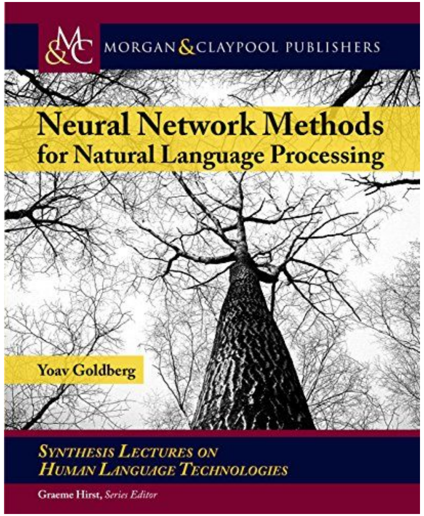
Dense Representations

Neural Network Methods for Natural Language Processing

by Yoav Goldberg

https://www.amazon.in/Language-Processing-Synthesis-Lectures-Technologies-ebook/dp/B071FGKZMH

Chapter 8



One-hot representations vs. Dense Representations

- We want a representation of input to pass to the statistical classifier

One-hot representations vs. Dense Representations

- We want a representation of input to pass to the statistical classifier

- One-hot: Each feature of the input is assigned a unique dimension
- Document: *Ind_(wk)*: 1 if word wk is present in the Document
  <Ind_(w0), Ind_(w1), Ind_(w2) ... , Ind_(wn)>

One-hot representations vs. Dense Representations

- We want a representation of input to pass to the statistical classifier

- One-hot: Each feature of the input is assigned a unique dimension
- Document: *Ind_(wk)*: 1 if word wk is present in the Document
  <Ind_(w0), Ind_(w1), Ind_(w2) ... , Ind_(wn)>

- Dense-Representation: Each feature of the input is assigned a vector
- Document:
- w0: <v1, v2, …, vn>
- w1: <v1, v2, …, vn>

One-hot representations vs. Dense Representations

- We want a representation of input to pass to the statistical classifier

- One-hot: Each feature of the input is assigned a unique dimension
- Document: *Ind_(wk)*: 1 if word wk is present in the Document
  <Ind_(w0), Ind_(w1), Ind_(w2) ... , Ind_(wn)>

- Dense-Representation: Each feature of the input is assigned a vector
- Document:
- w0: <v1, v2, …, vn>
- w1: <v1, v2, …, vn>

- Feature representations are learnt through SGD algorithm

Continuous Bag of Words Representation

- Dense-Representation: Each feature of the input is assigned a vector
- Features may be words, PoS tags, manually defined features

Continuous Bag of Words Representation

- Dense-Representation: Each feature of the input is assigned a vector
- Features may be words, PoS tags, manually defined features

- Classifiers require fixed-size inputs
- Aggregating various feature representations for a document

Continuous Bag of Words Representation

- Dense-Representation: Each feature of the input is assigned a vector
- Features may be words, PoS tags, manually defined features

- Classifiers require fixed-size inputs
- Aggregating various feature representations for a document

- Aggregation by summing all feature vectors
- Involves Bag of Words assumption
- As position of the word/feature is ignored

Continuous Bag of Words Representation

- Dense-Representation: Each feature of the input is assigned a vector
- Features may be words, PoS tags, manually defined features

- Classifiers require fixed-size inputs
- Aggregating various feature representations for a document

- Aggregation by summing all feature vectors
- Involves Bag of Words assumption
- As position of the word/feature is ignored

- CBOW - Continuous Bag of Words Representation

x+y

$$
\begin{tikzcd}
 & & & MLP(\square) & & & \\
 & & & sum(\square) \arrow[u] & & & \\
v(f_1) \arrow[urrr] & & v(f_2) \arrow[ur] & & v(f_3) \arrow[ul] & & v(f_4) \arrow[ulll] \\
f_1 \arrow[u] & & f_2 \arrow[u] & & f_3 \arrow[u] & & f_4 \arrow[u]
\end{tikzcd}
$$

Properties of learned representations

- Feature Representations are learnt through back-propagation

Properties of learned representations

- Feature Representations are learnt through back-propagation

- “Task-Specific” representations are learnt
- Features which have same meaning for the task will be grouped together

Properties of learned representations

- Feature Representations are learnt through back-propagation

- “Task-Specific” representations are learnt
- Features which have same meaning for the task will be grouped together

- “Excellent”, “Outstanding” will be *nearer* to each other compared to “Excellent”, “Disappointing” for sentiment analysis

Properties of learned representations

- Feature Representations are learnt through back-propagation

- "Task-Specific" representations are learnt
- Features which have same meaning for the task will be grouped together

- "Excellent", "Outstanding" will be *nearer* to each other compared to "Excellent", "Disappointing" for sentiment analysis

- *Nearness* measured by cosine similarity, euclidean distance between vectors

Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks

Yossi Adi[1,2], Einat Kermany[2], Yonatan Belinkov[3], Ofer Lavi[2], Yoav Goldberg[1]

[1]Bar-Ilan University, Ramat-Gan, Israel
{yoav.goldberg, yossiadidrum}@gmail.com
[2]IBM Haifa Research Lab, Haifa, Israel
{einatke, oferl}@il.ibm.com
[3]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
belinkov@mit.edu

Surprising Effectiveness of CBOW

- Understanding “document” embeddings of CBOW
- Contains more information than we might imagine!
- Can estimate the length of the sentence
- Contains a non-trivial amount of word-order
- Can be used to identify individual words in the sentence

Learning Task-Agnostic Word Embeddings

- Word representations learnt through document classification

- Representations/Embeddings
- Embedding of word in a higher-dimensional space

- Embeddings are task-specific
- How to learn task-agnostic word embeddings?

Learning Task-Agnostic Word Embeddings

- Word representations learnt through document classification

- Representations/Embeddings
- Embedding of word in a higher-dimensional space

- Embeddings are task-specific
- How to learn task-agnostic word embeddings?

- The *Distributional Hypothesis* is that words that occur in the same contexts tend to have similar meanings (Firth, 1957)
- A word is known by the company it keeps!

