What is Word2Vec?

Traian Rebedea

Bucharest Machine Learning reading group

Intro

- About n-grams: “simple models trained on huge amounts of data outperform complex systems trained on less data”
- Solution: “possible to train more complex models on much larger data set, and they typically outperform the simple models”
- Why? “neural network based language models significantly outperform N-gram models”
- How? “distributed representations of words” (Hinton, 1986 – not discussed today)

Goal

- “learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary”
- Resulting word representations
  - Similar words tend to be close to each other
  - Words can have multiple degrees of similarity

Previous work

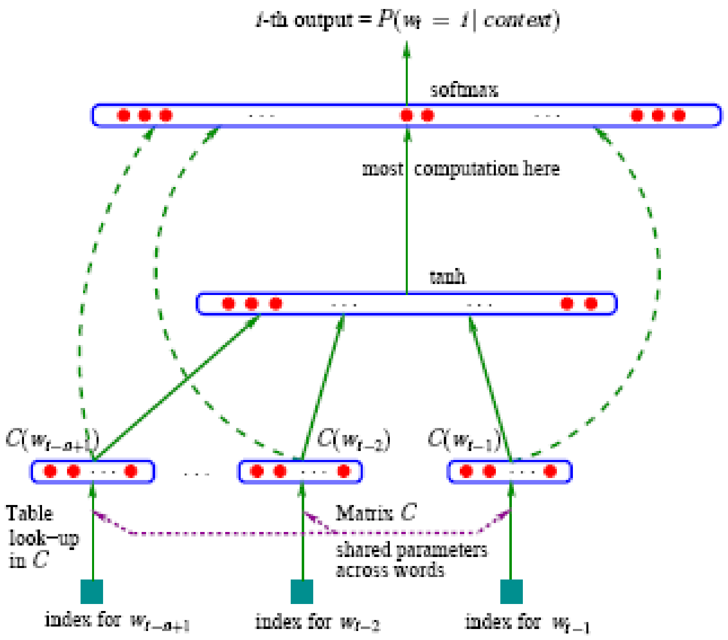
- Representation of words as continuous vectors
- Neural network language model (NNLM) (Bengio et al., 2003 – not discussed today)
- Mikolov previously proposed (MSc thesis, PhD thesis, other papers) to first learn word vectors “using neural network with a single hidden layer” and then train the NNLM independently
- Word2vec directly extends this work => “word vectors learned using a simple model”
- These word vectors were useful in various NLP applications
- Many architectures and models have been proposed for computing these word vectors (e.g. see Socher’s Stanford group work which resulted in GloVe - http://nlp.stanford.edu/projects/glove/)
- “these architectures were significantly more computationally expensive for training than” word2vec (in 2013)

Model Architectures

- Some “classic” NLP for estimating continuous representations of words
  - LSA (Latent Semantic Analysis)
  - LDA (Latent Dirichlet Allocation)
- Distributed representations of words learned by neural networks outperform LSA on various tasks that require to preserve linear regularities among words
- LDA is computationally expensive and cannot be trained on very large datasets

Model Architectures

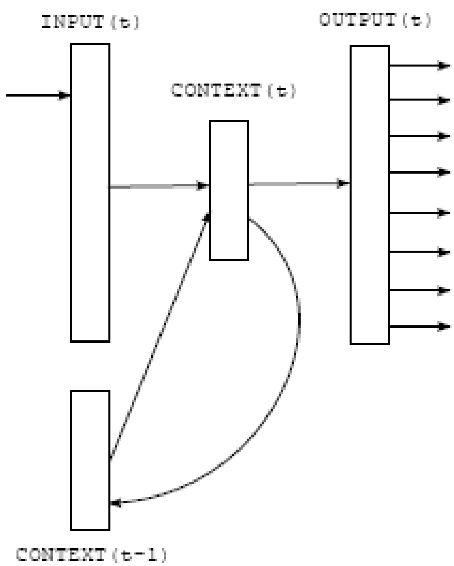
- Feedforward Neural Net Language Model (NNLM)



