Efficient Estimation of Word Representations in Vector Space

By Tomas Mikolov, Kai Chen , Greg Corrado, Jeffrey Dean. Google Inc.,

Hyopil Shin @SNU

Contents

n-gram model for NLP

- Traditional NLP models are based on prediction of next word given previous $n-1$words. Also known as n-gram model
- An n-gram model is defined as probability of a word w, given previous words $x_1, x_2 \ldots x_{n-1}$using $(n-1)^{th}$order Markov assumption

n-gram model for NLP

- Mathematically, the parameter

$$q(w|x_1, x_2 \ldots x_{n-1}) = \frac{count(w, x_1, x_2 \ldots x_{n-1})}{count(x_1, x_2 \ldots x_{n-1})}$$

- where w, $x_1, x_2 \ldots x_{n-1} \in V$ and V is some definite size vocabulary
- Above model is based on Maximum Likelihood estimation
- Probability of occurrence of any sentence can be obtained by multiplying the n-gram model of every word
- Estimation can be done using linear interpolation or discounting methods

Drawbacks associated with n-gram models

- Curse of dimensionality: large number of parameters to be learned even with the small size of vocabulary
- n-gram model has discrete space, so it's difficult to generalize the parameters for that model. On the other hand, generalization is easier when the model has continuous space
- Simple scaling up of n-gram models do not show expected performance improvement for vocabularies containing limited data
- n-gram models do not perform well in word similarity tasks

Distributed representation of words as vectors

- Associate with each word in the vocabulary a distributed word feature vector in $\mathbb{R}m$

- A vocabulary V of size V will therefore have $V \times m$ free parameters, which needs to learned using some learning algorithm.
- These distributed feature vectors can either be learned in an unsupervised fashion as part of pre-training procedure or can also be learned in a supervised way as well.

Why word vector model?

- One-hot encoding
  - For a vocabulary size D = 10, the one-hot vector of word ID w = 4 is

  e(w) = [0 0 0 1 0 0 0 0 0 0]
  - This method doesn't make any assumption about word similarity
  - This choice has several good reasons – simplicity, robustness and can be trained on huge amount of data. However, the performance is highly dependent on the size and quality of data.
  - In recent years it has become possible to train more complex models on much larger data set and it has been found that distributed representation of words using neural network based language models significantly outperform N-gram models.

Why word vector model?

- Continuous word representation
  - The idea is to replace the One-hot encoding which doesn't take into account similarity of word and replace it with learned vector representation.

Word	w	$C(w)$
" the "	1	[0.6762, -0.9607, 0.3626, -0.2410, 0.6636]
" a "	2	[0.6859, -0.9266, 0.3777, -0.2140, 0.6711]
" have "	3	[0.1656, -0.1530, 0.0310, -0.3321, -0.1342]
" be "	4	[0.1760, -0.1340, 0.0702, -0.2981, -0.1111]
" cat "	5	[0.5896, 0.9137, 0.0452, 0.7603, -0.6541]
" dog "	6	[0.5965, 0.9143, 0.0899, 0.7702, -0.6392]
" car "	7	[-0.0069, 0.7995, 0.6433, 0.2898, 0.6359]