Representation Discovery

(Slides by Piotr Mirowski, Hugo Larochelle, Omer Levy, Yoav Goldberg, Graham Neubig, and Tomas Mikolov)

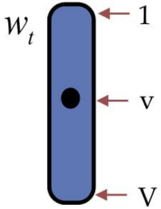
Distributed Representation

- Each word is associated with a continuous valued vector

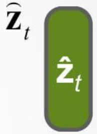
Word	w	$C(w)$
" the "	1	[0.6762, -0.9607, 0.3626, -0.2410, 0.6636]
" a "	2	[0.6859, -0.9266, 0.3777, -0.2140, 0.6711]
" have "	3	[0.1656, -0.1530, 0.0310, -0.3321, -0.1342]
" be "	4	[0.1760, -0.1340, 0.0702, -0.2981, -0.1111]
" cat "	5	[0.5896, 0.9137, 0.0452, 0.7603, -0.6541]
" dog "	6	[0.5965, 0.9143,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

Vector-space representation of words

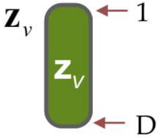
"**One-hot**" of "**one-of-V**" representation of a word token at position *t* in the text corpus, with **vocabulary of size *V***



Vector-space representation of the prediction of **target word w_t** (we predict a vector of size *D*)



Vector-space representation of any word *v* in the vocabulary using a vector of **dimension *D***



Also called **distributed representation**

Vector-space representation of the t^{th} **word history**: e.g., concatenation of *n*-1 vectors of size *D*

Predictive

- Input:
  - word history/context (**one-hot** or **distributed representation**)
- Output:
  - target word(s) (**one-hot** or **distributed representation**)
- **Function** that **approximates word likelihood**:
  - **Collobert & Weston**
  - **Continuous bag-of-words**
  - **Skip-gram**
  - …

Learning continuous space models

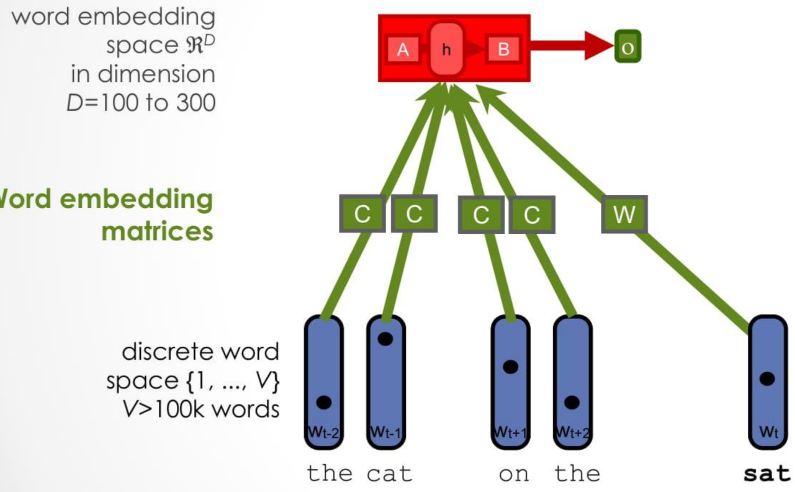
- How do we **learn the word representations z** for each word in the vocabulary?
- How do we **learn the model** that predicts a word or its representation $\hat{z}_t$ given a word context?

- Simultaneous learning of **model** and **representation**

Recap

Collobert & Weston

Prediction network: 2 layer network outputting a scalar



$$P\left(w_t \mid \mathbf{w}_{t-c}^{t-1}, \mathbf{w}_{t+1}^{t+c}\right) = \frac{e^{o(w)}}{\sum_v e^{o(v)}}$$

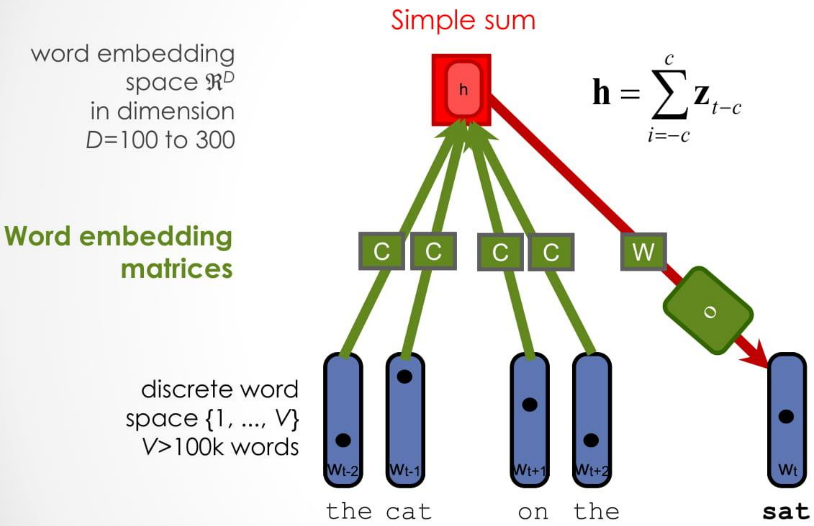
Solution: negative sampling
Max margin Loss:

max{0, 1-(o(w)-o(w'))}

Parameters: (2?)DxV + (2c+1)DxH + Hx1
Denominator: Iterate over V <not feasible>

[Mikolov et al, 2013a; Mnih & Kavukcuoglu, 2013; http://code.google.com/p/word2vec]

Continuous Bag-of-Words



$$\mathbf{h} = \sum_{i=-c}^{c} \mathbf{z}_{t-c}$$

$$\mathbf{o} = \mathbf{W}\mathbf{h}$$

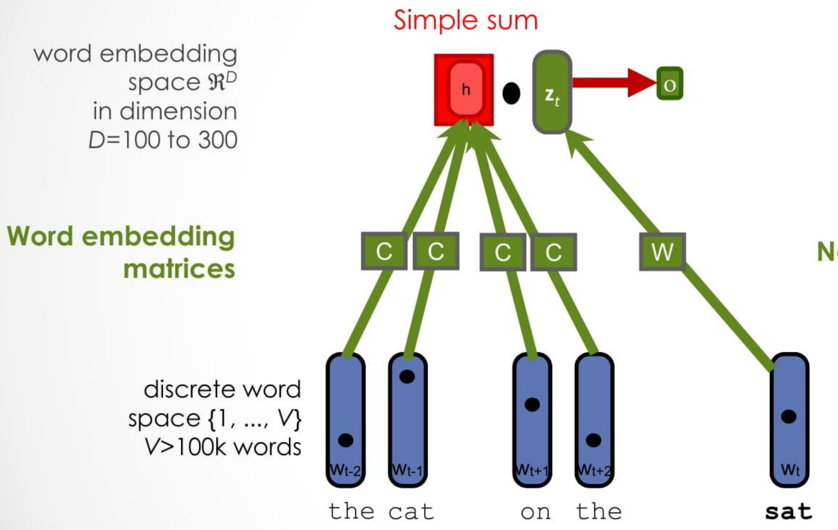
$$P\left(w_t \mid \mathbf{w}_{t-c}^{t-1}, \mathbf{w}_{t+1}^{t+c}\right) = \frac{e^{o(w)}}{\sum_v e^{o(v)}}$$

Parameters: 2DxV + 2c×D + D×V

Problem: large output space!