$$Q = N \times D + N \times D \times H + H \times V,$$

Model Architectures

- Recurrent Neural Net Language Model (RNNLM)
- Simple Elman RNN



$$Q = H \times H + H \times V,$$

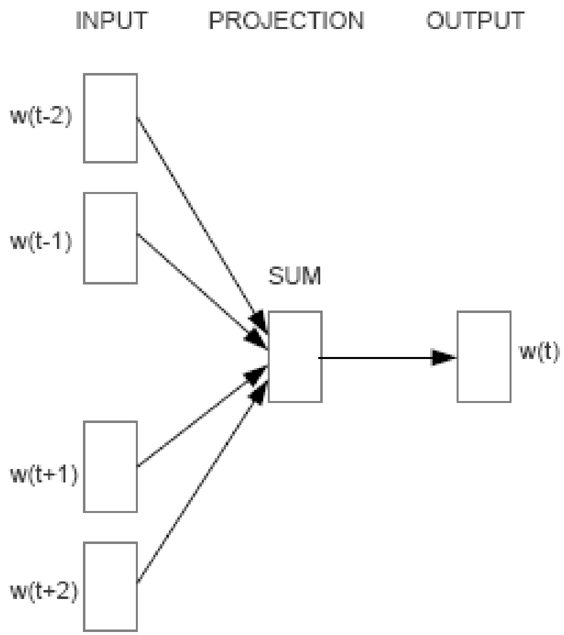
Word2vec (log-linear) Models

- Previous models - the complexity is in the non-linear hidden layer of the model
- Explore simpler models
  - Not able to represent the data as precisely as NN
  - Can be trained on more data
- In earlier works, Mikolov found that “neural network language model can be successfully trained in two steps”:
  - Continuous word vectors are learned using simple model
  - The N-gram NNLM is trained on top of these distributed representations of words

Continuous BoW (CBOW) Model

- Similar to the feedforward NNLM, but
- Non-linear hidden layer removed
- Projection layer shared for all words
  - Not just the projection matrix
- Thus, all words get projected into the same position
  - Their vectors are just averaged
- Called CBOW (continuous BoW) because the order of the words is lost
- Another modification is to use words from past and from future (window centered on current word)

CBOW Model



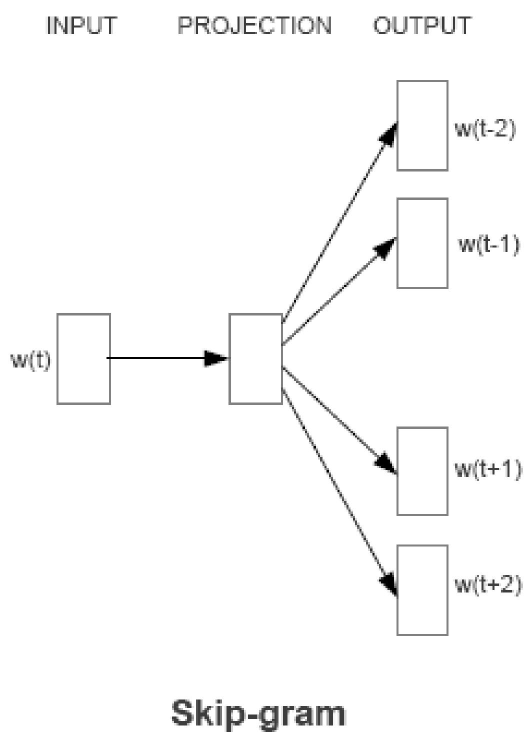
CBOW

$$Q = N \times D + D \times log_2(V).$$

Continuous Skip-gram Model

- Similar to CBOW, but instead of predicting the current word based on the context
- Tries to maximize classification of a word based on another word in the same sentence
- Thus, uses each current word as an input to a log-linear classifier
- Predicts words within a certain window
- Observations
  - Larger window size => better quality of the resulting word vectors, higher training time
  - More distant words are usually less related to the current word than those close to it
  - Give less weight to the distant words by sampling less from those words in the training examples

Continuous Skip-gram Model



$$Q = C \times (D + D \times log_2(V)),$$

Results

- Training high dimensional word vectors on a large amount of data captures "subtle semantic relationships between words"
  - Mikolov has made a similar observation for the previous models he has proposed (e.g. the RNN model, see Mikolov, T., Yih, W. T., & Zweig, G. (2013, June). Linguistic Regularities in Continuous Space Word Representations. In *HLT-NAACL* (pp. 746-751).)

