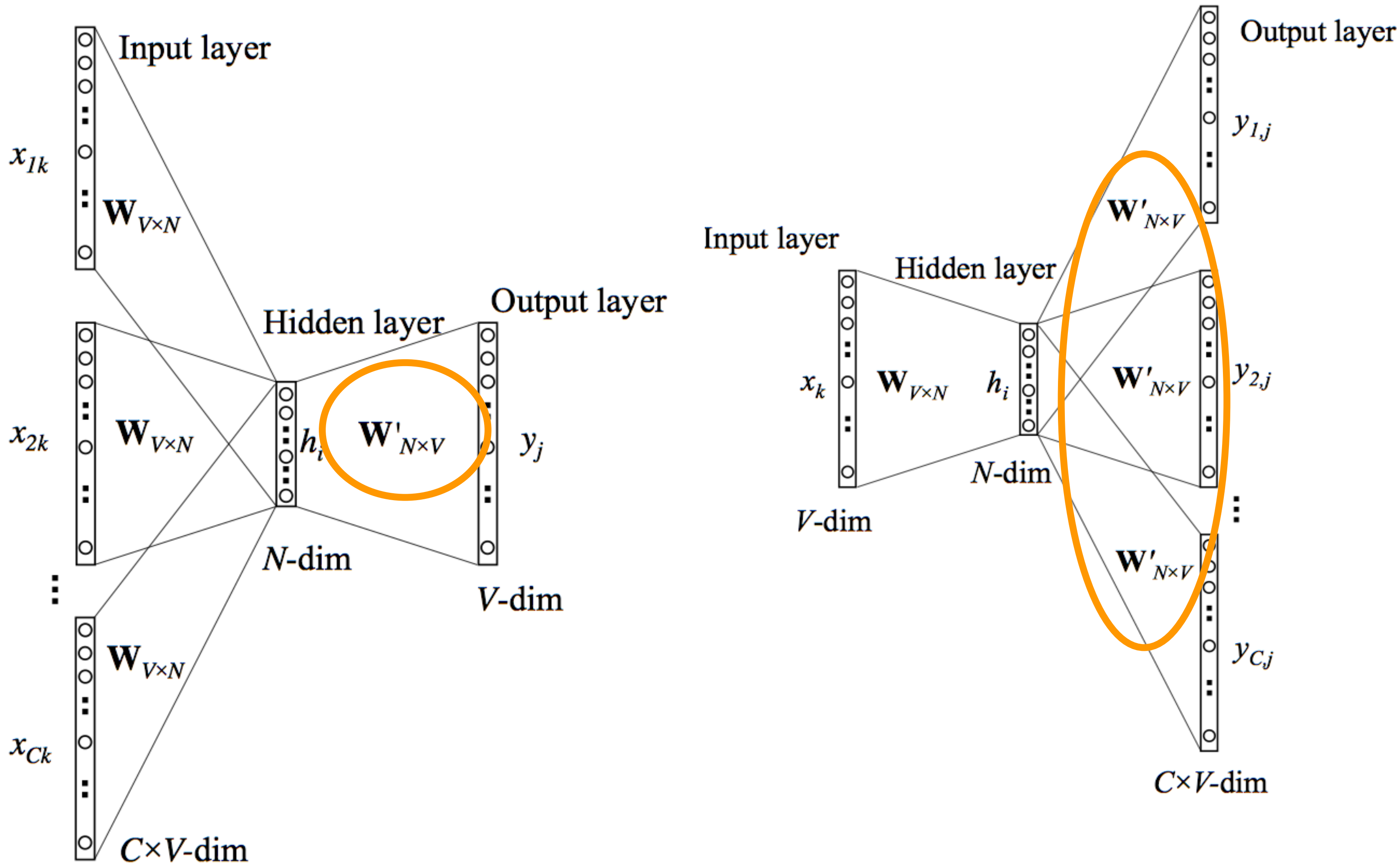


Figure from [6]

Advantages

- It scales
  - Train on billion word corpora
  - In limited time
  - Mikolov mentions parallel training
- Word embeddings trained by one can be used by others
  - For entirely different tasks
  - This happens indeed!
- Incremental training
  Train on one piece of data, save results, continue training later on

- The code is available
  - Original source code from Mikolov: https://code.google.com/p/word2vec/
  - There is a very nice Python module for it!
    Gensim word2vec: http://radimrehurek.com/gensim/models/word2vec.html

What can you do with it?