[Mikolov et al, 2013a; Mnih & Kavukcuoglu, 2013; http://code.google.com/p/word2vec]

Continuous Bag-of-Words



$$\mathbf{h} = \sum_{i=-c}^{c} \mathbf{z}_{t-c}$$

$$\mathbf{o} = \mathbf{h}.\mathbf{z}_t$$

Negative sampling for scalability (6B words)

$$\Pr(D=1 \mid c) = \sigma(h.w)$$
$$\Pr(D=0 \mid c) = \sigma(-h.w')$$

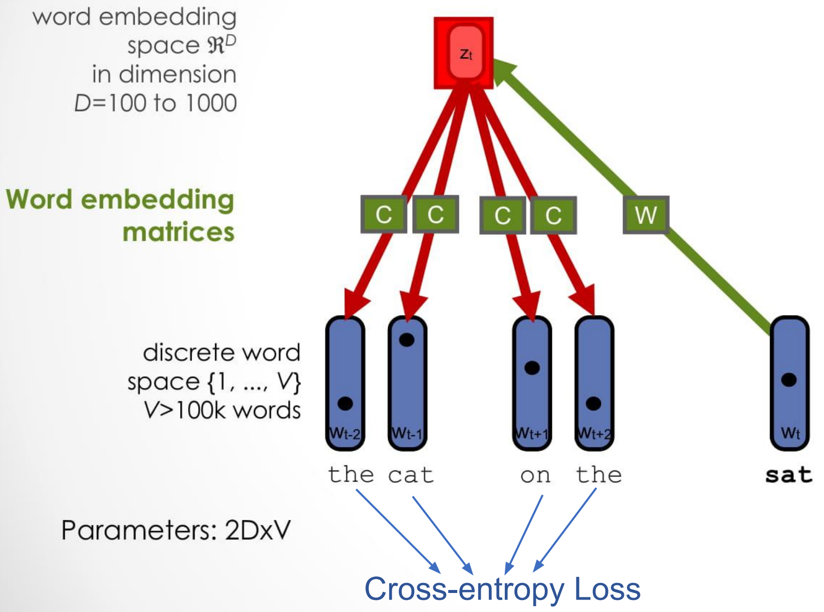
Parameters: 2DxV

good word+context pairs

bad word+context pairs

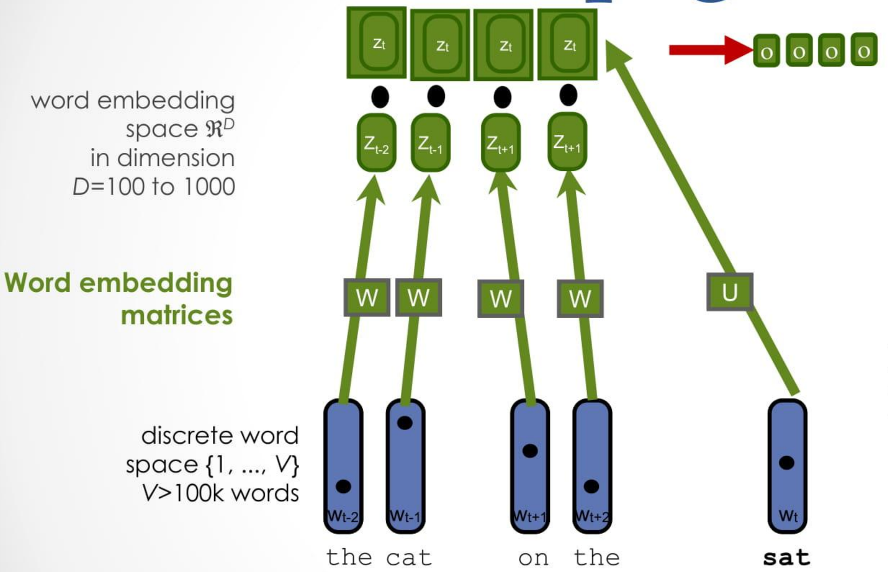
$$\mathcal{L}(\Theta; D, \bar{D}) = \sum_{(w,c)\in D} \log P(D=1|w,c) + \sum_{(w',c)\in \bar{D}} \log P(D=0|w',c)$$

Skip-gram



[Mikolov et al, 2013a, 2013b; Mnih & Kavukcuoglu, 2013;
http://code.google.com/p/word2vec]

Skip-gram



$o = z_{t+i} . z_t \quad i=-2,-1,1,2$

$\Pr(D=1 \mid c) = \sigma(c_i . w)$
$\Pr(D=0 \mid c) = \sigma(-c_i . w')$

Parameters: 2DxV
(Scales to 33B words)

[Mikolov et al, 2013a, 2013b; Mnih & Kavukcuoglu, 2013; http://code.google.com/p/word2vec]

Examples of Word2Vec embeddings

Example of word embeddings obtained using Word2Vec on the 3.2B word Wikipedia:

- Vocabulary V=2M
- Continuous vector space D=200
- Trained using CBOW

debt	aa	decrease	met	slow	france	jesus	xbox
debts	aaarm	increase	meeting	slower	marseille	christ	playstation
repayments	samavat	increases	meet	fast	french	resurrection	wii
repayment	obukhovskii	decreased	meets	slowing	nantes	savior	xbla
monetary	emerlec	greatly	had	slows	vichy	miscl	wiiware
payments	gunss	decreasing	welcomed	slowed	paris	crucified	gamecube
repay	dekhen	increased	insisted	faster	bordeaux	god	nintendo
mortgage	minizini	decreases	acquainted	sluggish	aubagne	apostles	kinect
repaid	bf	reduces	satisfied	quicker	vend	apostle	dsiware
refinancing	mortardept h	reduce	first	pace	vienne	bickertonite	eshop
bailouts	ee	increasing	persuaded	slowly	toulouse	pretribulational	dreamcast

[Mikolov et al, 2013a, 2013b; http://code.google.com/p/word2vec]

Semantic-syntactic word evaluation task

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

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

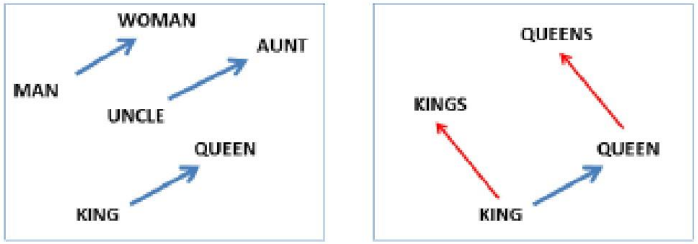
[Image credits: Mikolov et al (2013) "Efficient Estimation of Word Representation in Vector Space", *arXiv*]

