Learning Distributed Document Representations for Multi-Label Document Categorization

Nitish Gupta

B.Tech - M.Tech Dual Degree

Thesis Defense

Electrical Engineering

IIT Kanpur

May 16, 2015

Outline

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class. For E.g. an article on Music Piracy

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class. For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.
- Multi-label classification belongs to a general class of supervised learning algorithms where :

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.
- Multi-label classification belongs to a general class of supervised learning algorithms where :
  - Training instances in the form of document-category pairs are used to learn a classifier $\mathcal{H}$

Intoduction to Multi-Label Document Categorization

- Text Documents usually belong to more than one conceptual class. For E.g. an article on Music Piracy
- Task of assigning documents to one or more predefined categories is called *Multi-Label Document Categorization*
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.
- Multi-label classification belongs to a general class of supervised learning algorithms where :
  - Training instances in the form of document-category pairs are used to learn a classifier $\mathcal{H}$
  - Learned classifier $\mathcal{H}$ is used to assign categories to new test documents

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$
- Training data for n ($n < |D|$) documents, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$
- Training data for n ($n < |D|$) documents, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$
  Each label vector $l_{d_i} \in \{0, 1\}^{|C|}$ denotes relevance of categories to the document d_i

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$
- Training data for n ($n < |D|$) documents, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$
  Each label vector $l_{d_i} \in \{0, 1\}^{|C|}$ denotes relevance of categories to the document d_i

Example :

Documents	Sports	Music	Arts	Technology	Literature	Politics
d_1	0	0	1	0	1	0
d_2	0	1	1	0	0	1
d_3	1	0	0	1	0	1
d_4	×	×	×	×	×	×
d_5	×	×	×	×	×	×

Introduction to Multi-Label Document Categorization

Given,

- A set of documents $D = \{d_1, \ldots, d_{|D|}\}$
- A set of categories $C = \{c_1, \ldots, c_{|C|}\}$
- Training data for n ($n < |D|$) documents, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$
  Each label vector $l_{d_i} \in \{0, 1\}^{|C|}$ denotes relevance of categories to the document d_i

Example :

Documents	**Sports**	**Music**	**Arts**	**Technology**	**Literature**	**Politics**
d_1	0	0	1	0	1	0
d_2	0	1	1	0	0	1
d_3	1	0	0	1	0	1
d_4	×	×	×	×	×	×
d_5	×	×	×	×	×	×

Using $\mathcal{T}$, D and C the learning algorithm learns a multi-label classifier $\mathcal{H}$ to estimate category label vectors, l_{d_j} ($j > n$) for the test documents.

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
    - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
    - Vectors (v_{d_i}) should encode the semantic content of the documents

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
   - Encoding documents in a k-dimensional space using such represenation is called the *Vector Space Model*

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
   - Encoding documents in a k-dimensional space using such represenation is called the *Vector Space Model*
   - The complete document set D can be represented by a document representation matrix $\mathrm{D} \in \mathbb{R}^{k \times |D|}$

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
   - Encoding documents in a k-dimensional space using such represenation is called the *Vector Space Model*
   - The complete document set D can be represented by a document representation matrix $\mathrm{D} \in \mathbb{R}^{k \times |D|}$

In this thesis, we focus on learning efficient document representations, D

Introduction to Multi-Label Document Categorization

Document Categorization task has the following two components :

1. *Learning Document Representations* : Representing text documents using numerical vectors that are inputs to the multi-label classifier $\mathcal{H}$
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
   - Encoding documents in a k-dimensional space using such represenation is called the *Vector Space Model*
   - The complete document set D can be represented by a document representation matrix $\mathrm{D} \in \mathbb{R}^{k \times |D|}$

In this thesis, we focus on learning efficient document representations, D

2. *Learning Algorithm* : Algorithm to learn the multi-label classifier $\mathcal{H}$

Background on Learning Algorithms

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier
   - k-Nearest Neighbor (k-NN)

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier
   - k-Nearest Neighbor (k-NN)
   - Linear Least Square Fit

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier
   - k-Nearest Neighbor (k-NN)
   - Linear Least Square Fit
   - Decision Trees