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

Results

- “Comprehensive test set that contains five types of semantic questions, and nine types of syntactic questions”
  - 8869 semantic questions
  - 10675 syntactic questions
- E.g. “For example, we made a list of 68 large American cities and the states they belong to, and formed about 2.5K questions by picking two word pairs at random.”
- Methodology
- Input: ”What is the word that is similar to small in the same sense as biggest is similar to big?”
- Compute: X = vector("biggest") – vector("big") + vector("small") and then find closest word to X using cosine

Results

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

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

- Other results are reported as well

Skip-gram Revisited

- Formally, the skip-gram model proposes that for a give sequence of words to maximize:

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

- Where T = size of the sequence (or number of words considered for training)
- c = window/context size
- Mikolov, also says that for each word the model only uses a random window size r = random(1..c)
  - This way words that are closer to the “input” word have a higher probability of being used in training than words that are more distant

Skip-gram Revisited

- As already seen, $p(w_{t+j}|w_t)$ should be the output of a classifier (e.g. softmax)

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

- w_I is the "input" vector representation of word w
- w_O is the "output" (or "context") vector representation of word w
- Computing $\log p(w_O|w_I)$ takes $O(W)$ time, where W is the vocabulary dimension

Skip-gram Alternative View

- Training $$\arg\max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta)$$
  – Where $$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}$$
- Getting to $$\arg\max_{\theta} \sum_{(w,c)\in D} \log p(c|w) = \sum_{(w,c)\in D} (\log e^{v_c \cdot v_w} - \log \sum_{c'} e^{v_{c'} \cdot v_w})$$

Skip-gram Improvements

- Hierarchical softmax
- Negative sampling
- Subsampling of frequent words

Hierarchical Softmax

- Computationally efficient approximation of the softmax
- When W output nodes, need to evaluate only about log(W) nodes to obtain the softmax probability distribution

Negative Sampling

- Noise Contrastive Estimation (NCE) is an alternative to hierarchical softmax
- NCE – "a good model should be able to differentiate data from noise by means of logistic regression"
- "While NCE can be shown to approximately maximize the log probability of the softmax, the Skip-gram model is only concerned with learning high-quality vector representations, so we are free to simplify NCE as long as the vector representations retain their quality. We define Negative sampling (NEG) by the objective":

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

Subsampling of Frequent Words

- Frequent words provide less information value than the rare words
- "More, the vector representations of frequent words do not change significantly after training on several million examples"
- Each word w_i in the training set is discarded with a probability depending on the frequency of the word

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

Other remarks

- Mikolov also developed a method to extract relevant n-grams (bigrams and trigrams) using something similar to PMI
- Effects of improvements

Method	Dimensionality	No subsampling [%]	10^{-5} subsampling [%]
NEG-5	300	24	27
NEG-15	300	27	42
HS-Huffman	300	19	47

- Vectors can also be summed

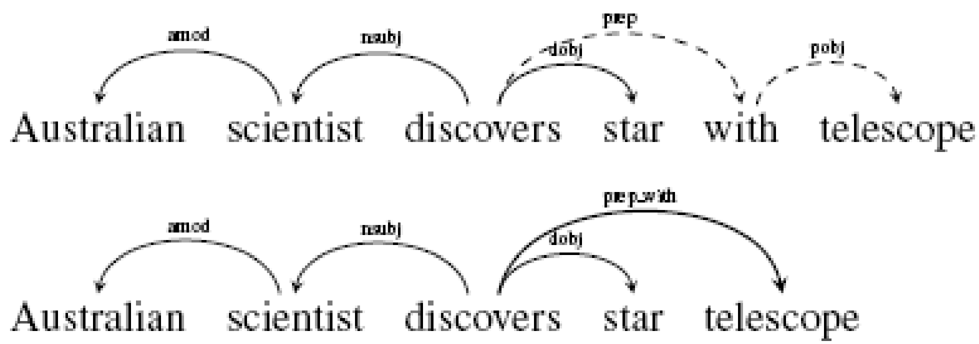
Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese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

Other Applications

- Dependency-based contexts
- Word2vec for machine learning translation

Dependency-based Contexts

- Levi and Goldberg, 2014: Propose to use dependency-based contexts instead of linear BoW (windows of size k)



