A word is worth a thousand vectors

Pramod Srinivasan

University of Illinois, Urbana-Champaign

May 31, 2016

Outline

What is a word?

"cat"

What is a word?

"cat"

2

"feline"

What is a word?

"cat" "feline"

2 299,999

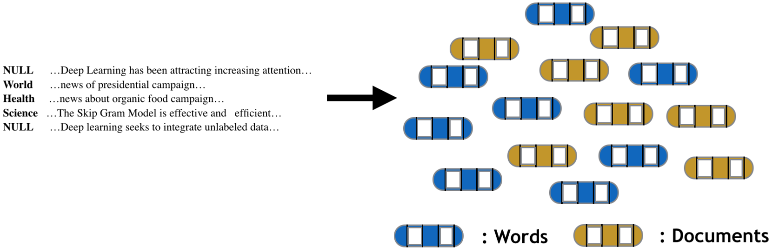
$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

What is a word?

"cat"	"feline"
2	299,999
$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$

Solving the Problem of Data Sparseness($|V|$)

- representing words and documents in a reduced parameter space
- improving generalization
- somehow transfer or share knowledge between words



The Distributional Hypothesis

> "You shall know a word by the company it keeps"
>
> — (J. R. Firth, 1957)

Words with **high similarity** occur in the **same contexts** as one another.

- A **word** ought to be able to **predict its context**
- A **context** ought to be able to **predict its missing word**

Distributed Representations

- Learns from raw text
- Not treating words as blocks rather as relationships
- Pretty simple algorithm
- Comes pre-trained

$$king - man + women = queen$$

Not a Deep Learning Algorithm : rather a Shallow Learning Algorithm

word2vec : Learning Algorithm

"The quick brown fox jumped over the lazy dog"

Objective : learn word vector over from its surrounding context

word2vec : Learning Algorithm

"The quick brown fox jumped over the lazy dog"

Objective : learn word vector over from its surrounding context
Maximize the likelihood of seeing this context given the word over

$$P(the|over)$$

$$P(quick|over)$$

$$P(brown|over)$$

$$P(fox|over)$$

$$P(jumped|over)$$

$$P(the|over)$$

$$P(lazy|over)$$

$$P(dog|over)$$

word2vec : Learning Algorithm(Contd.)

We assign two vectors for every word. Also a context window around every word.
$P(v_{OUT}|v_{IN})$

The quick brown fox jumped over the lazy dog.

$$\underset{v_{OUT}}{\uparrow}\text{ jumped} \quad \underset{v_{IN}}{\uparrow}\text{ over}$$

The quick brown fox jumped over the lazy dog.

$$\underset{v_{OUT}}{\uparrow}\text{ The} \quad \underset{v_{IN}}{\uparrow}\text{ over}$$

The quick brown fox jumped over the lazy dog.

$$\underset{v_{IN}}{\uparrow}\text{ lazy} \quad \underset{v_{OUT}}{\uparrow}\text{ dog}$$

word2vec : Learning Algorithm(Contd.)

We assign two vectors for every word. Also a context window around every word.

$$P(v_{OUT}|v_{IN})$$

We assign two vectors for every word. Also a context window around every word.

$$P(v_{OUT}|v_{IN})$$

Measure loss v_{OUT} and v_{IN}

$$v_{OUT} \cdot v_{IN}$$

We assign two vectors for every word. Also a context window around every word.

$$P(v_{OUT}|v_{IN})$$

Measure loss v_{OUT} and v_{IN}

$$v_{OUT} \cdot v_{IN}$$

Well, we had wanted to quantify probability but,

$$v_{OUT} \cdot v_{IN} \in [-1, 1]$$

word2vec : Learning Algorithm(Contd.)

We assign two vectors for every word. Also a context window around every word.

$$P(v_{OUT}|v_{IN})$$

Measure loss v_{OUT} and v_{IN}

$$v_{OUT} \cdot v_{IN}$$

Well, we had wanted to quantify probability but,

$$v_{OUT} \cdot v_{IN} \in [-1, 1]$$

So, we resort to a softmax

$$softmax(v_{OUT} \cdot v_{IN}) \in [0, 1]$$

word2vec : Performance

$$softmax = \frac{exp(v_{in} \cdot v_{out})}{\sum_{k \in V} exp(v_{in}.v_k)}$$

- V operations for every update
- VC operations per input word
- VCN over the whole corpus

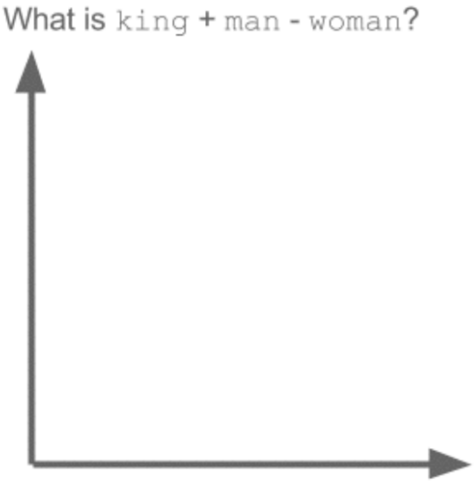
Naive word2vec : O(VCN) v/s O(NV^2) of that of SVD
Can we do better?