Background on Learning Algorithms

1. *Learning Multiple Binary Classifiers* :
   Algorithms that treat each category assignment independently and learn multiple binary classifiers, one for each category, to make the category assignments
   - Logistic Regression
   - Support Vector Machines (SVM)
   - Neural Networks
   - Naive Bayes
2. *Learning Single Joint Classifier* :
   Algorithms that jointly assign all the categories to a document d_i, i.e. estimate the complete label vector l_{d_i} using a single classifier
   - k-Nearest Neighbor (k-NN)
   - Linear Least Square Fit
   - Decision Trees
   - Generative Probabilistic Models

Background on Text Representation

Bag of Words Model

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (*tf*)
  - Inverse Document Frequency (*idf*)
  - Term Frequency - Inverse Document Frequency (*tf-idf*) : $tf \times idf$

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (*tf*)
  - Inverse Document Frequency (*idf*)
  - Term Frequency - Inverse Document Frequency (*tf-idf*) : $tf \times idf$

Drawbacks of the Bag-of-Words model

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (*tf*)
  - Inverse Document Frequency (*idf*)
  - Term Frequency - Inverse Document Frequency (*tf-idf*) : $tf \times idf$

Drawbacks of the Bag-of-Words model

- High-dimensionality

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (*tf*)
  - Inverse Document Frequency (*idf*)
  - Term Frequency - Inverse Document Frequency (*tf-idf*) : $tf \times idf$

Drawbacks of the Bag-of-Words model

- High-dimensionality
- Sparsity

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (*tf*)
  - Inverse Document Frequency (*idf*)
  - Term Frequency - Inverse Document Frequency (*tf-idf*) : $tf \times idf$

Drawbacks of the Bag-of-Words model

- High-dimensionality
- Sparsity
- Inability to encode word contexts

Background on Text Representation

Bag of Words Model

- Document d_i represented by $v_{d_i} \in \mathbb{R}^{|V|}$
- Each element in v_{d_i} denotes presence/absence of each word
- Weighing techniques employed to give importance to important terms
  - Term Frequency (*tf*)
  - Inverse Document Frequency (*idf*)
  - Term Frequency - Inverse Document Frequency (*tf-idf*) : $tf \times idf$

Drawbacks of the Bag-of-Words model

- High-dimensionality
- Sparsity
- Inability to encode word contexts
- Ignores word order

Background on Feature Selection / Dimensionality Reduction

Techniques to deal with sparsity and high-dimensionality in BOW

Background on Feature Selection / Dimensionality Reduction

Techniques to deal with sparsity and high-dimensionality in BOW

- Information Gain

$$G(t) = -\sum_{i=1}^{|C|} P(c_i)\log P(c_i) + P(t)\sum_{i=1}^{|C|} P(c_i|t)\log P(c_i|t) + P(\sim t)\sum_{i=1}^{|C|} P(c_i|\sim t)\log P(c_i|\sim t) \quad (1)$$

Background on Feature Selection / Dimensionality Reduction

Techniques to deal with sparsity and high-dimensionality in BOW

- Information Gain

$$G(t) = -\sum_{i=1}^{|C|} P(c_i) \log P(c_i) + P(t) \sum_{i=1}^{|C|} P(c_i|t) \log P(c_i|t) + P(\sim t) \sum_{i=1}^{|C|} P(c_i| \sim t) \log P(c_i| \sim t) \quad (1)$$

- Mutual Information

$$I(t,c) = \log \frac{P(t \wedge c)}{P(t) \times P(c)}, \qquad I_{avg}(t) = \sum_{i=1}^{|C|} P(c_i) I(t, c_i) \quad (2)$$

Background on Feature Selection / Dimensionality Reduction

Techniques to deal with sparsity and high-dimensionality in BOW

- Information Gain

$$G(t) = -\sum_{i=1}^{|C|} P(c_i)\log P(c_i) + P(t)\sum_{i=1}^{|C|} P(c_i|t)\log P(c_i|t) + P(\sim t)\sum_{i=1}^{|C|} P(c_i|\sim t)\log P(c_i|\sim t) \quad (1)$$

- Mutual Information

$$I(t,c) = \log \frac{P(t \wedge c)}{P(t) \times P(c)}, \qquad I_{avg}(t) = \sum_{i=1}^{|C|} P(c_i) I(t, c_i) \quad (2)$$