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Dependency-based Contexts

- Why?
  - Syntactic dependencies are “more inclusive and more focused” than BoW
  - Capture relations to words that are far apart and that are not used by small window BoW
  - Remove “coincidental contexts which are within the window but not directly related to the target word”
- A possible problem
  - Dependency parsing is still somewhat computational expensive
  - However, English Wikipedia can be parsed on a small cluster and the results can then be persisted

Dependency-based Contexts

- Examples of syntactic contexts

batman	hogwarts	turing
superman/conj^{-1}	students/prep_at^{-1}	machine/nn^{-1}
spider-man/conj^{-1}	educated/prep_at^{-1}	test/nn^{-1}
superman/conj	student/prep_at^{-1}	theorem/poss^{-1}
spider-man/conj	stay/prep_at^{-1}	machines/nn^{-1}
robin/conj	learned/prep_at^{-1}	tests/nn^{-1}
florida	**object-oriented**	**dancing**
marlins/nn^{-1}	programming/amod^{-1}	dancing/conj
beach/appos^{-1}	language/amod^{-1}	dancing/conj^{-1}
jacksonville/appos^{-1}	framework/amod^{-1}	singing/conj^{-1}
tampa/appos^{-1}	interface/amod^{-1}	singing/conj
florida/conj^{-1}	software/amod^{-1}	ballroom/nn

Dependency-based Contexts

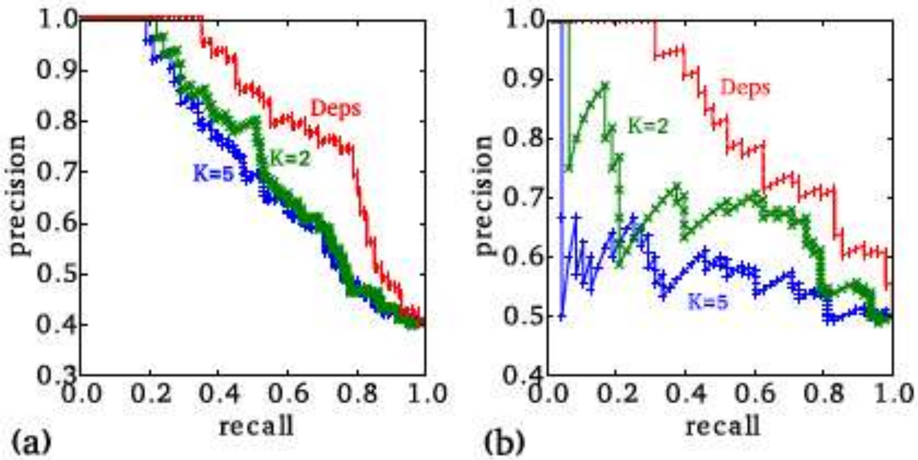
- Comparison with BoW word2vec

Target Word	BoW5	BoW2	Deps
batman	nightwing aquaman catwoman superman manhunter	superman superboy aquaman catwoman batgirl	superman superboy supergirl catwoman aquaman
hogwarts	dumbledore hallows half-blood malfoy snape	evernight sunnydale garderobe blandings collinwood	sunnydale collinwood calarts greendale millfield
turing	nondeterministic non-deterministic computability deterministic finite-state	non-deterministic finite-state nondeterministic buchi primality	pauling hotelling heting lessing hamming

florida	gainesville fla jacksonville tampa lauderdale	fla alabama gainesville tallahassee texas	texas louisiana georgia california carolina
object-oriented	aspect-oriented smalltalk event-driven prolog domain-specific	aspect-oriented event-driven objective-c dataflow 4gl	event-driven domain-specific rule-based data-driven human-centered
dancing	singing dance dances dancers tap-dancing	singing dance dances breakdancing clowning	singing rapping breakdancing miming busking

Dependency-based Contexts

- Evaluation on the WordSim353 dataset with pairs of similar words
  - Relatedness (topical similarity)
  - Similarity (functional similarity)
  - Both (these pairs have been ignored)
- Task: “rank the similar pairs in the dataset above the related ones”
- Simple ranking: Pairs ranked by cosine similarity of the embedded words



Dependency-based Contexts

- Main conclusion
- Dependency-based context is more useful to capture functional similarities (e.g. similarity) between words
- Linear BoW context is more useful to capture topical similarities (e.g. relatedness) between words
  - The larger the size of the window, the better it captures related concepts
- Therefore, dependency-based contexts would perform poorly in analogy experiments