Syntactic and Semantic tests

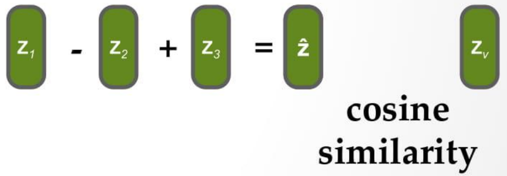
Observed that word embeddings obtained by RNN-LDA have linguistic regularities “a” is to “b” as “c” is to _

Syntactic: king is to kings as queen is to **queens**

Semantic: clothing is to shirt as dish is to **bowl**



Vector offset method



$$\arg\max_{b^* \in V} \left(\cos\left(b^*, b - a + a^*\right)\right)$$

$$\arg\max_{b^* \in V} \frac{\cos\left(b^*, b\right)\cos\left(b^*, a^*\right)}{\cos\left(b^*, a\right) + \varepsilon}$$

$$\arg\max_{b^* \in V} \left(\cos\left(b^*, b\right) - \cos\left(b^*, a\right) + \cos\left(b^*, a^*\right)\right)$$

3]

Linguistic Regularities - Examples

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

Speed-up over full softmax

LBL with **full softmax**, trained on APNews data, **14M words**, **V=17k** **7days**

Skip-gram (context 5) with phrases, trained using **negative sampling**, on Google data, **33G** words, **V=692k + phrases** **1 day**

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -

[Image credits: Mikolov et al (2013) "Distributed Representations of Words and Phrases and their Compositionality", *NIPS*]

LBL (2-gram, 100d) with **full softmax**, **1 day**

LBL (2-gram, 100d) with **noise contrastive estimation** **1.5 hours**

RNN (100d) with **50-class hierarchical softmax** **0.5 hours** (own experience)

Training algorithm	Number of samples	Test PPL	Training time (h)

Penn TreeBank data (900k words, V=10k)

[Image credits: Mnih & Teh (2012) "A fast and simple algorithm for training neura probabilistic language models", *ICML*]

[Mnih & Teh, 2012; Mikolov et al, 2010-2012, 2013b]

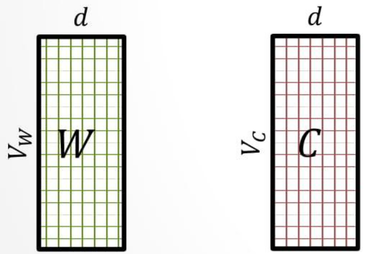
What is `word2vec`?

- `word2vec` is **not** a single algorithm
- It is a **software package** for representing words as vectors, containing:
  - Two distinct models
    - CBoW
    - **Skip-Gram** **(SG)**
  - Various training methods
    - **Negative Sampling** **(NS)**
    - Hierarchical Softmax
  - A rich preprocessing pipeline
    - Dynamic Context Windows
    - Subsampling
    - Deleting Rare Words

What is SGNS learning?

What is SGNS learning?

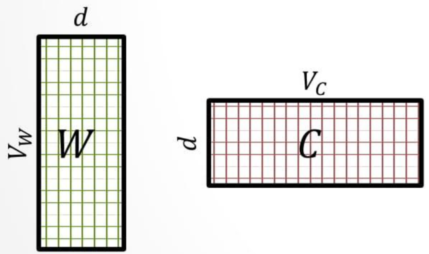
- Take SGNS's embedding matrices (W and C)



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- Take SGNS's embedding matrices (W and C)
- Multiply them
- What do you get?

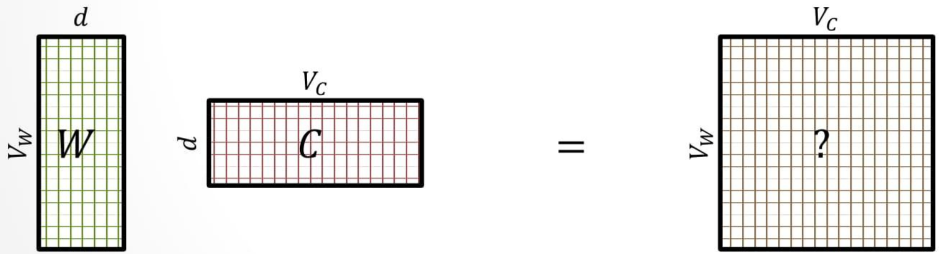


"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- A $V_W \times V_C$ matrix
- Each cell describes the relation between a specific word-context pair

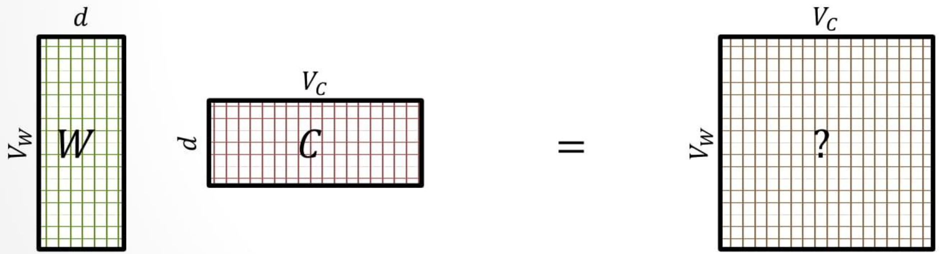
$$\vec{w} \cdot \vec{c} = ?$$



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

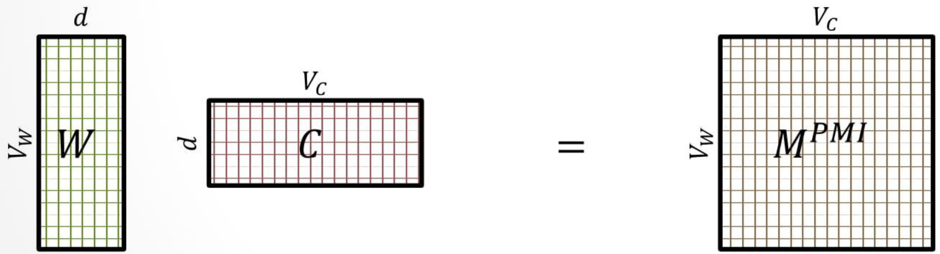
- We **prove** that for large enough d and enough iterations



"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- We **prove** that for large enough d and enough iterations
- We get the word-context PMI matrix