- Latent Semantic Indexing (LSI)

$$X = TSD^T \quad (3)$$

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ $(k \in [50, 300])$

Need for Distributed Word Representations

- Curse of Dimensionality

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ $(k \in [50, 300])$

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary
  - Parameters in language modelling grow exponentially with the size of vocabulary

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ $(k \in [50, 300])$

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary
  - Parameters in language modelling grow exponentially with the size of vocabulary
- No Word Similarity Measure

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ $(k \in [50, 300])$

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary
  - Parameters in language modelling grow exponentially with the size of vocabulary
- No Word Similarity Measure
  - One-hot representations are orthogonal representations

Distributed Word Representations

Representating each word w_i using vector $v_{w_i} \in \mathbb{R}^k$ ($k \in [50, 300]$)

Need for Distributed Word Representations

- Curse of Dimensionality
  - One-hot representations grow with the size of vocabulary
  - Parameters in language modelling grow exponentially with the size of vocabulary
- No Word Similarity Measure
  - One-hot representations are orthogonal representations
  - Cannot capture semantic similarity between words

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors

Neural Probabilistic Language Model

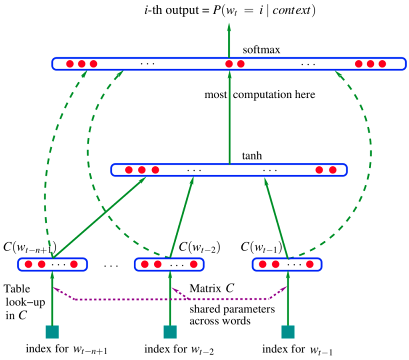
Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

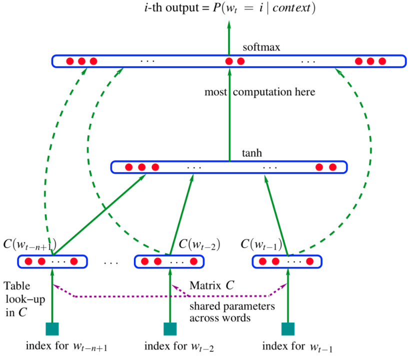
1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language



Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language

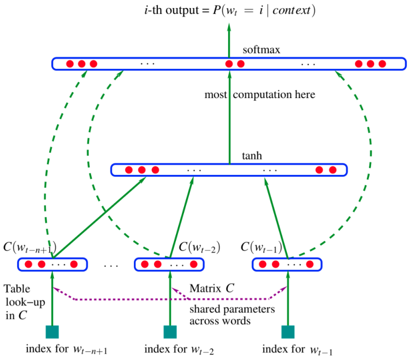


$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \qquad (4)$$

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language



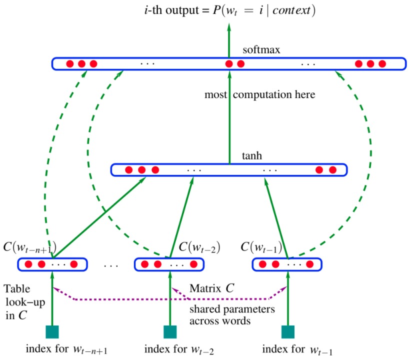
$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \qquad (4)$$

$$y = b + U tanh(d + Hx), \quad y \in \mathbb{R}^{|V|} \qquad (5)$$

Neural Probabilistic Language Model

Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. A probability function that, using these word vectors, learns a statistical model of language



$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \tag{4}$$

$$y = b + U tanh(d + Hx), \quad y \in \mathbb{R}^{|V|} \tag{5}$$

$$P(w_t = i|w_{t-1}, \ldots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{6}$$

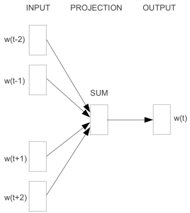
Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [6]. These models use word vectors to predict other words in the context.

Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [6]. These models use word vectors to predict other words in the context.