Estimating Similarities Across Languages

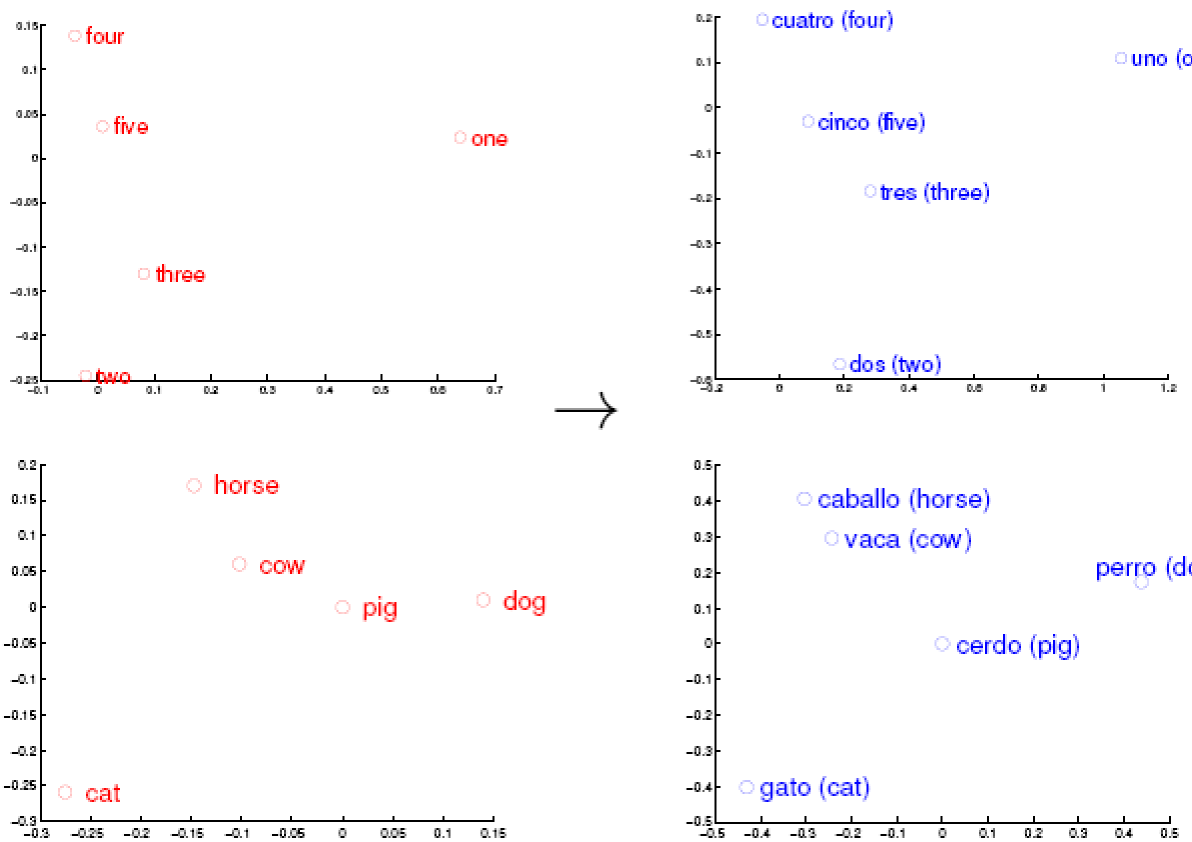
- Given a set of word pairs in two languages (or different types of corpora) and their associated vector representations (x_i and z_i)
- They can have even different dimensions (d_1 and d_2)
- Find a transformation matrix, $W(d_2, d_1)$, such that Wx_i approximates as close as possible z_i, for all pairs i

$$\min_W \sum_{i=1}^{n} \|Wx_i - z_i\|^2$$

- Solved using stochastic gradient descent
- The transformation is seen as a linear transformation (rotation and scaling) between the two spaces

Estimating Similarities Across Languages

- Authors also highlight this using a manual rotation (between En and Sp) and a visualization with 2D-PCA