Addressing the Performance problem

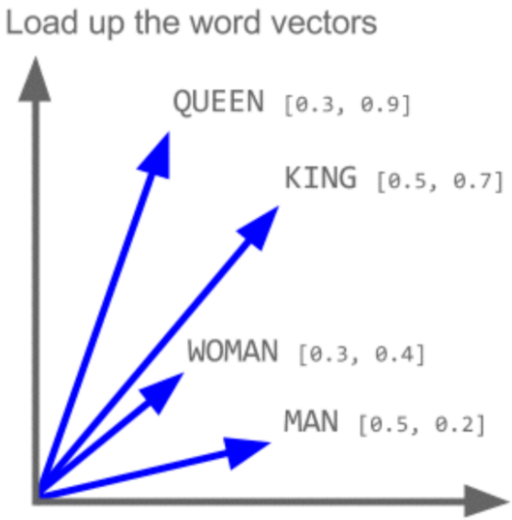
Have an O(V) Problem?
Build a tree and get a O(logV) problem!!

Hierarchial Softmax

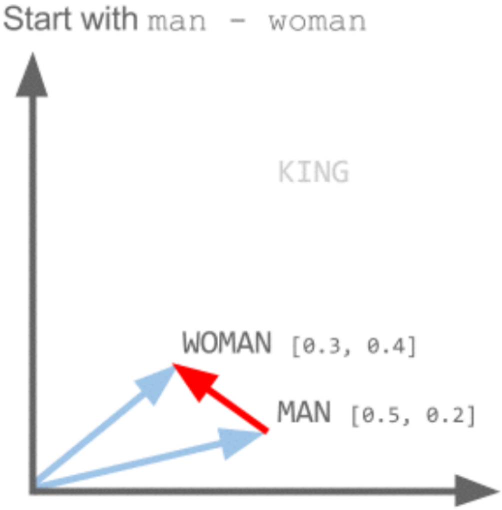
word2vec : nuts and bolts



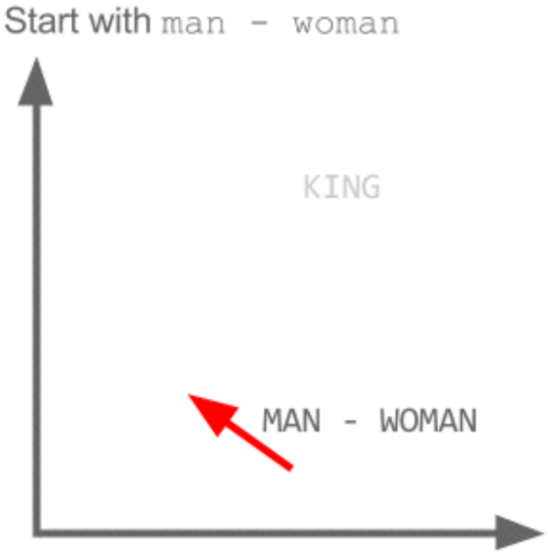
word2vec : nuts and bolts

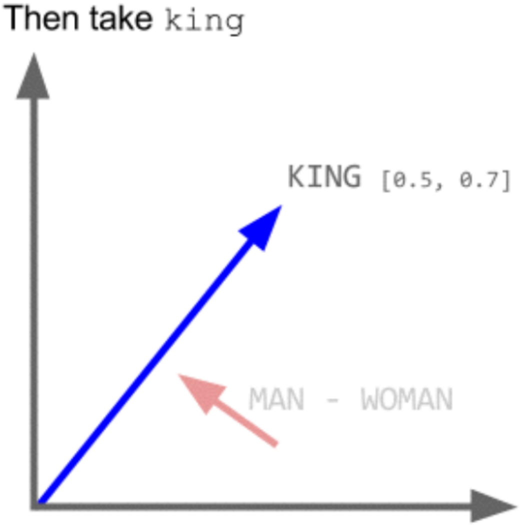


word2vec : nuts and bolts

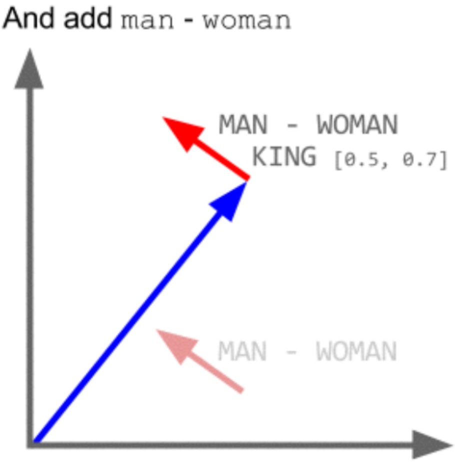


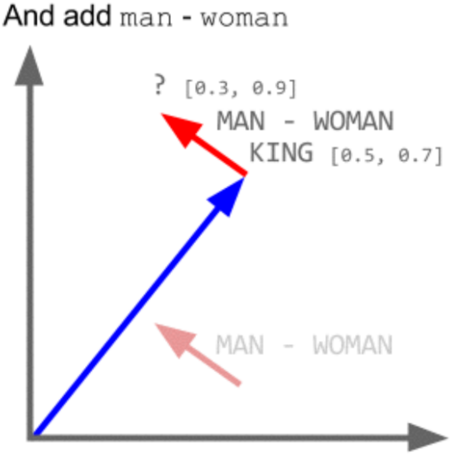
word2vec : nuts and bolts




Then take king
KING [0.5, 0.7]
MAN - WOMAN

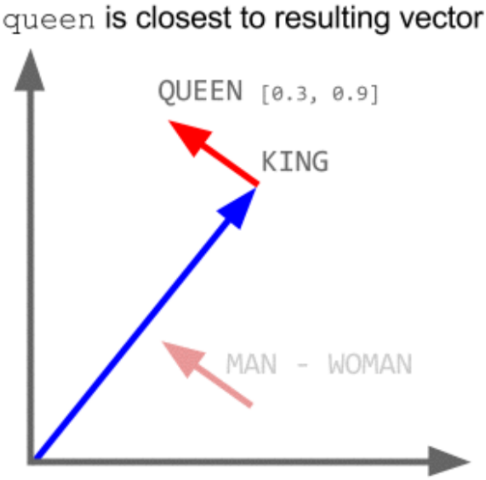
word2vec : nuts and bolts



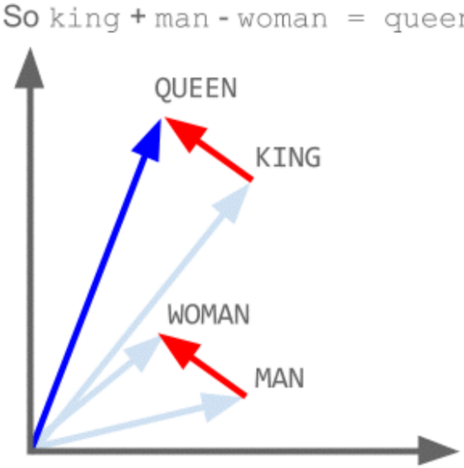

And add man - woman
? [0.3, 0.9]
MAN - WOMAN
KING [0.5, 0.7]
MAN - WOMAN


Find nearest word to result
? [0.3, 0.9]
KING
MAN - WOMAN

word2vec : nuts and bolts



word2vec : nuts and bolts



word2vec refinement : Improvement in performance

O(log(V)CN) : license to scale and consume enormouse amounts of data
So, word2vec is a:

- Language Modeling Algorithm
- predicts words locally - given one word it can predict the following word