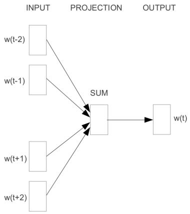
1. Continuous Bag-of-Words Model



Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [6]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model

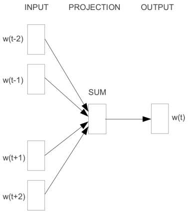


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [6]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model



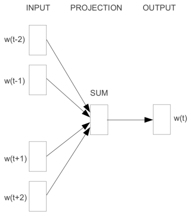
$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [6]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model



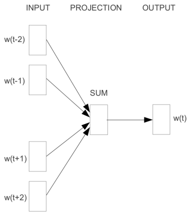
$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

$$P(w_t | w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \quad (9)$$

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [6]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model

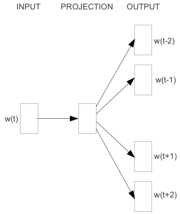


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

$$P(w_t | w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \quad (9)$$

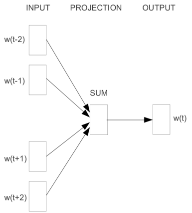
2. Skip-Gram Model



Log-Linear Models

Log-Linear Models for learning distributed word vectors are proposed in Mikolov et al. [6]. These models use word vectors to predict other words in the context.

1. Continuous Bag-of-Words Model

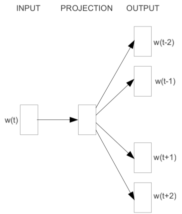


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

$$P(w_t | w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \quad (9)$$

2. Skip-Gram Model



$$P(w_{t+j} | w_t) = \frac{e^{(v_{w_t} \cdot v_{w_{t+j}})}}{\sum_i e^{(v_{w_t} \cdot v_{w_i})}} \quad (10)$$

Distributed Document Representations

Motivation for learning distributed document representations

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering
3. Compositionality of word vectors beyond weighted average [8, 14, 13, 4, 7] is not simple

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering
3. Compositionality of word vectors beyond weighted average [8, 14, 13, 4, 7] is not simple
4. Socher et al. [12] propose a Recursive Tensor Neural Network (RTNN) to compose word vectors for learning sentence representations using the parse-tree of the sentence in a bottom-up fashion

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering
3. Compositionality of word vectors beyond weighted average [8, 14, 13, 4, 7] is not simple
4. Socher et al. [12] propose a Recursive Tensor Neural Network (RTNN) to compose word vectors for learning sentence representations using the parse-tree of the sentence in a bottom-up fashion
   - Parsing, a computationally expensive step required for each sentence

Distributed Document Representations

Motivation for learning distributed document representations

1. Traditional representations do not encode semantic similarity between documents. Therefore, cannot handle synonyms
2. Drawbacks in BOW like sparsity, high-dimensionality, inability to encode context information and consider word ordering
3. Compositionality of word vectors beyond weighted average [8, 14, 13, 4, 7] is not simple
4. Socher et al. [12] propose a Recursive Tensor Neural Network (RTNN) to compose word vectors for learning sentence representations using the parse-tree of the sentence in a bottom-up fashion
    - Parsing, a computationally expensive step required for each sentence
    - Composing sentence vectors to represent documents is not straight-forward

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Hypothesis

> *Document Representations that encode semantic content of the document should be able to predict words in the document*

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Hypothesis

> *Document Representations that encode semantic content of the document should be able to predict words in the document*

Our model,

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Hypothesis

> *Document Representations that encode semantic content of the document should be able to predict words in the document*

Our model,

1. Learns distributed representations for document (and words) that encode the different semantic content in the documents

Our Model for Learning Document Representations

Inspired by the log-linear models to learn word vectors, we present model, to learn universal distributed representations for documents and words

Hypothesis

Document Representations that encode semantic content of the document should be able to predict words in the document

Our model,

1. Learns distributed representations for document (and words) that encode the different semantic content in the documents
2. Embeds documents and words in the same k-dimensional space such that semantically similar entities have similar vector representations

Our Model for Learning Document Representations

We present an unsupervised neural network model that,

Our Model for Learning Document Representations

We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
   Vectors are stored as columns of the matrix $\mathrm{D} = \left[\mathrm{v}_1^D, \ldots, \mathrm{v}_{|D|}^D\right] \in \mathbb{R}^{k \times |D|}$