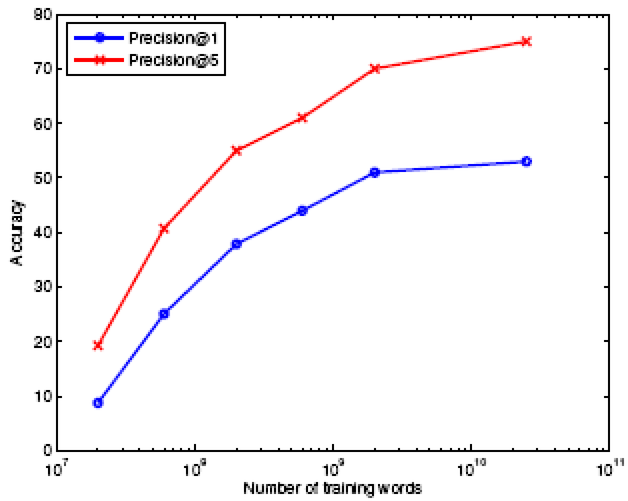
Estimating Similarities Across Languages

- The most frequent 5K words from the source language and their translations given GT – training data for learning the Translation Matrix
- Subsequent 1K words in the source language and their translations are used as a test set

Estimating Similarities Across Languages

- Very simple baselines

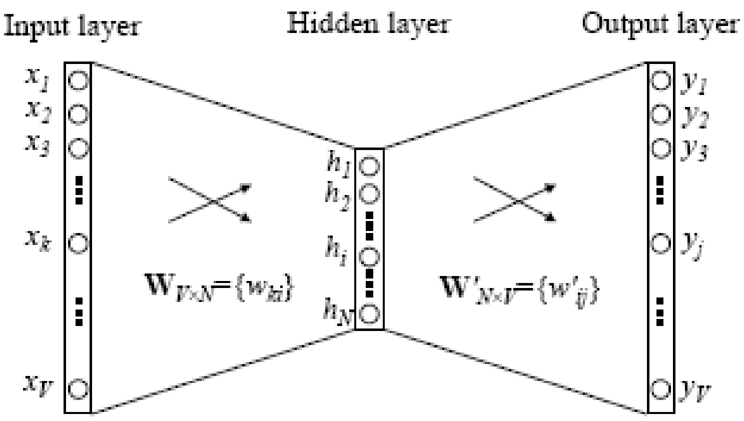
Translation	Edit Distance		Word Co-occurrence		Translation Matrix		ED + TM		Coverage
	P@1	P@5	P@1	P@5	P@1	P@5	P@1	P@5	
En $\rightarrow$ Sp	13%	24%	19%	30%	33%	51%	43%	60%	92.9%
Sp $\rightarrow$ En	18%	27%	20%	30%	35%	52%	44%	62%	92.9%
En $\rightarrow$ Cz	5%	9%	9%	17%	27%	47%	29%	50%	90.5%
Cz $\rightarrow$ En	7%	11%	11%	20%	23%	42%	25%	45%	90.5%



More Explanations

- CBOW model with a single input word

$$\mathbf{h} = \mathbf{x}^T \mathbf{W} = \mathbf{W}_{(k,\cdot)} := \mathbf{v}_{w_I},$$



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

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

$$p(w_j|w_I) = \frac{\exp\left({\mathbf{v}'_{w_O}}^T \mathbf{v}_{w_I}\right)}{\sum_{j'=1}^{V} \exp\left({\mathbf{v}'_{w'_j}}^T \mathbf{v}_{w_I}\right)}$$

Update Equations

- Maximize the conditional probability of observing the actual output word w_O (denote its index in the output layer as j) given the input context word w_I with regard to the weights

$$\begin{aligned}\max p(w_O|w_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E,\end{aligned}$$

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j$$

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathrm{EH}_i$$

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{u_j}{\partial w'_{ij}} = e_j \cdot h_i$$

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_{ki}} = \mathrm{EH}_i \cdot x_k$$