Motivation

- LDA is able to create document (and topic) representations that are not so flexible but mostly interpretable to humans.
- LDA treats a set of documents as a set of documents, whereas word2vec works with a set of documents as with a very long text string.

In a nutshell

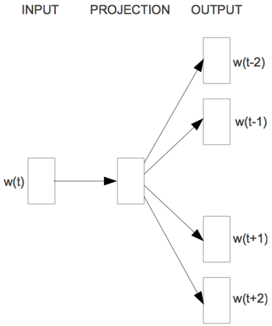
- word2vec models word-to-word relationships
- LDA models document-to-word relationships

Unsupervised Text Embedding

- CBOW (Mikolov et al. 2013)
- Skip-Gram (Mikolov et al. 2013)
- Paragraph Vector (Le et al. 2014)

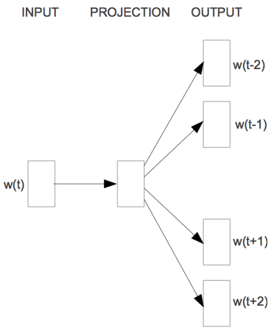
Unsupervised Text Embedding

- CBOW (Mikolov et al. 2013)
- Skip-Gram (Mikolov et al. 2013)
- Paragraph Vector (Le et al. 2014)



Unsupervised Text Embedding

- CBOW (Mikolov et al. 2013)
- Skip-Gram (Mikolov et al. 2013)
- Paragraph Vector (Le et al. 2014)

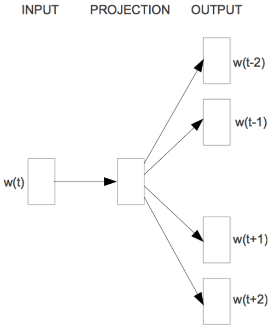


Pros

- Scalable, yet simple model
- Insensitive parameters
- Potential to leverage a large amount of unlabeled data, embeddings are general for different tasks