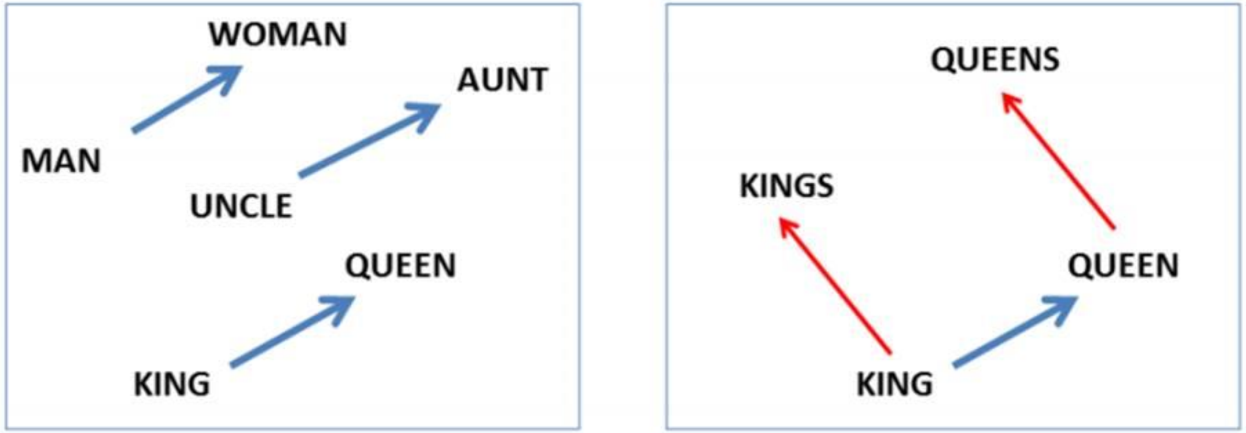
Why word vector model?

- Multiple degrees of similarity : similarity between words goes beyond basic syntactic and semantic regularities.

  For example:

  - $vector(King) - vector(Man) + vector(Woman) \approx vector(Queen)$
  - $vector(Paris) - vector(France) + vector(Italy) \approx vector Rome$

- Easier to train vector models on unsupervised data



(Mikolov et al., NAACL HLT, 2013)

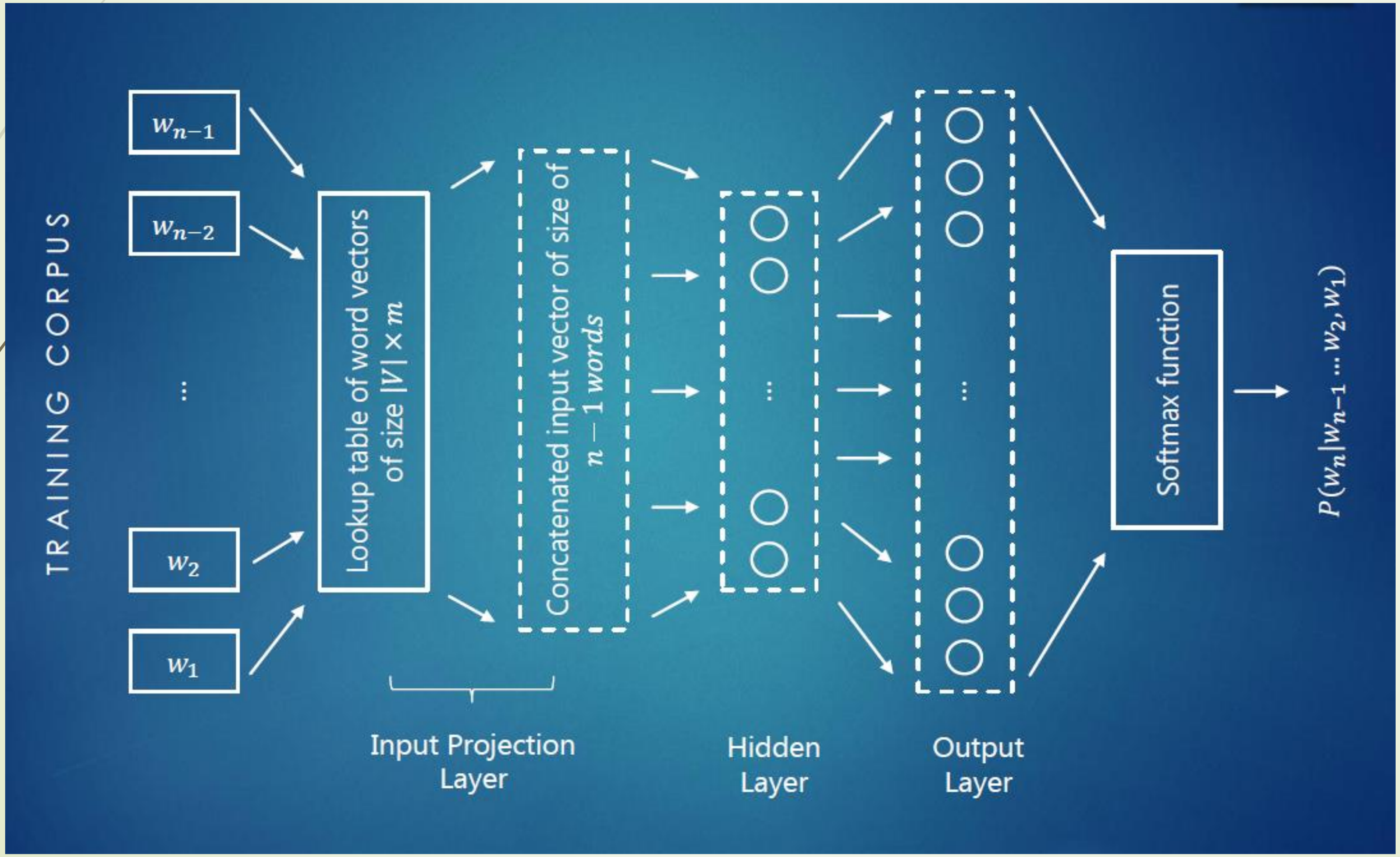
Learning distributed word vector representations