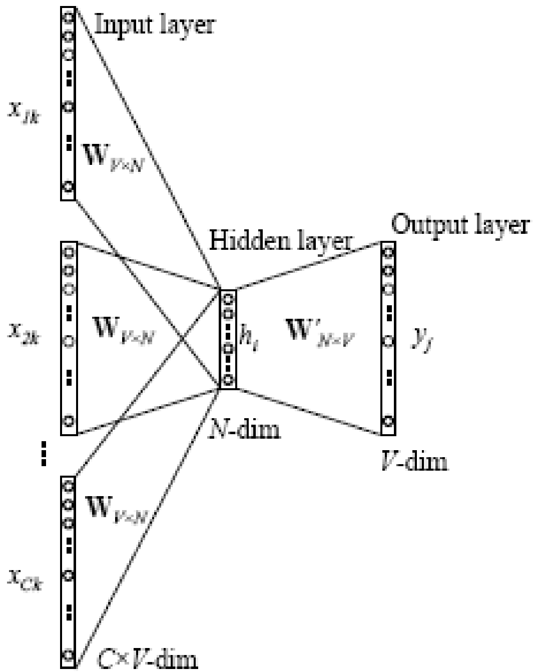
$$\frac{\partial E}{\partial \mathbf{W}} = \mathrm{x} \cdot \mathrm{EH}$$

$${w'_{ij}}^{(\mathrm{new})} = {w'_{ij}}^{(\mathrm{old})} - \eta \cdot e_j \cdot h_i.$$

$$\mathbf{v}_{w_I}^{(\mathrm{new})} = \mathbf{v}_{w_I}^{(\mathrm{old})} - \eta \cdot \mathrm{EH}$$

$${\mathbf{v}'_{w_j}}^{(\mathrm{new})} = {\mathbf{v}'_{w_j}}^{(\mathrm{old})} - \eta \cdot e_j \cdot \mathrm{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

CBOW with Larger Context



$$\begin{aligned}\mathbf{h} &= \frac{1}{C}\mathbf{W}\cdot(\mathbf{x}_1+\mathbf{x}_2+\cdots+\mathbf{x}_C)\\ &= \frac{1}{C}\cdot(\mathbf{v}_{w_1}+\mathbf{v}_{w_2}+\cdots+\mathbf{v}_{w_C})\end{aligned}$$

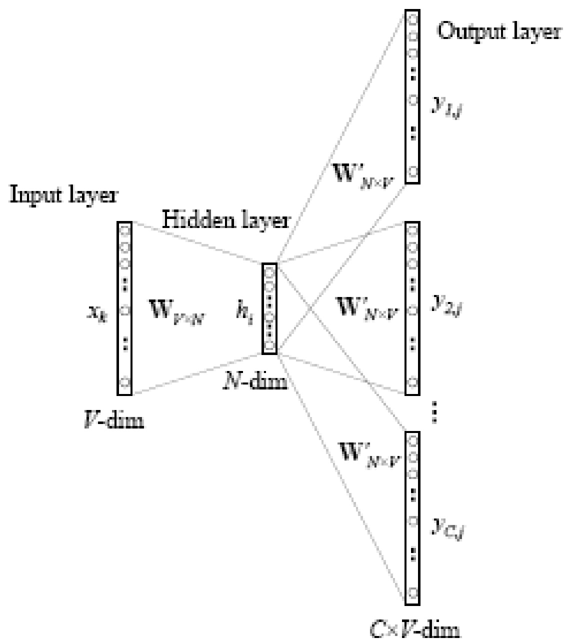
$$\begin{aligned}E &= = -\log p(w_O|w_{I,1},\cdots,w_{I,C})\\ &= -u_{j^*}+\log\sum_{j'=1}^{V}\exp(u_{j'})\\ &= -{\mathbf{v}'_{w_O}}^T\cdot\mathbf{h}+\log\sum_{j'=1}^{V}\exp({\mathbf{v}'_{w_j}}^T\cdot\mathbf{h})\end{aligned}$$

$${\mathbf{v}'_{w_j}}^{(\text{new})} = {\mathbf{v}'_{w_j}}^{(\text{old})} - \eta\cdot e_j\cdot\mathbf{h} \qquad \text{for } j=1,2,\cdots,V.$$

$$\mathbf{v}^{(\text{new})}_{w_{I,c}} = \mathbf{v}^{(\text{old})}_{w_{I,c}} - \frac{1}{C}\cdot\eta\cdot\mathrm{EH} \qquad \text{for } c=1,2,\cdots,C.$$

Skip-gram Model

- Context and input word have changed order



$$\mathbf{h} = \mathbf{W}_{(k,\cdot)} := \mathbf{v}_{w_I},$$

$$u_{c,j} = u_j = {\mathbf{v}'_{w_j}}^T \cdot \mathbf{h}, \text{ for } c = 1, 2, \cdots, C$$