Unsupervised Text Embedding

- CBOW (Mikolov et al. 2013)
- Skip-Gram (Mikolov et al. 2013)
- Paragraph Vector (Le et al. 2014)



Pros

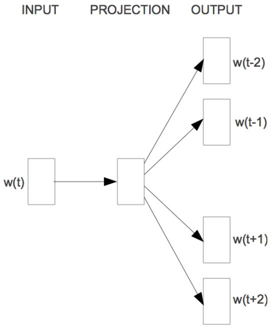
- Scalable, yet simple model
- Insensitive parameters
- Potential to leverage a large amount of unlabeled data, embeddings are general for different tasks

Cons

- Fully unsupervised, not tuned for specific tasks

Unsupervised Text Embedding

- CBOW (Mikolov et al. 2013)
- Skip-Gram (Mikolov et al. 2013)
- Paragraph Vector (Le et al. 2014)



Pros

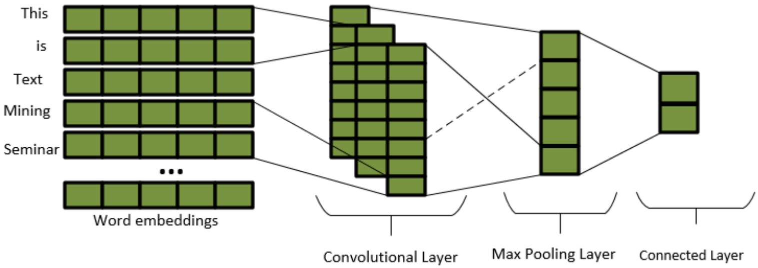
- Scalable, yet simple model
- Insensitive parameters
- Potential to leverage a large amount of unlabeled data, embeddings are general for different tasks

Cons

- Fully unsupervised, not tuned for specific tasks

(Deep) Neural Networks

- Recurrent Neural Networks (Mikolov et al. 2010)
- Recursive Neural Networks (Socher et al. 2012)
- Convolutional Neural Network (Kim et al. 2014)



Supervised Learning Model

- Recurrent Neural Networks (Mikolov et al. 2010)
- Recursive Neural Networks (Socher et al. 2012)
- Convolutional Neural Network (Kim et al. 2014)

Pros

- State-of-the-art performance on specific tasks

Cons

- Computationally expensive
- Require a large number of labeled data, hard to leverage unlabeled data
- *Very* sensitive to parameters, difficult to tune
- Potential to leverage a large amount of unlabeled data, embeddings are general for different tasks

Information Network Embedding

- Embedding one instance of some mathematical structure contained within another instance.
- Words that are used together with many similar words are likely to have similar meanings.

Information Network Embedding

- Embedding one instance of some mathematical structure contained within another instance.
- Words that are used together with many similar words are likely to have similar meanings.

The Ideal Embedding Model

- Reduced Parameter Space

The Ideal Embedding Model

- Reduced Parameter Space
- Improve generalization

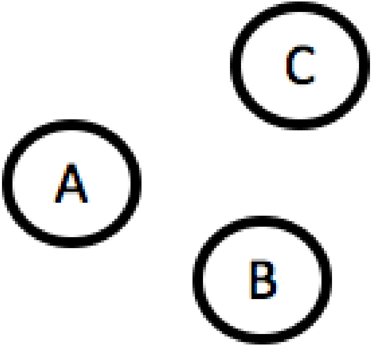
The Ideal Embedding Model

- Reduced Parameter Space
- Improve generalization
- Transfer or share knowledge between entities

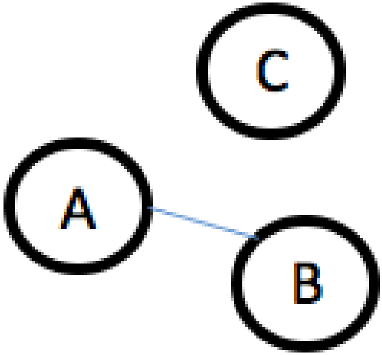
The Ideal Embedding Model

- Reduced Parameter Space
- Improve generalization
- Transfer or share knowledge between entities

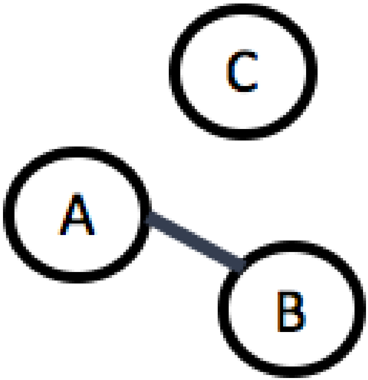
LINE : Intuition



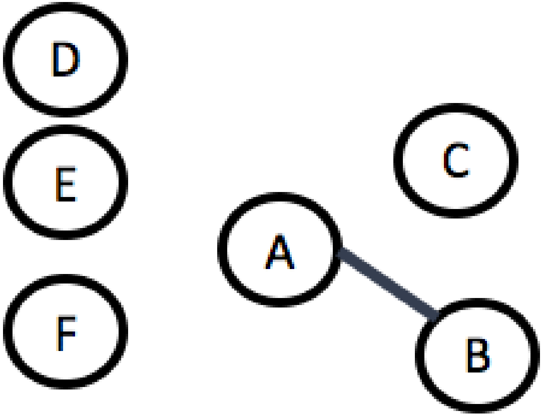
LINE : Intuition(contd)