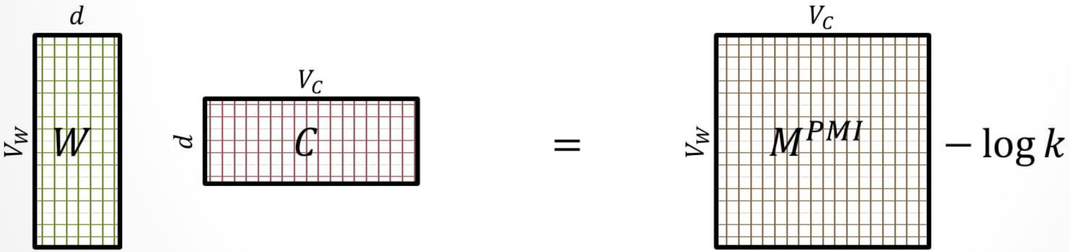


"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- We **prove** that for large enough d and enough iterations
- We get the word-context PMI matrix, shifted by a global constant

$$Opt(\vec{w} \cdot \vec{c}) = PMI(w, c) - \log k$$

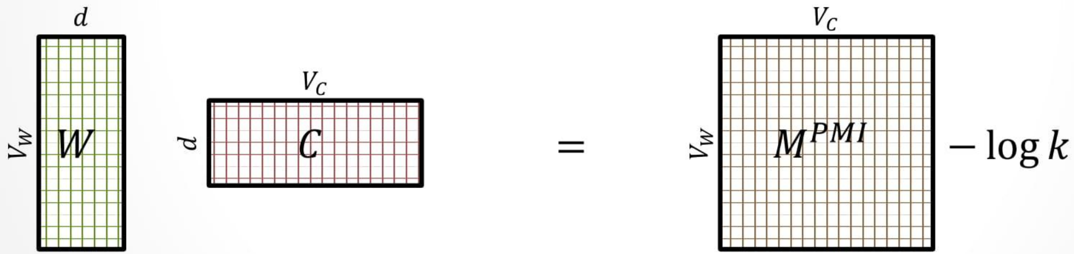


"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

What is SGNS learning?

- We **prove** that for large enough d and enough iterations
- We get the word-context PMI matrix, shifted by a global constant

$$Opt(\vec{w} \cdot \vec{c}) = PMI(w, c) - \log k$$



$$PMI(w, c) = \log \frac{\#(w, c) \cdot |D|}{\#(w) \cdot \#(c)}$$

"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014

GLOVE

- SGNS

$$\vec{w} \cdot \vec{c} = \mathrm{PMI}(w, c) - \log k$$

$$\ell = \sum_{w \in V_W} \sum_{c \in V_C} \#(w, c) \left(\log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \mathbb{E}_{c_N \sim P_D} \left[\log \sigma(-\vec{w} \cdot \vec{c}_N) \right] \right)$$

- GLOVE

$$\vec{w} \cdot \vec{c} + b_w + b_c = \log\left(\#(w, c)\right) \quad \forall (w, c) \in D$$

-

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Deep Learning Trivia

Interesting Read:
https://www.newyorker.com/magazine/2018/12/10/the-friendship-that-made-google-huge

Issues with Word2Vec and Glove

- Learning one embedding for each word in training data

- What to do with words missing in training data?

Issues with Word2Vec and Glove

- Learning one embedding for each word in training data

- What to do with words missing in training data?

- Option 1: Learn UNK embedding
- Replace words occurring only once or twice in the training data with UNK

Issues with Word2Vec and Glove

- Option 1: Learn UNK embedding
- Replace words occurring only once or twice in the training data with UNK

Issues with Word2Vec and Glove

- Option 1: Learn UNK embedding
- Replace words occurring only once or twice in the training data with UNK

- Issues:
- Loss of information
- Not using rich internal structure present in words - Morphology

- We can have a rough idea of Embedding(*‘taller’*) from Embedding(*‘tall’*)

Fasttext Representations

-

Enriching Word Vectors with Subword Information

Piotr Bojanowski* and **Edouard Grave*** and **Armand Joulin** and **Tomas Mikolov**
Facebook AI Research
{bojanowski,egrave,ajoulin,tmikolov}@fb.com

Fasttext Representations

- Train embedding for character n-grams

Fasttext Representations

- Train embedding for character n-grams

- Embedding of word = Sum of embedding of character n-grams

Fasttext Representations

- Train embedding for character n-grams
- Embedding of word = Sum of embedding of character n-grams
- Train skip-gram model based on these embeddings

Fasttext Representations

- Train embedding for character n-grams
- Embedding of word = Sum of embedding of character n-grams
- Train skip-gram model based on these embeddings
- Output: Learnt character n-gram embeddings

Fasttext Representations

- Train embedding for character n-grams
- Embedding of word = Sum of embedding of character n-grams
- Train skip-gram model based on these embeddings
- Output: Learnt character n-gram embeddings
- Unknown words - divide into constituent character n-grams
- Sum their embeddings

Issues with Word2Vec, Glove and Fasttext

- Context-insensitive embeddings
- Same embedding for Amazon in the two sentences
- Jeff Bezos, CEO of *Amazon* makes $2,219 per second — more than twice what the median US worker makes in one week.
- *Amazon* Rainforest wildfires this year at their highest level since 2010

Context-Sensitive Word Representations

- Assign topics for each word - using Word Sense Disambiguation, or LDA
- Represent each topic with a vector
- Jointly learn topic and word embeddings