Our Model for Learning Document Representations

We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
   Vectors are stored as columns of the matrix $\mathrm{D} = \left[\mathrm{v}_1^D, \ldots, \mathrm{v}_{|D|}^D\right] \in \mathbb{R}^{k \times |D|}$

2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$ Vectors are stored as columns of the matrix $\mathrm{W} = \left[\mathrm{v}_1^W, \ldots, \mathrm{v}_{|V|}^W\right] \in \mathbb{R}^{k \times |V|}$

Our Model for Learning Document Representations

We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
   Vectors are stored as columns of the matrix $\mathrm{D} = \left[\mathrm{v}_1^D, \ldots, \mathrm{v}_{|D|}^D\right] \in \mathbb{R}^{k \times |D|}$

2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$ Vectors are stored as columns of the matrix $\mathrm{W} = \left[\mathrm{v}_1^W, \ldots, \mathrm{v}_{|V|}^W\right] \in \mathbb{R}^{k \times |V|}$

3. Given a sequence of words, $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, estimates

$$p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$$

Our Model for Learning Document Representations

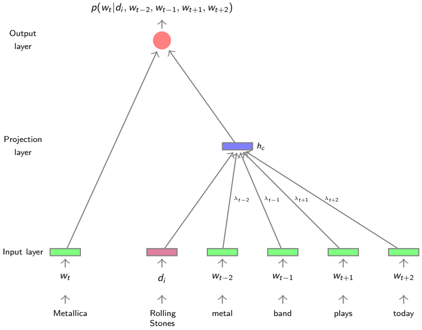
We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
   Vectors are stored as columns of the matrix $\mathrm{D} = \left[\mathrm{v}_1^D, \ldots, \mathrm{v}_{|D|}^D\right] \in \mathbb{R}^{k \times |D|}$
2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$ Vectors are stored as columns of the matrix $\mathrm{W} = \left[\mathrm{v}_1^W, \ldots, \mathrm{v}_{|V|}^W\right] \in \mathbb{R}^{k \times |V|}$
3. Given a sequence of words, $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, estimates

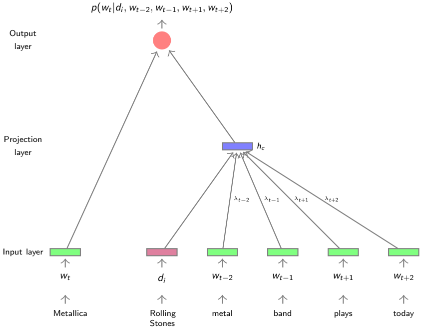
$$p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$$

4. Maximizes the probability of predicting the words correctly to learn D and W and the parameters of the probability function

Our Model for Learning Document Representations

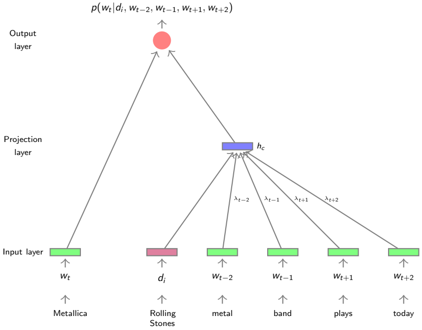


Our Model for Learning Document Representations



Context Representation :

$$h_c = \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t-1}\mathrm{v}^W_{w_{t-1}} + \lambda_{t+1}\mathrm{v}^W_{w_{t+1}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}} \quad (11)$$



Context Representation :

$$h_c = \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t-1}\mathrm{v}^W_{w_{t-1}} + \lambda_{t+1}\mathrm{v}^W_{w_{t+1}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}} \tag{11}$$

Probability Estimation :

$$s_{w_i} = \sigma(\mathrm{v}^W_{w_i} \cdot h_c), \quad \sigma(x) = \frac{1}{1+e^{-x}} \tag{12}$$

$$p(w_t|d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}) = \frac{e^{s_{w_t}}}{\sum_{i \in V} e^{s_{w_i}}} \tag{13}$$

Training Objective

1. Training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}\}_{m=1}^{m=M}$

Training Objective

1. Training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}\}_{m=1}^{m=M}$
2. Learn optimum parameter set $\Theta = (\mathrm{D}, \mathrm{W}, \Lambda)$, i.e. document and word vectors and the neural network weights Λ