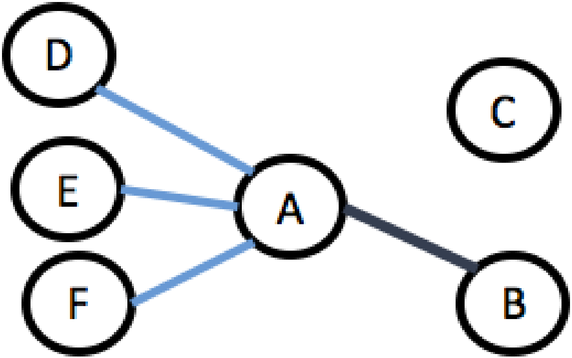
LINE : Intuition(contd)



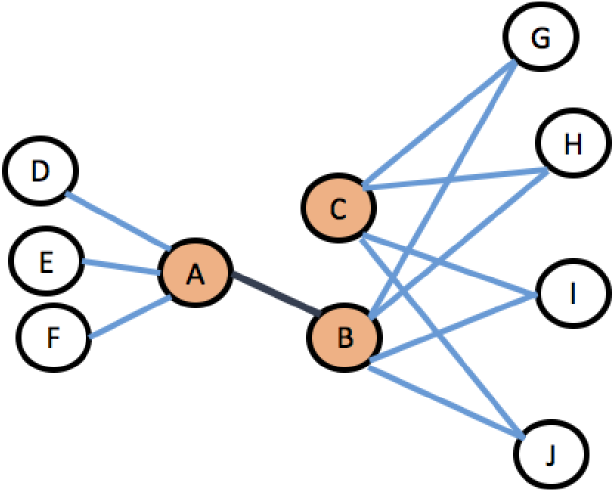
LINE : Intuition(contd)



LINE : Intuition(contd)



LINE : Intuition(contd)



Large-scale Information Network Embedding

– LINE extends the embedding idea to general information networks, more specifically, it transfers the vertices in a graph to vectors.

Large-scale Information Network Embedding

- LINE extends the embedding idea to general information networks, more specifically, it transfers the vertices in a graph to vectors.
- Preserves the *first-order* and *second-order* proximity between the vertices

Large-scale Information Network Embedding

- LINE extends the embedding idea to general information networks, more specifically, it transfers the vertices in a graph to vectors.
- Preserves the *first-order* and *second-order* proximity between the vertices
- **First-order proximity** : Observed Link between vertices

Large-scale Information Network Embedding

- LINE extends the embedding idea to general information networks, more specifically, it transfers the vertices in a graph to vectors.
- Preserves the *first-order* and *second-order* proximity between the vertices
- **First-order proximity** : Observed Link between vertices
- **Second-order proximity** : Proximity between their neighborhood structures

Large-scale Information Network Embedding

- LINE extends the embedding idea to general information networks, more specifically, it transfers the vertices in a graph to vectors.
- Preserves the *first-order* and *second-order* proximity between the vertices
- **First-order proximity** : Observed Link between vertices
- **Second-order proximity** : Proximity between their neighborhood structures

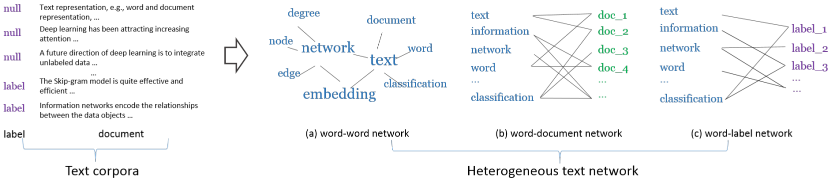
Predictive Text Embedding(PTE)

- Adapt the advantages of unsupervised text embedding approaches but naturally utilize the **labeled** data for specific tasks
- How to uniformly represent unsupervised and supervised information?