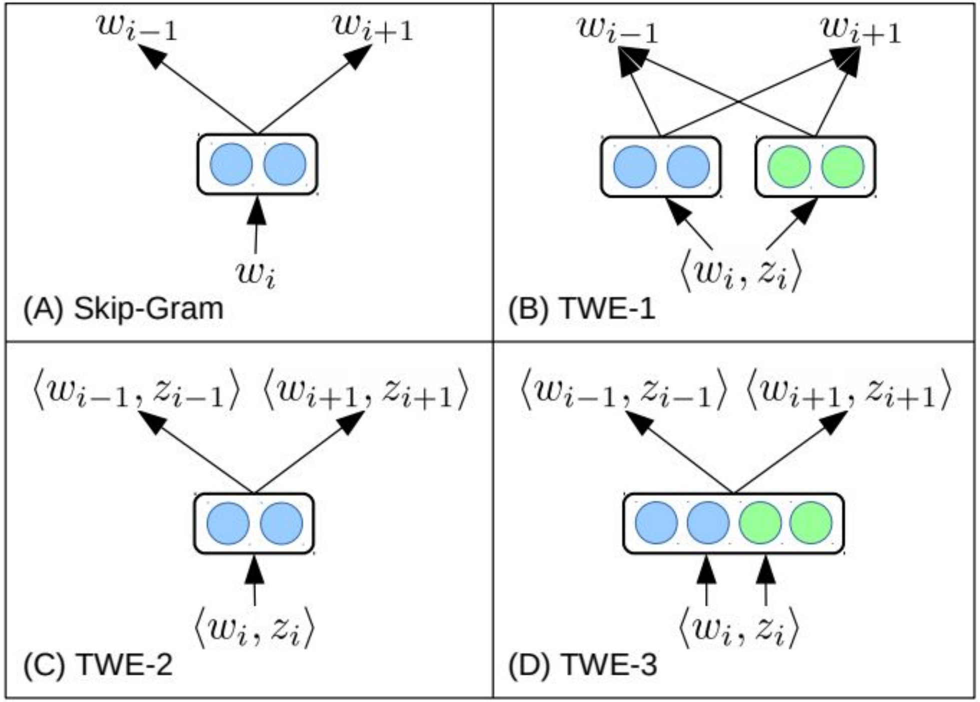


Figure 1: Skip-Gram and TWE models. Blue circles indicate word embeddings and green circles indicate topic embeddings. Since TWE-2 does not reserve stand-alone word / topic embeddings, we simply represent topical word embeddings in TWE-2 using blue circles.

Topical Word Embeddings, AAAI 2015

Document Embeddings

- Word2Vec presents an “unsupervised” algorithm for training word embeddings

- Can we come up with a similar technique for generating document embeddings?

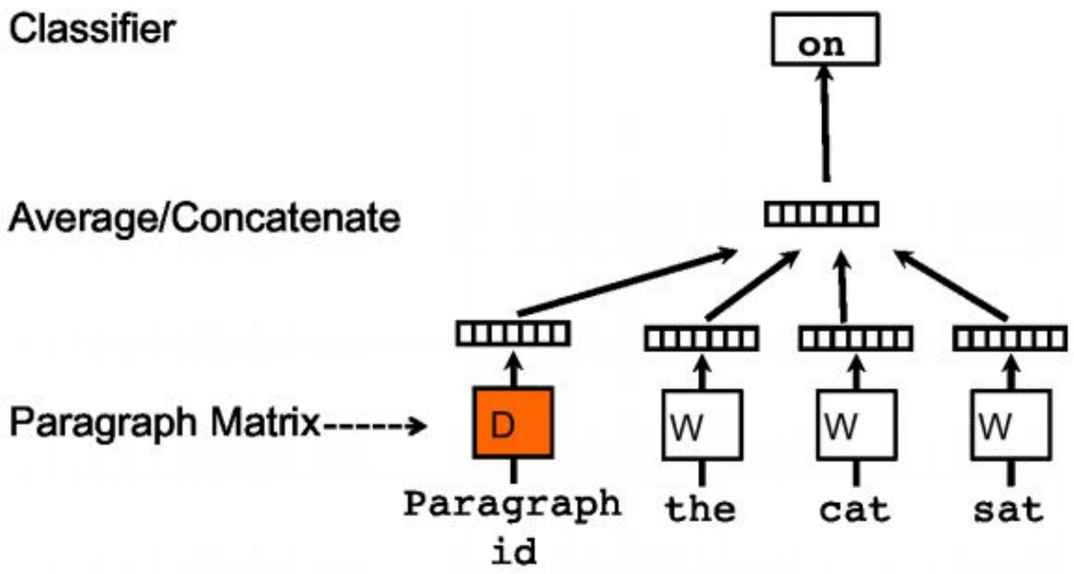


Figure 2. A framework for learning paragraph vector. This framework is similar to the framework presented in Figure 1; the only change is the additional paragraph token that is mapped to a vector via matrix D. In this model, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph.

Distributed Memory Model of Paragraph Vectors (PV-DM)

Distributed Representations of Sentences and Documents, ICML 2014

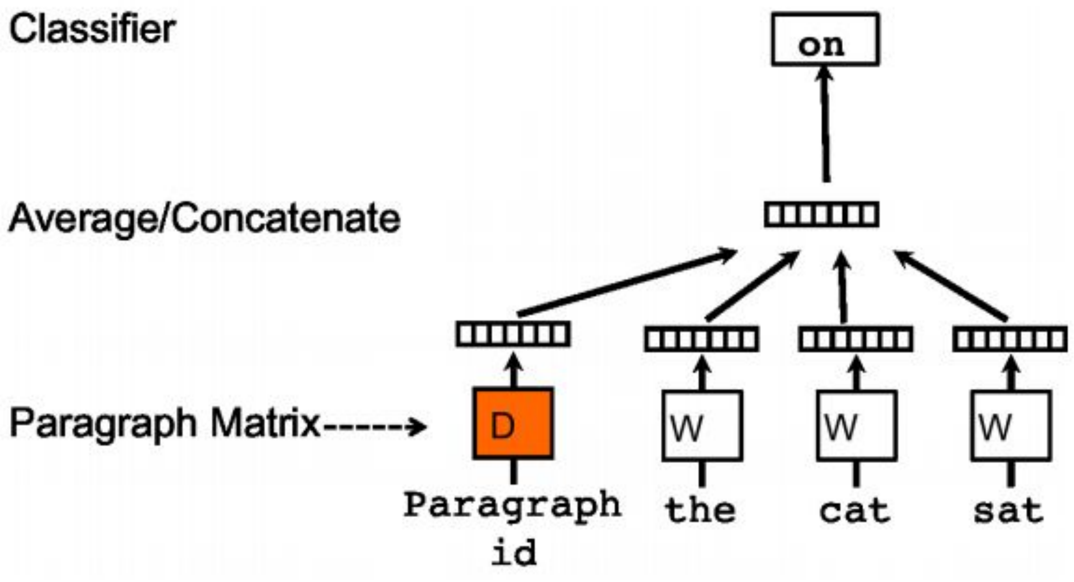


Figure 2. A framework for learning paragraph vector. This framework is similar to the framework presented in Figure 1; the only change is the additional paragraph token that is mapped to a vector via matrix D. In this model, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph.

Issue: Needs gradient descent at test time!