A is to B as C is to ?

This is the famous example:

vector(*king*) – vector(*man*) + vector(*woman*)

=

vector(*queen*)

Actually, what the original paper says is: if you substract the vector for ‘man’ from the one for ‘king’ and add the vector for ‘woman’, the <u>vector closest to the one you end up with</u> turns out to be the one for ‘queen’.

What can you do with it?

A is to B as C is to ?

It also works for syntactic relations:

X = vector(*biggest*) - vector(*big*) + vector(*small*).

X = vector(*smallest*)

What can you do with it?

A is to B as C is to ?

More interesting: which country does a particular city belong to?

France is to *Paris* as *Germany* is to *Berlin*.

Why does this happen?

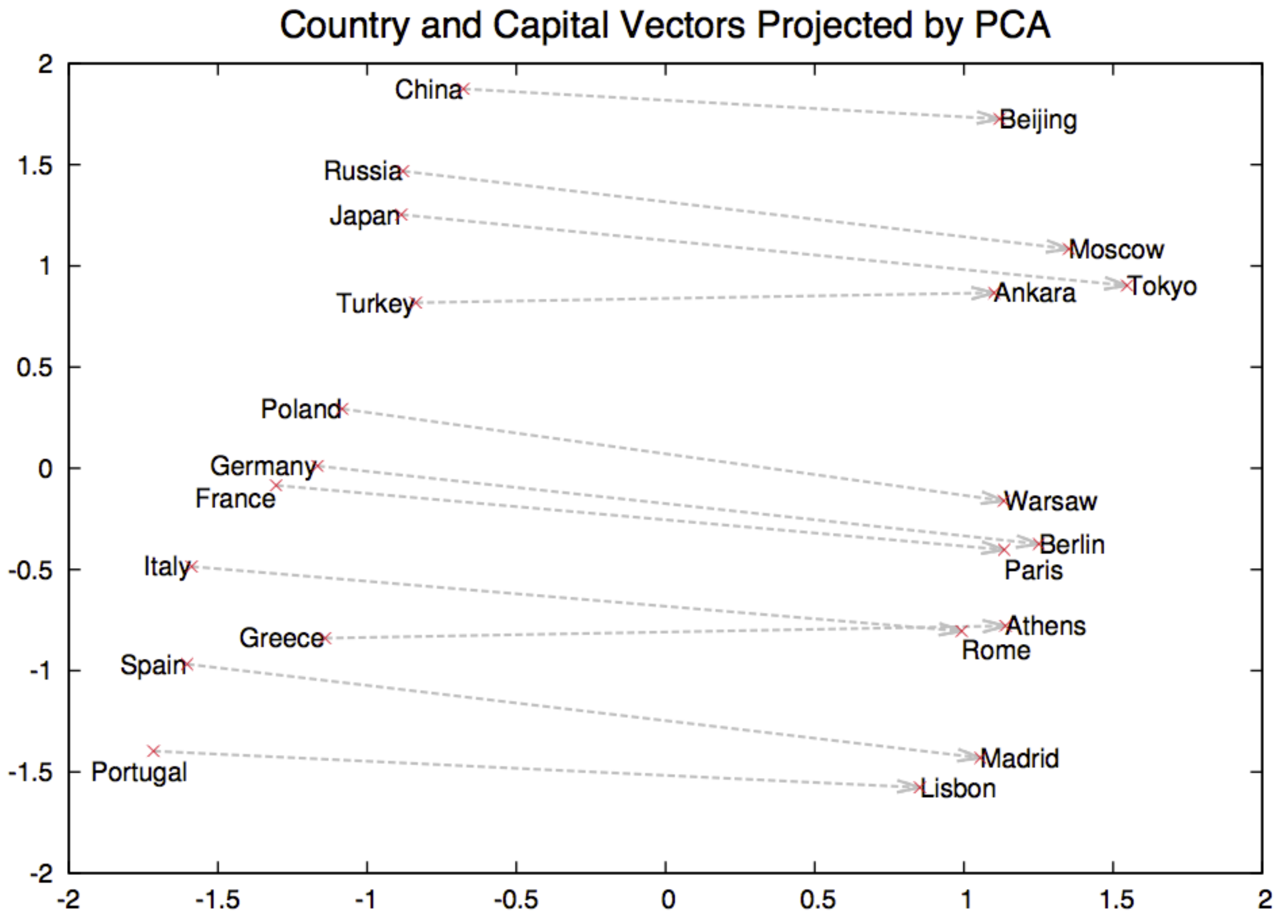


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised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

Figure from [5]

Evaluation

- Mikolov presents an evaluation set in [4]

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

Semantic
8869
of these

Syntactic
10675
of these

Table from [4]

Evaluation

Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

Table from [4]

What can you do with it?

- Sentence completion
  Microsoft Sentence Completion Challenge
- Selecting out-of-the-list words
  Example: which word does not belong in
  [*monkey, lion, dog, truck]*
- Bilingual Word Embeddings for Phrase-Based Machine Translation. *EMNLP* – 2013.
- Synonym detection
  Choose among 4 candidates which one is the best synonym for a given word.

What can you do with it?

- Concept categorization
  Group *helicopter* and *motorcycle* together, and *dog* an *elephant*
- Selectional preference
  Predict typical verb noun pairs. Like *people* go well with to *eat* as a subject (but not as an object).