- Feedforward Neural Network Language Model : Joint probability distribution of words sequences is learned along with word feature vectors using feed forward neural network
- Recurrent Neural Network Language Models : These NNLM are based on recurrent neural networks
- Continuous Bag of Words : It is based on log linear classifier, but the input will be average of past and future word vectors. In short, here our goal is to predict word surrounding a context
- Continuous Skip-gram Model: It is also based on log linear classifier, but here it will try to predict the past and future words surrounding a given word

Feedforward Neural Network Language Model

- Initially proposed by Yoshua Bengio et al
- It is slightly related to n-gram language model, as it aims to learn the probability function of word sequences of length n
- Here input will be a concatenated feature vector of words $w_{n-1}, w_{n-2} \ldots w_2, w_1$ and training criteria will be to predict the word w_n
- Output of the model will give us the estimated probability of a given sequence of n words
- Neural network architecture consists of a projection layer, a hidden layer of neurons, output layer and a softmax function to evaluate the joint probability distribution of words

Feedforward NNLM



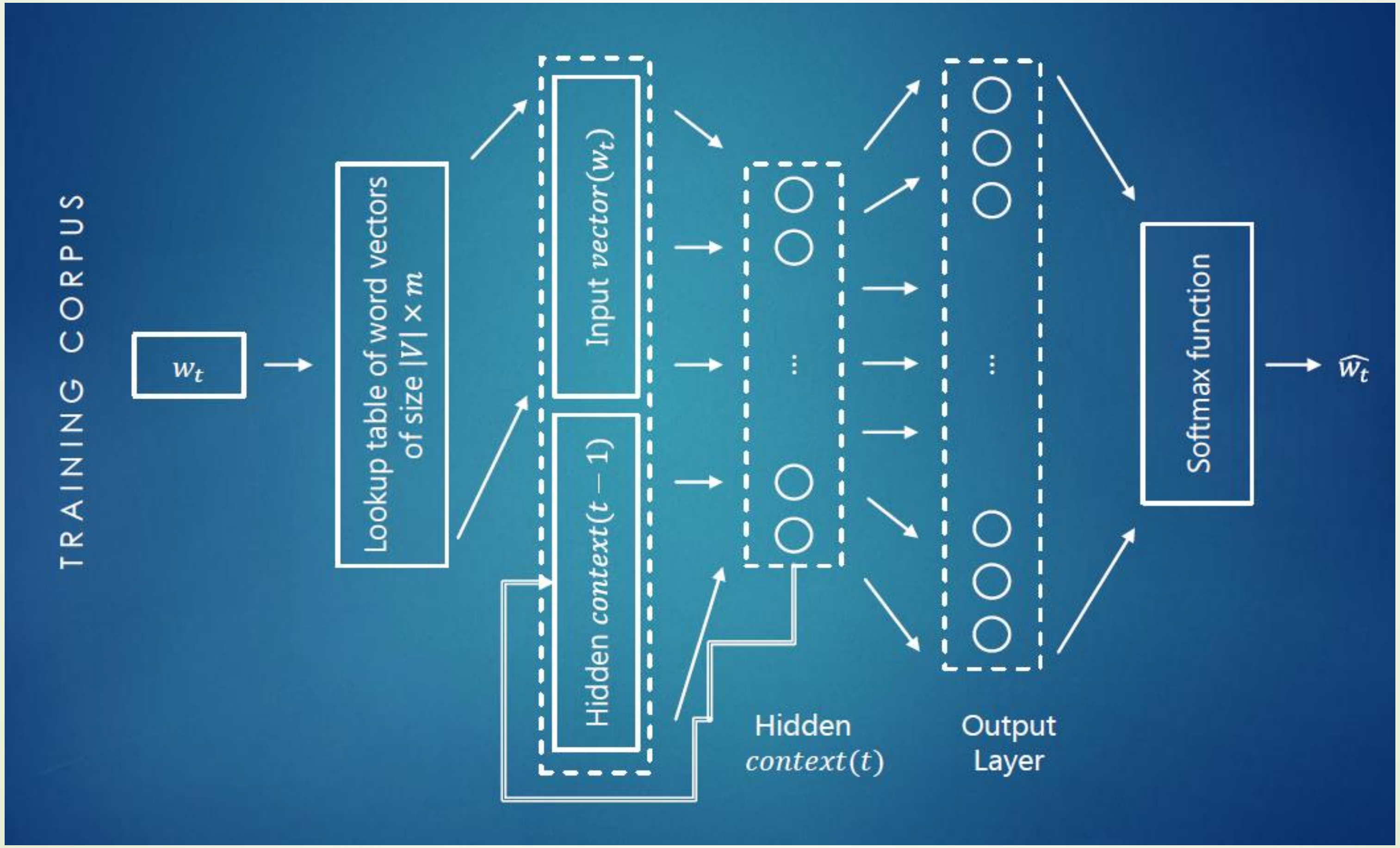
Feedforward NNLM

- Fairly huge model in terms of free parameters
- Neural network parameters consist of $(n-1)\times m\times H+H\times |V|$ parameters
- Training criteria is to predict n^{th} word
- Uses forward propagation and backpropagation algorithm for training using mini batch gradient descent
- Number of output layers in neural network can be reduced to $\log_2|V|$ using hierarchical softmax layers. This will significantly reduce the training time of model

Recurrent Neural Network Language Model

- Initially implemented by Tomas Mikolov, but probably inspired by Yoshua Bengio's seminal work on NNLM
- Uses a recurrent neural network, where input layer consists of the current word vector and hidden neuron values of previous word
- Training objective is to predict the current word
- Contrary to Feedforward NNLM, it keeps on building a kind of history of previous words which got trained using the model. Therefore context window of analysis is variable here

Recurrent Neural Network Language Model



Recurrent Neural Network Language Model

- Requires less number of hidden units in comparison to feedforward NNLM, though one may have to increase the same with increase in vocabulary size
- Stochastic gradient descent is used along with backpropagation algorithm to train the model over several epochs
- Number of output layers can be reduced to $\log_2 V$ using hierarchical softmax layers
- Recurrent NNLM models as much as twice reduction in perplexity as compared to n-gram models
- In practice recurrent NNLM models are much faster to train than feedforward NNLM models

New Log-linear Models

- The main observation from the previous section was that most of the complexity is caused by the non-linear hidden layer in the model.
- While this is what makes neural networks so attractive, the author explains a simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.
- The two proposed architectures are:

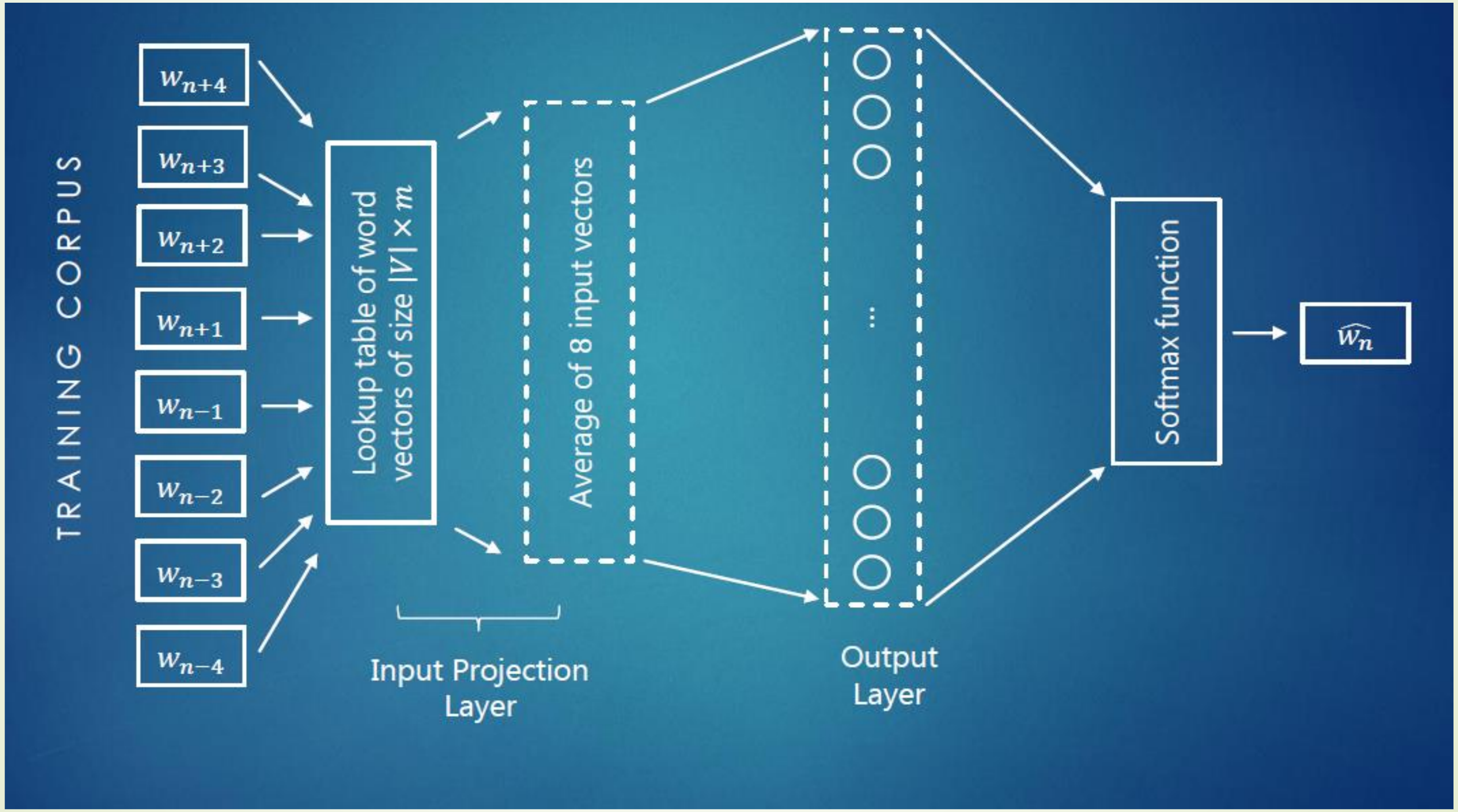
  a) Continuous Bag-of-words Model

  b) Continuous Skip-gram Model

Continuous Bag of Words

- It is similar to feedforward NNLM with no hidden layer. This model only consists of an input and an output layer
- In this model, words in sequences from past and future are input and they are trained to predict the current sample
- Owing to its simplicity, this model can be trained on huge amount of data in a small time as compared to other neural network models
- This model actually does the current word estimation provided context or a sentence.