Variants of Document Embeddings

Weighted Neural Bag-of-n-grams Model:
New Baselines for Text Classification, COLING 2016

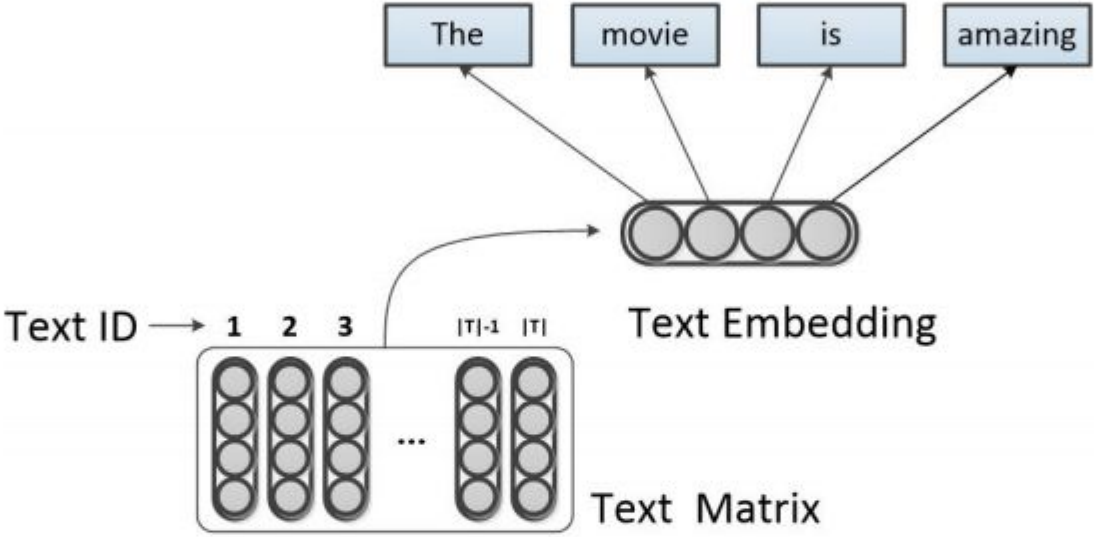


Figure 1: Illustration of the original Paragraph Vector model. The model only considers the uni-grams and they are equally treated in the model.

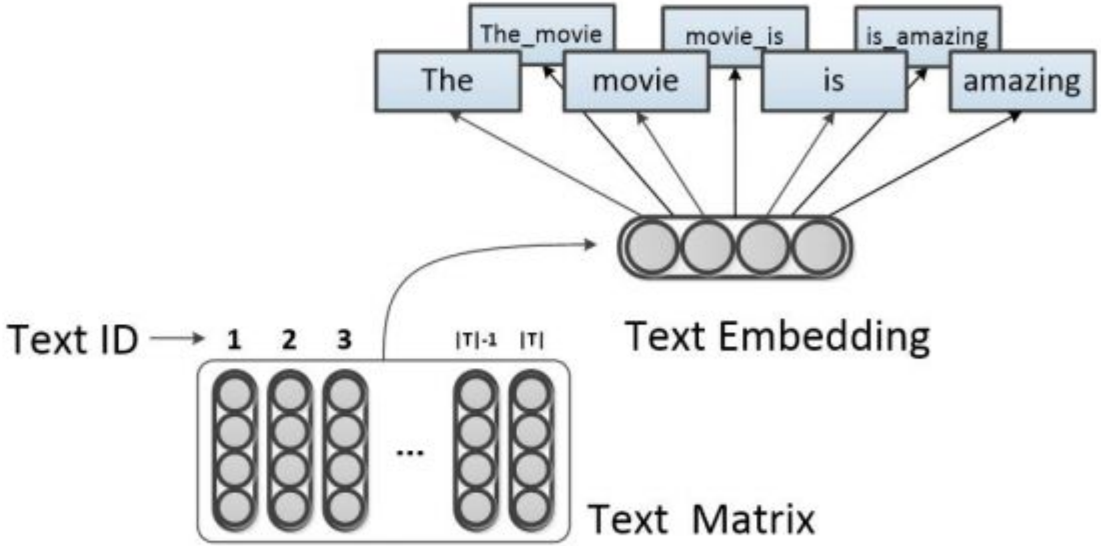


Figure 2: Illustration of n-gram PV model, where n-grams are predicted by the text embedding.

Variants of Document Embeddings

Weighted Neural Bag-of-n-grams Model:
New Baselines for Text Classification, COLING 2016

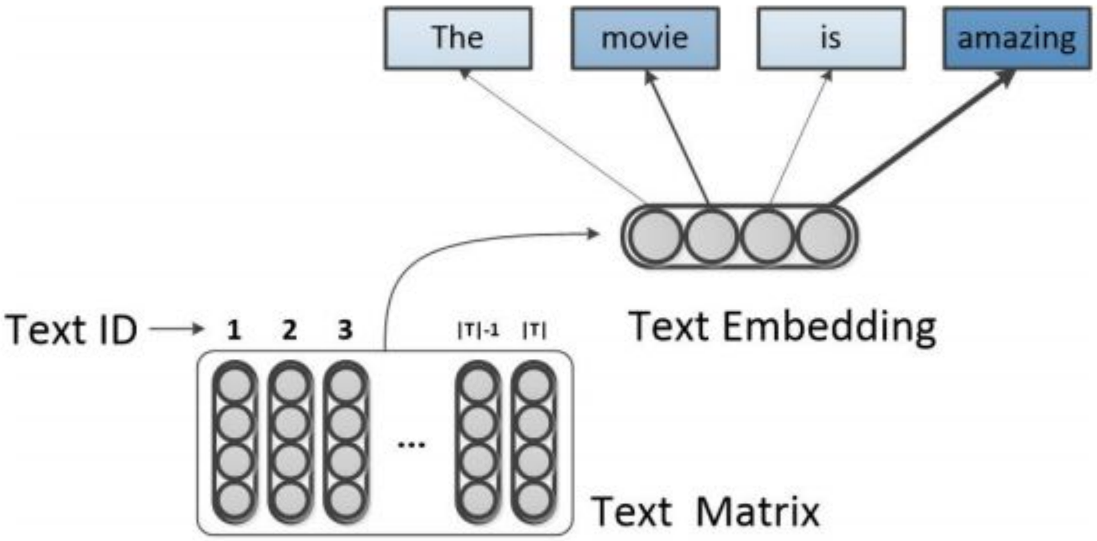


Figure 3: Illustration of weighted PV model, where important words are given more attention during the training process.