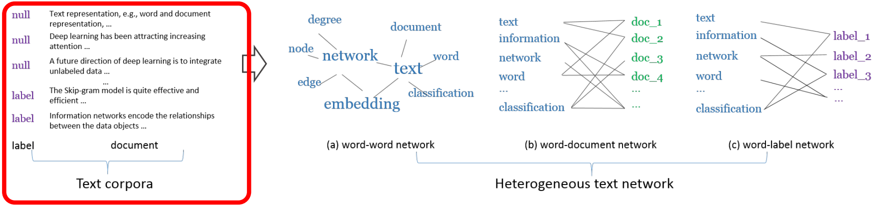
Predictive Text Embedding(PTE)

- Adapt the advantages of unsupervised text embedding approaches but naturally utilize the **labeled** data for specific tasks
- How to uniformly represent unsupervised and supervised information?
  - Heterogeneous Text Network
- Different Levels of Word Occurrences : *Word-Word Network, Word-Document Network, Word-Label Network*

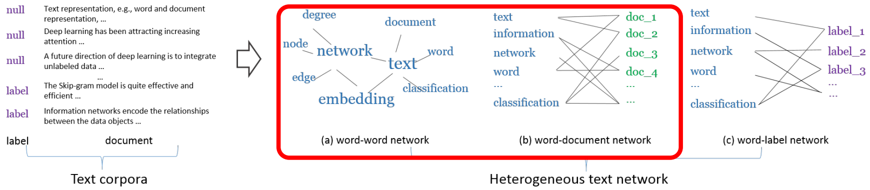
Converting Text Corpora



Partially Labeled Text Corpora

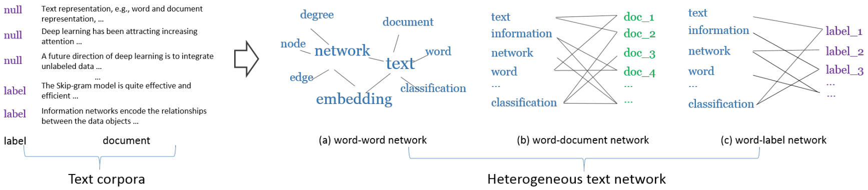


Word-Word and Word-Document Network



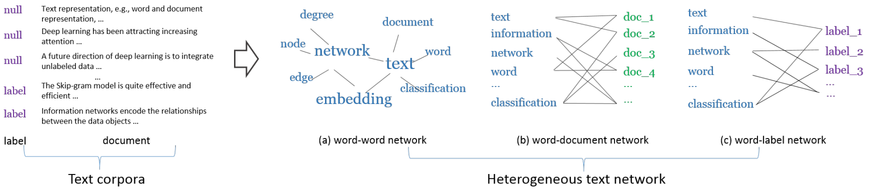
– Both word-document and word-word networks encode unsupervised information

Word-Word and Word-Document Network



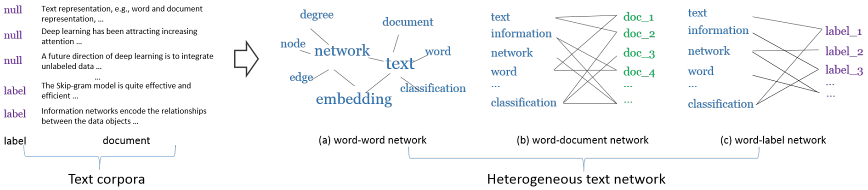
– Both word-document and word-word networks encode unsupervised information

Word-Word and Word-Document Network



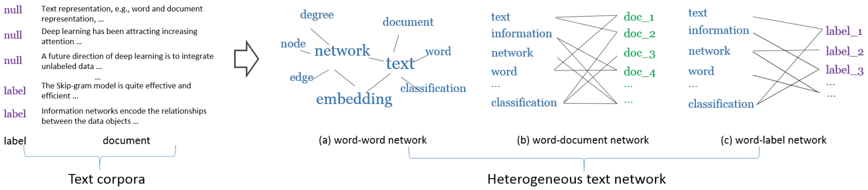
- Both word-document and word-word networks encode unsupervised information
- Word-Document $\equiv$ Topic Models, LDA

Word-Word and Word-Document Network



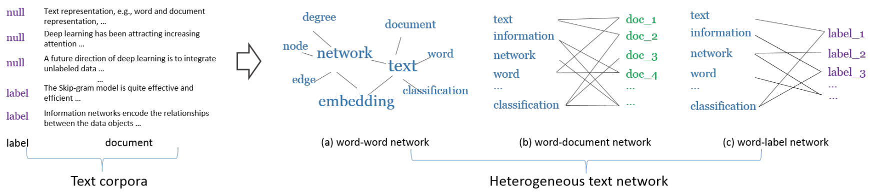
- Both word-document and word-word networks encode unsupervised information
- Word-Document $\equiv$ Topic Models, LDA
- Word-Word $\equiv$ Skip-Gram

Word-Word and Word-Document Network



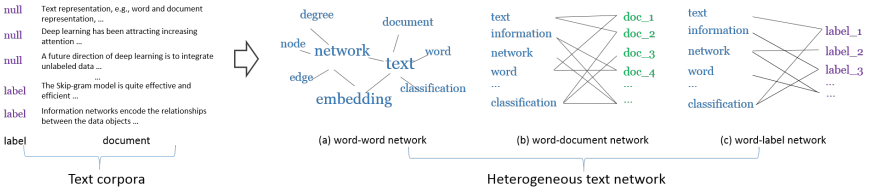
- Both word-document and word-word networks encode unsupervised information
- Word-Document $\equiv$ Topic Models, LDA
- Word-Word $\equiv$ Skip-Gram
- The weight between the word and document is its frequency in the document.