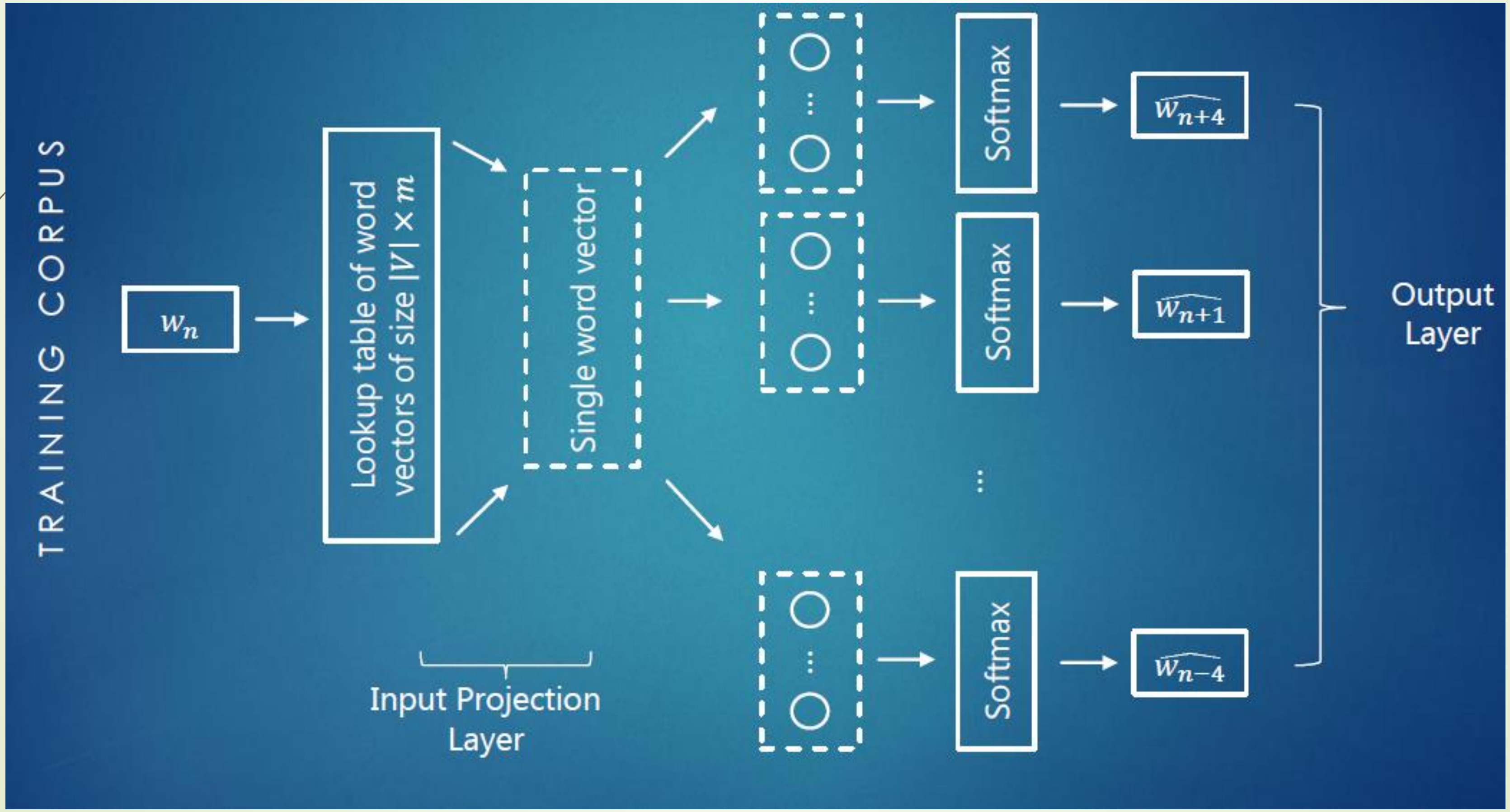
Continuous Bag of Words



Continuous Skip-gram Model

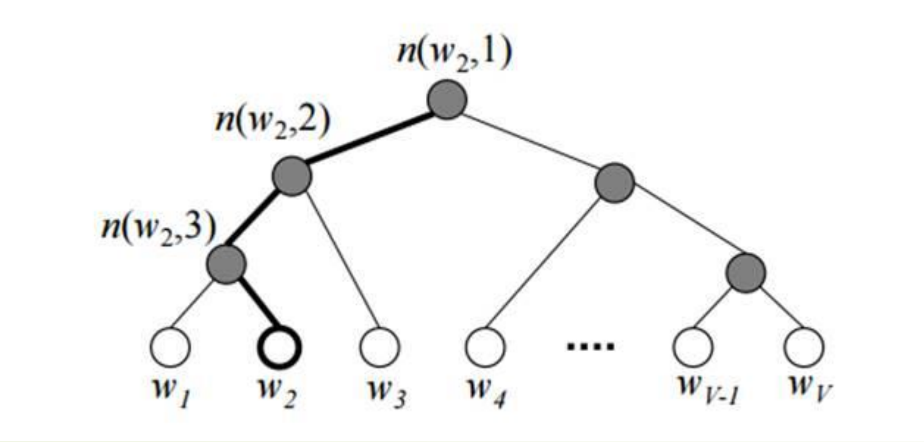
- This model is similar to continuous bag of words model, its just the roles are reversed for input and output
- Here model attempts to predict the words around the current word
- Input layer consists of the word vector from single word, while multiple output layers are connected to input layer

Continuous Skip-gram Model



Complexity Reduction: Hierarchical Softmax

- To reduce computation of V in Softmax
  - Hierarchical Softmax and Negative Sampling
- Uses Multinomial distribution function
- Binary tree for Hierarchical Softmax



$$p(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w, j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {\mathbf{v}'_{n(w,j)}}^{T} \mathbf{h} \right)$$

Remember? Softmax in MaxEnt Model

$$P(y = true \mid x) = \sum_{i=0}^{N} w_i \times f_i$$
$$= w \bullet f$$

$$\frac{P(y = true \mid x)}{1 - P(y = true \mid x)} = w \bullet f$$

$$\ln\left(\frac{P(y = true \mid x)}{1 - P(y = true \mid x)}\right) = w \bullet f$$

$$\frac{1}{1 + e^{-x}}$$

$$\ln\left(\frac{P(y = true \mid x)}{1 - P(y = true \mid x)}\right) = w \bullet f$$