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})}$$

Skip-gram Model

$$
\begin{aligned}
E &= -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} | w_I) \\
&= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} \exp(u_{j'})} \\
&= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'})
\end{aligned}
$$

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j}$$

$$\mathrm{EI}_j = \sum_{c=1}^{C} e_{c,j}$$

$$\mathrm{EH}_i = \sum_{j=1}^{V} \mathrm{EI}_j \cdot w'_{ij}.$$

$$\frac{\partial E}{\partial w'_{ij}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w'_{ij}} = \mathrm{EI}_j \cdot h_i$$

$$w'^{(\text{new})}_{ij} = w'^{(\text{old})}_{ij} - \eta \cdot \mathrm{EI}_j \cdot h_i$$

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot \mathrm{EI}_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

$$\mathbf{v}^{(\text{new})}_{w_I} = \mathbf{v}^{(\text{old})}_{w_I} - \eta \cdot \mathrm{EH}$$

More…

- Why does word2vec work?
- It seems that SGNS (skip-gram negative sampling) is actually performing a (weighted) implicit matrix factorization
- The matrix is using the PMI between words and contexts
- PMI and implicit matrix factorizations have been widely used in NLP

$$M_{ij}^{\text{SGNS}} = W_i \cdot C_j = \vec{w}_i \cdot \vec{c}_j = PMI(w_i, c_j) - \log k$$

- It is interesting that the PMI matrix emerges as the optimal solution for SGNS's objective

WS353 (WordSim) [13]			MEN (WordSim) [4]			Mixed Analogies [20]			Synt. Analogies [22]		
Representation		Corr.	Representation		Corr.	Representation		Acc.	Representation		Acc.
SVD	(k=5)	0.691	SVD	(k=1)	0.735	SPPMI	(k=1)	0.655	SGNS	(k=15)	0.627
SPPMI	(k=15)	0.687	SVD	(k=5)	0.734	SPPMI	(k=5)	0.644	SGNS	(k=5)	0.619
SPPMI	(k=5)	0.670	SPPMI	(k=5)	0.721	SGNS	(k=15)	0.619	SGNS	(k=1)	0.59
SGNS	(k=15)	0.666	SPPMI	(k=15)	0.719	SGNS	(k=5)	0.616	SPPMI	(k=5)	0.466
SVD	(k=15)	0.661	SGNS	(k=15)	0.716	SPPMI	(k=15)	0.571	SVD	(k=1)	0.448
SVD	(k=1)	0.652	SGNS	(k=5)	0.708	SVD	(k=1)	0.567	SPPMI	(k=1)	0.445
SGNS	(k=5)	0.644	SVD	(k=15)	0.694	SGNS	(k=1)	0.540	SPPMI	(k=15)	0.353
SGNS	(k=1)	0.633	SGNS	(k=1)	0.690	SVD	(k=5)	0.472	SVD	(k=5)	0.337
SPPMI	(k=1)	0.605	SPPMI	(k=1)	0.688	SVD	(k=15)	0.341	SVD	(k=15)	0.208

Final

- “PMI matrices are commonly used by the traditional approach to represent words (often dubbed "distributional semantics"). What's really striking about this discovery, is that word2vec (specifically, SGNS) is doing something very similar to what the NLP community has been doing for about 20 years; it's just doing it really well.”

Omer Levy - http://www.quora.com/How-does-word2vec-work

References

Word2vec & related papers:

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
- Mikolov, T., Yih, W. T., & Zweig, G. (2013, June). Linguistic Regularities in Continuous Space Word Representations. In *HLT-NAACL* (pp. 746-751).

Explanations

- Rong, X. (2014). word2vec Parameter Learning Explained. *arXiv preprint arXiv:1411.2738*.
- Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.
- Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems* (pp. 2177-2185).
- Dyer, C. (2014). Notes on Noise Contrastive Estimation and Negative Sampling. *arXiv preprint arXiv:1410.8251*.

Applications of word2vec

- Mikolov, T., Le, Q. V., & Sutskever, I. (2013). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.
- Levy, O., & Goldberg, Y. (2014). Dependency based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (Vol. 2, pp. 302-308).