Word-Word and Word-Document Network



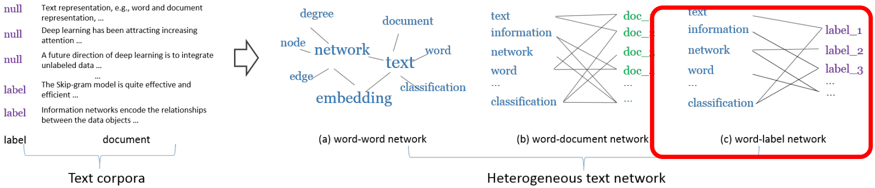
- Both word-document and word-word networks encode unsupervised information
- Word-Document $\equiv$ Topic Models, LDA
- Word-Word $\equiv$ Skip-Gram
- The weight between the word and document is its frequency in the document.
- The weight between two words is the number of times the two words co-occur in a *given window size*

Word-Word and Word-Document Network



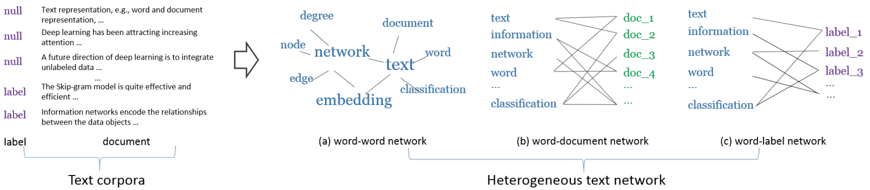
- Both word-document and word-word networks encode unsupervised information
- Word-Document $\equiv$ Topic Models, LDA
- Word-Word $\equiv$ Skip-Gram
- The weight between the word and document is its frequency in the document.
- The weight between two words is the number of times the two words co-occur in a *given window size*

Word Label Network



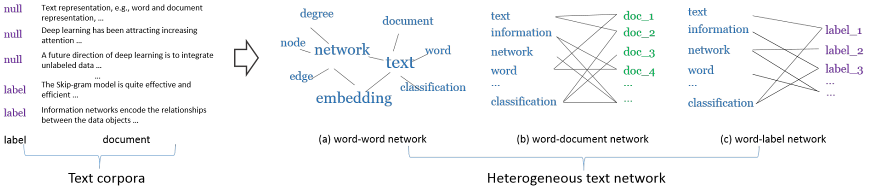
– Both word-document and word-word networks encode unsupervised information

Word Label Network



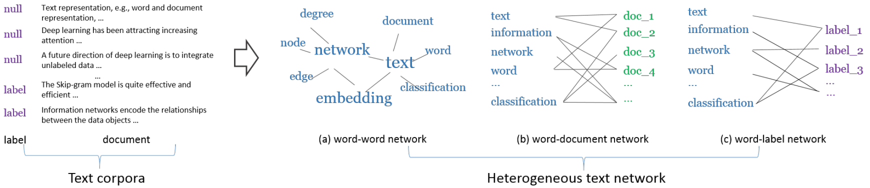
– Both word-document and word-word networks encode unsupervised information

Word Label Network



- Both word-document and word-word networks encode unsupervised information
- Word-Label Network encodes the supervised information

Word Label Network



- Both word-document and word-word networks encode unsupervised information
- Word-Label Network encodes the supervised information

Heterogeneous Text Network

Heterogeneous Text Network

- Three Bipartite Networks : Word-word(word-context), word-document and word-label network
- Encodes different levels of word co-occurency

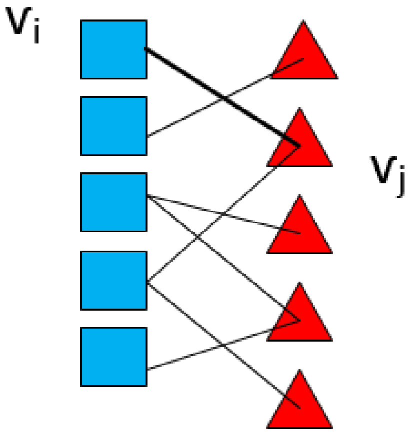
Heterogeneous Text Network

- Three Bipartite Networks : Word-word(word-context), word-document and word-label network
- Encodes different levels of word co-occurency
- Contains both `supervised` and `unsupervised` information

Heterogeneous Text Network

- Three Bipartite Networks : Word-word(word-context), word-document and word-label network
- Encodes different levels of word co-occurency
- Contains both `supervised` and `unsupervised` information
- Embedding a Heterogeneous Text Network, we obtain a very robust and optimized word embeddings for a specific task.