Variants of Document Embeddings

Weighted Neural Bag-of-n-grams Model:
New Baselines for Text Classification, COLING 2016

Variants of Document Embeddings

Weighted Neural Bag-of-n-grams Model:
New Baselines for Text Classification, COLING 2016

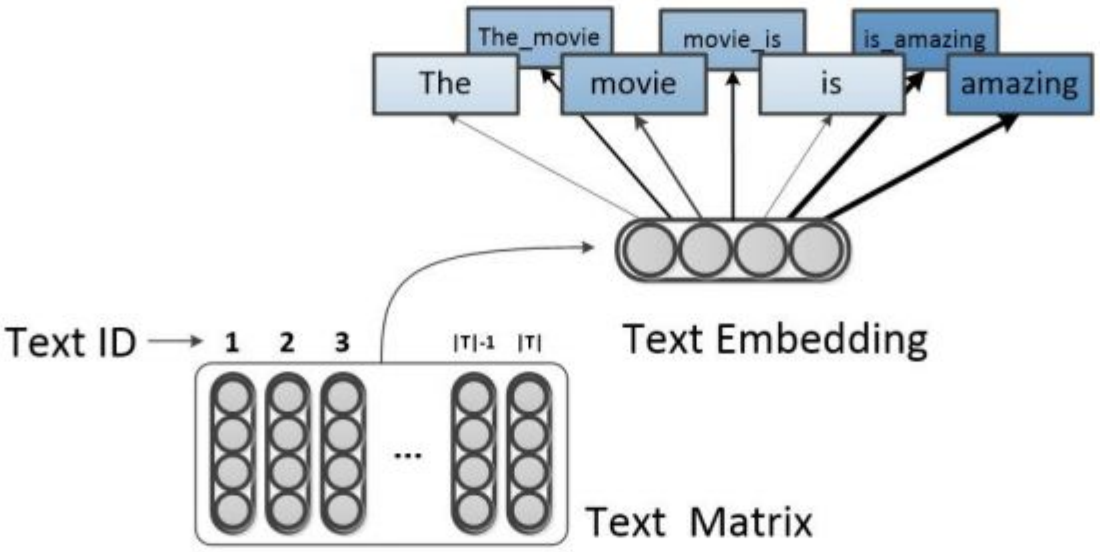


Figure 4: Combination of n-gram and weighting techniques. Text embeddings are trained to be useful to predict important n-grams during the training process.

Is this relevant?

PapersWithCode Leaderboard on IMDB Review Classification

RANK	MODEL	ACCURACY	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	NB-weighted-BON + dv-cosine	97.4	✓	Sentiment Classification Using Document Embeddings Trained with Cosine Similarity			2019
2	GraphStar	96.0	✓	Graph Star Net for Generalized Multi-Task Learning			2019
3	BERT large finetune UDA	95.8	✓	Unsupervised Data Augmentation for Consistency Training			2019
4	L MIXED	95.68	✓	Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function			2020
5	BERT large	95.49	✓	Unsupervised Data Augmentation for Consistency Training			2019

NB-Weighted-BON + dv-cosine

- Naive-Bayes weighted Bag of N-grams
- Train with cosine similarity instead of dot product

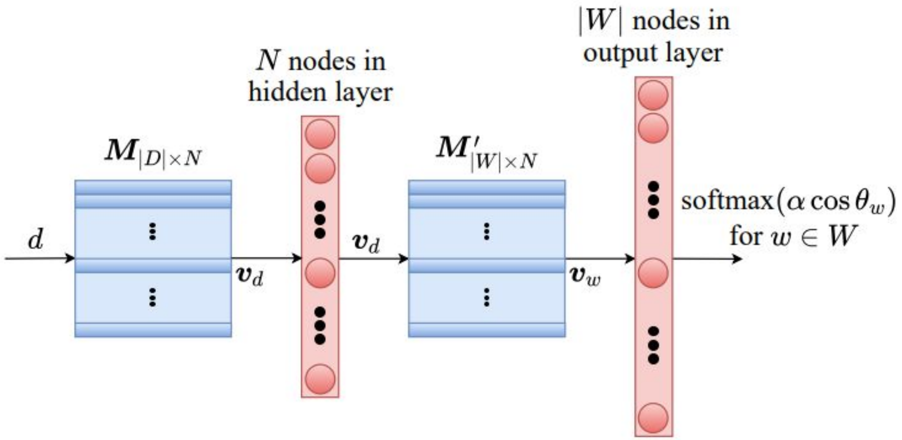


Figure 1: Proposed Architecture.

Limitations of Distributional Similarity

- What kind of similarity is hard to ~control?
  - Small context: more syntax-based embedding
  - Large context: more topical embeddings
  - Context based on parses: more functional embeddings
- Sensitive to superficial differences
  - Dog/dogs
- Black sheep
  - People don't say the obvious
- Antonyms
- Corpus bias
  - "encode every kind of psychological bias we can look for"
  - Females<->family and not career;
- Lack of context
  - See Elmo [2018]
- Not interpretable

Retrofitting Embeddings

- Additional evidence – e.g., Wordnet
- Graph: nodes – words, edges – related
- New objective: find matrix $\hat{W}$ such that
  - $\hat{w}$ is close to W for each word
  - $\hat{w}$ of words related in the graph is close

$$\Psi(Q) = \sum_{i=1}^{n} \left[\alpha_i \| w_i - \hat{w}_i \|^2 + \sum_{(i,j) \in E} \beta_{ij} \| \hat{w}_i - \hat{w}_j \|^2 \right]$$

Sparse Embeddings

- Each dimension of word embedding is not interpretable

- Add a sparsity constraint to
  - Increase the information content of non-zero dimensions in each word

De-biasing Embeddings

(Bolukbasi etal 16)

Extreme *she*	Extreme *he*
1. homemaker	1. maestro
2. nurse	2. skipper
3. receptionist	3. protege
4. librarian	4. philosopher
5. socialite	5. captain
6. hairdresser	6. architect
7. nanny	7. financier
8. bookkeeper	8. warrior
9. stylist	9. broadcaster
10. housekeeper	10. magician

Gender stereotype *she-he* analogies

sewing-carpentry	registered nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	lovely-brilliant

Gender appropriate *she-he* analogies

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Identify pairs to “neutralize”, find the direction of the trait to neutralize, and ensure that they are neutral in that direction

More Reading resources

- https://web.stanford.edu/~jurafsky/li15/lec3.vector.pdf
- https://ruder.io/word-embeddings-1/
- https://ruder.io/word-embeddings-softmax/index.html
- https://ruder.io/secret-word2vec/index.html

Finally, for the brave-hearted…

- Word2Vec - highly optimized C code:
- https://github.com/tmikolov/word2vec
- Note of Caution: Lots of malloc, calloc

- Readable version of the code:
- https://github.com/chrisjmccormick/word2vec_commented

- Python implementation:
- https://github.com/RaRe-Technologies/gensim

Pytorch Worksheet

- Link: https://colab.research.google.com/drive/1_2Ge4OLWj6I8O9Odp-OGYzHmr04tKC96?usp=sharing
- Contains 7 problems with varying levels of difficulty
- Will help improve your understanding of Pytorch
- Please attempt them before the next class
- We will share the solutions in the next class

Next Class

- CNN-based n-gram embeddings
- RNN: Recurrent Neural Networks
- LSTM: Long Short Term Memory
- GRU: Gated Recurrent Units