$$= \frac{P(y = true \mid x)}{1 - P(y = true \mid x)} = e^{w \bullet f}$$ (로그를 없애고)

$$P(y = true \mid x) = (1 - P(y = true \mid x))e^{w \bullet f}$$ (대각선 곱에 의해)

$$P(y = true \mid x) = e^{w \bullet f} - P(y = true \mid x)e^{w \bullet f}$$

$$P(y = true \mid x) + P(y = true \mid x)e^{w \bullet f} = e^{w \bullet f}$$

$$P(y = true \mid x)(1 + e^{w \bullet f}) = e^{w \bullet f}$$

$$P(y = true \mid x) = \frac{e^{w \bullet f}}{1 + e^{w \bullet f}}$$

$$P(y = false \mid x) = \frac{1}{1 + e^{w \bullet f}}$$

Complexity Reduction: Remember? Softmax in MaxEnt Model

- Exponential (log-linear, maxent, logistic, Gibbs) models:
  - Make a probabilistic model from the linear combination $\Sigma\lambda_i f_i(c,d)$

$$P(c \mid d,\lambda)=\frac{\exp\sum_i \lambda_i f_i(c,d)}{\sum_{c'}\exp\sum_i \lambda_i f_i(c',d)}$$

    - P(LOCATION|*in Québec*) = $e^{1.8}e^{-0.6}/(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.586$
    - P(DRUG|*in Québec*) = $e^{0.3} /(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.238$
    - P(PERSON|*in Québec*) = $e^{0} /(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.176$
  - The weights are the parameters of the probability model, combined via a “soft max” function