Bipartite Network Embedding



Bipartite Network Embedding(contd.)

What is the ideal proximity measure?
Hint : It's either **First Order** or **Second Order**

Bipartite Network Embedding(contd.)

What is the ideal proximity measure?
Hint : It's either **First Order** or **Second Order**

Bipartite Network Embedding : Second-order proximity

– For each edge (v_i, v_j), define a conditional probability:

$$p_2(v_j|v_i) = \frac{e^{\vec{u}_j'^T \cdot \vec{u}_i}}{\sum_{k=1}^{|V|} e^{\vec{u}_k'^T \cdot \vec{u}_i}} \tag{1}$$

$$O_2 = - \sum_{(i,j) \in E} w_{ij} \log p_2(v_j|v_i). \tag{2}$$

Heterogeneous Text Network Embedding

– Jointly embed three bipartite networks

Heterogeneous Text Network Embedding

- Jointly embed three bipartite networks
- Objective Function :

$$O_{pte} = O_{ww} + O_{wd} + O_{wl}, \tag{3}$$

where

$$O_{ww} = -\sum_{(i,j)\in E_{ww}} w_{ij} \log p(v_i|v_j) \tag{4}$$

$$O_{wd} = -\sum_{(i,j)\in E_{wd}} w_{ij} \log p(v_i|d_j) \tag{5}$$

$$O_{wl} = -\sum_{(i,j)\in E_{wl}} w_{ij} \log p(v_i|l_j) \tag{6}$$

Optimization

Two different ways of optimization : Depends on when **labeled data(word-label network)** are utilized.

- Joint Training
  - Train the unlabeled data and the labeled data simultaneously
- Pre-training + Fine-Tuning
  - Jointly train the G_{ww} and G_{wd} networks
  - Fine tuning the word embeddings with the word-label network

Learning Word Representations

- *Robust* and *Optimized* word Embeddings for *specific tasks*
  - Containing different levels of word co-occurences.
  - Encoding both supervised and unsupervised data
- Given an arbitrary textpiece $d = w_1 w_2 ... w_n$
- For every w_i, the text embedding is given by $\vec{u}_i$.
- The vector representation of the embedding can be computed as :

$$\vec{d} = \frac{1}{n} \sum_{i}^{n} \vec{u_i} \tag{7}$$

Evaluating Effectiveness of PTE

Task : Text Classification

- Embeddings as Features
- Classifier : Logistic Regression

Compared Algorithms

- **BOW** : Classical "bag-of-words" representation

Evaluating Effectiveness of PTE

Task : Text Classification

- Embeddings as Features
- Classifier : Logistic Regression

Compared Algorithms

- **BOW** : Classical "bag-of-words" representation
- **Unsupervised Text Embedding**
  - **Skip-Gram**
  - **Paragraph Vector(PV)**
  - **LINE**, applied to text networks

Evaluating Effectiveness of PTE

Task : Text Classification

- Embeddings as Features
- Classifier : Logistic Regression

Compared Algorithms

- **BOW** : Classical "bag-of-words" representation
- **Unsupervised Text Embedding**
  - **Skip-Gram**
  - **Paragraph Vector(PV)**
  - **LINE**, applied to text networks
- **Predictive Text Embedding : incorporates labels a.k.a Supervised Learning**

Evaluating Effectiveness of PTE

Task : Text Classification

- Embeddings as Features
- Classifier : Logistic Regression

Compared Algorithms

- **BOW** : Classical "bag-of-words" representation
- **Unsupervised Text Embedding**
  - **Skip-Gram**
  - **Paragraph Vector(PV)**
  - **LINE**, applied to text networks
- **Predictive Text Embedding : incorporates labels a.k.a Supervised Learning**
  - **CNN**
  - **PTE**

Unsupervised Embedding

Unsupervised Embedding

- Long Documents
  - Documemt-level word co-occurences are more useful than local Context-Level word co-occurences.
  - *No improvement observed* when these two co-occurences are combined.

Unsupervised Embedding

- Long Documents
  - Documemt-level word co-occurences are more useful than local Context-Level word co-occurences.
  - *No improvement observed* when these two co-occurences are combined.
- Short Documents
  - Local context-level word co-occurences are more useful than document-Level word co-occurences.
  - Combination further improves the embedding.

Summary

Summary

- Predictive Text Embedding
  - Adapt the advantages of unsupervised text embedding approaches
  - Naturally incorporate the labeled data
- Encode unsupervised and supervised information through Large-scale heterogeneous information networks
- Outperform or comparable to sophisticated methods such as CNN
  - Outperform CNN on long documents
  - Comparable to CNN on short documents

Takeaway

Predictive Text Embedding

Given a large collection of text data with unlabeled and *labeled* information, PTE aims to learn *low-dimensional* representations of words by **embedding** the **heterogeneous** representations of words by embedding the heterogeneous text network constructed from the collection into a low dimensional vector space.

Thank You !