Training Objective

1. Training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}\}_{m=1}^{m=M}$
2. Learn optimum parameter set $\Theta = (\mathrm{D}, \mathrm{W}, \Lambda)$, i.e. document and word vectors and the neural network weights Λ
3. Maximize average log-probability of predicting w_t correctly in each sequence in $\mathcal{T}$

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{14}$$

$$l(\mathcal{T}, \Theta) = \frac{1}{M} \sum_{m=1}^{M} \log \left[p(w_t^{(m)} | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t-1}^{(m)}, w_{t+1}^{(m)}, \ldots, w_{t+c}^{(m)}) \right] \tag{15}$$

Training Objective

1. Training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}\}_{m=1}^{m=M}$
2. Learn optimum parameter set $\Theta = (\mathrm{D}, \mathrm{W}, \Lambda)$, i.e. document and word vectors and the neural network weights Λ
3. Maximize average log-probability of predicting w_t correctly in each sequence in $\mathcal{T}$

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{14}$$

$$l(\mathcal{T}, \Theta) = \frac{1}{M} \sum_{m=1}^{M} \log \left[p(w_t^{(m)} | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t-1}^{(m)}, w_{t+1}^{(m)}, \ldots, w_{t+c}^{(m)}) \right] \tag{15}$$

4. Use Stochastic Gradient Descent (SGD) to update parameters

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{16}$$

Noise Contrastive Estimation

1. Computing soft-max for each training sequence is expensive, $\mathcal{O}(V)$

Noise Contrastive Estimation

1. Computing soft-max for each training sequence is expensive, $\mathcal{O}(V)$
2. Speed-ups in softmax computation can be attained using Hierarchical soft-max [11] and importance sampling to approximate the likelihood gradient [3, 1]

Noise Contrastive Estimation

1. Computing soft-max for each training sequence is expensive, $\mathcal{O}(V)$
2. Speed-ups in softmax computation can be attained using Hierarchical soft-max [11] and importance sampling to approximate the likelihood gradient [3, 1]
    - Finding well-performing trees in Hierarchical soft-max is not trivial

Noise Contrastive Estimation

1. Computing soft-max for each training sequence is expensive, $\mathcal{O}(V)$
2. Speed-ups in softmax computation can be attained using Hierarchical soft-max [11] and importance sampling to approximate the likelihood gradient [3, 1]
    - Finding well-performing trees in Hierarchical soft-max is not trivial
    - Importance sampling suffers from stability issues
3. **Noise Contrastive Estimation** (NCE) [5] fits unnormalized probabilities

Noise Contrastive Estimation

1. Computing soft-max for each training sequence is expensive, $\mathcal{O}(V)$
2. Speed-ups in softmax computation can be attained using Hierarchical soft-max [11] and importance sampling to approximate the likelihood gradient [3, 1]
    - Finding well-performing trees in Hierarchical soft-max is not trivial
    - Importance sampling suffers from stability issues
3. **Noise Contrastive Estimation** (NCE) [5] fits unnormalized probabilities
    - Reduces the problem of *probability density estimation* to *probabilistic binary classification*

Noise Contrastive Estimation

1. Computing soft-max for each training sequence is expensive, $\mathcal{O}(V)$
2. Speed-ups in softmax computation can be attained using Hierarchical soft-max [11] and importance sampling to approximate the likelihood gradient [3, 1]
    - Finding well-performing trees in Hierarchical soft-max is not trivial
    - Importance sampling suffers from stability issues
3. **Noise Contrastive Estimation** (NCE) [5] fits unnormalized probabilities
    - Reduces the problem of *probability density estimation* to *probabilistic binary classification*
    - Adaptation to NPLM [10] and learning word embeddings [9] show significant training time speed-ups

References

[1] Y. Bengio and J.-S. Senecal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4):713–722, 2008.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[3] Y. Bengio, J.-S. Senécal, et al. Quick training of probabilistic neural nets by importance sampling. In *AISTATS Conference*, 2003.

[4] E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni. Multi-step regression learning for compositional distributional semantics. *arXiv preprint arXiv:1301.6939*, 2013.

[5] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361, 2012.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.