Complexity Reduction: Negative Sampling

- Softmax computation using some samples instead of |V|
  - Computation reduction from NxV to NxK(No. of Samples)
  - Positive sample target words
  - Negative sample
  - –In Word2Vec : Error Function

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

  - Negative sampling based on Noise Distribution
    - Unigram power of ¾ → best result

Analyzing language models

- Perplexity : A measurement of how well a language model is able to adapt the underlying probability distribution of a model
- Word error rate : Percentage of words misrecognized by the language model
- Semantic Analysis : Deriving semantic analogies of word pairs, filling the sentence with most logical word choice etc. These kind of tests are especially used for measuring the performance of word vectors. For example : Berlin : Germany :: Toronto : Canada
- Syntactic Analysis : For language model, it might be the construction of syntactically correct parse tree, for testing word vectors one might look for predicting syntactic analogies such as : possibly : impossibly :: ethical : unethical

Perplexity Comparison

	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

Perplexity of different models tested on Brown Corpus

Model	Weight	PPL
3-gram with Good-Turing smoothing (GT3)	0	165.2
5-gram with Kneser-Ney smoothing (KN5)	0	141.2
5-gram with Kneser-Ney smoothing + cache	0.0792	125.7
Maximum entropy model	0	142.1
Random clusterings LM	0	170.1
Random forest LM	0.1057	131.9
Structured LM	0.0196	146.1
Within and across sentence boundary LM	0.0838	116.6
Log-bilinear LM	0	144.5
Feedforward NNLM	0	140.2
Syntactical NNLM	0.0828	131.3
Combination of static RNNLMs	0.3231	102.1
Combination of adaptive RNNLMs	0.3058	101.0
ALL	1	**83.5**

Perplexity comparison of different models on Penn Treebank

Sentence Completion Task

5-gram: IN TOKYO FOREIGN EXCHANGE TRADING YESTERDAY THE UNIT INCREASED AGAINST THE DOLLAR

RNNLM: IN TOKYO FOREIGN EXCHANGE TRADING YESTERDAY THE YEN INCREASED AGAINST THE DOLLAR

5-gram: SOME CURRENCY TRADERS SAID THE UPWARD REVALUATION OF THE GERMAN MARK WASN'T BIG ENOUGH AND THAT THE MARKET MAY CONTINUE TO RISE

RNNLM: SOME CURRENCY TRADERS SAID THE UPWARD REVALUATION OF THE GERMAN MARKET WASN'T BIG ENOUGH AND THAT THE MARKET MAY CONTINUE TO RISE

5-gram: MEANWHILE QUESTIONS REMAIN WITHIN THE E. M. S. WEATHERED YESTERDAY'S REALIGNMENT WAS ONLY A TEMPORARY SOLUTION

RNNLM: MEANWHILE QUESTIONS REMAIN WITHIN THE E. M. S. WHETHER YESTERDAY'S REALIGNMENT WAS ONLY A TEMPORARY SOLUTION

5-gram: MR. PARNES FOLEY ALSO FOR THE FIRST TIME THE WIND WITH SUEZ'S PLANS FOR GENERALE DE BELGIQUE'S WAR

RNNLM: MR. PARNES SO LATE ALSO FOR THE FIRST TIME ALIGNED WITH SUEZ'S PLANS FOR GENERALE DE BELGIQUE'S WAR

5-gram: HE SAID THE GROUP WAS MARKET IN ITS STRUCTURE AND NO ONE HAD LEADERSHIP

RNNLM: HE SAID THE GROUP WAS ARCANE IN ITS STRUCTURE AND NO ONE HAD LEADERSHIP

WSJ Kaldi Rescoring

Semantic Syntactic Tests

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

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

Results

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

Different models with 640 dimensional word vectors

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Training Time comparison of different models

References

- [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] Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155, 2003
- [3] T. Mikolov, J. Kopecky, L. Burget, O. Glembek and J. ´ Cernock ˇ y. Neural network based language models for highly inflective languages, In: Proc. ICASSP 2009