Seeing changes in word meaning/usage over time

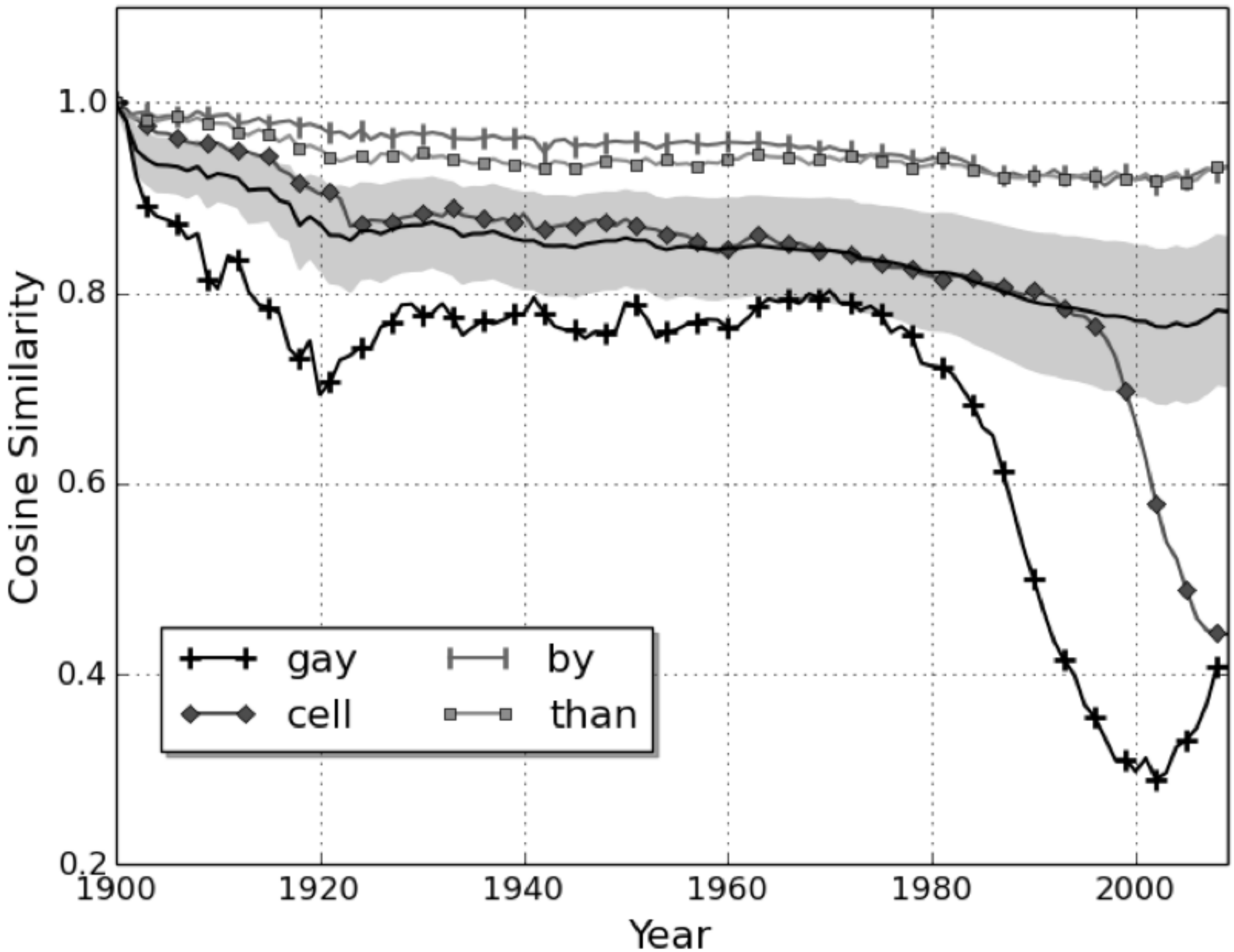


Figure from [3]

Challenge: dealing with phrases

- Deal with this during pre-processing step
  Find collocations the usual way, and feed them as single 'words' to the NN

- Just add/average/concatenate the vectors
  This is better than the previous approach if you happen to be dealing with entities like *Bengal tiger* versus *Sumatran Tiger*

- Paragraph2vec
  Make fixed-length representation of ***variable***-length input

- Build a convolutional NN on top of word2vec
  This is done in [2] for sentence classification

Disadvantages/challenges/future research

- Ambiguity
  What if there are two Amsterdams..???

- Evaluation is only extrinsic
  (but this goes for rankers as well)

- Parameter setting involves magic/luck/trial and error
  There is no such thing as free lunch

- Very picky on literal words
  But this affects all other word-based methods as well

- Can not deal with OOV words
  If a word is met at testing time that was not seen at training time, it will simply be ignored.

Who are the associated names

- Mikolov (Google)
- Bengio (of Deep Learning fame)
  He already proposed NN LMs and word embeddings a decade ago

- There are other algorithms for getting word embeddings. Most notably:

  GloVe: Global Vectors for Word Representation
  Jeffrey Pennington, Richard Socher, Christopher D. Manning – 2014
  All from Stanford

(By no means extensive) list of papers

Overview and evaluation of many different tasks
[1] M. Baroni, G. Dinu, and G. Kruszewski. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, volume 1, 2014

Sentence classification
[2] Y. Kim, Convolutional neural networks for sentence classification, 2014

This is where the 'gay' and 'cell' example came from
[3] Y. Kim, Y.-I. Chiu, K. Hanaki, D. Hedge, and S. Petrov. Temporal analysis of language through neural language models, 2014

For more elaborate discussion on the architectures used
[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013

More on Skip-gram, negative sampling, hierarchical softmax, and dealing with phrases
[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119, 2013

Very nice detailed explanation of how word2vec works, with derivations of the back propagation equations, etc.
[6] Xin Rong, "word2vec Parameter Learning Explained." *arXiv preprint arXiv:1411.2738*, 2014

Link to a demo (scroll down a bit to where it says 'Bonus App')
http://radimrehurek.com/2014/02/word2vec-tutorial/