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

1. Multi-Label Document Categorization
2. Related Work
    - Text Representations
    - Learning Algorithms
3. Distributed Word Representations
4. Learning Distributed Document Representations
5. Document Categorization Algorithm
6. Results
7. Conclusion and Future Work

Introduction to Multi-Label Document Categorization

Document Categorization is the task of assigning categories to documents

Introduction to Multi-Label Document Categorization

Document Categorization is the task of assigning categories to documents

Why need Multi-Label Document Categorization?

Introduction to Multi-Label Document Categorization

Document Categorization is the task of assigning categories to documents

Why need Multi-Label Document Categorization?

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

Introduction to Multi-Label Document Categorization

Document Categorization is the task of assigning categories to documents

Why need Multi-Label Document Categorization?

- Text Documents usually belong to more than one conceptual class.
  For E.g. an article on Music Piracy
- Wide range real-world applications :
  - Web-page tagging
  - Medical Patient Record Management
  - Wikipedia Article Management
  - Document Recommendation etc.

Introduction to Multi-Label Document Categorization

Multi-label classification belongs to a general class of *supervised learning* algorithms where, 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, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$

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

Multi-label classification belongs to a general class of *supervised learning* algorithms where, 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, $\mathcal{T} = \{l_{d_1}, \ldots, l_{d_n}\}$

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*
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Embedding documents in a k-dimensional space is called the *Vector Space Model*
   - The set D can be represented by a matrix $\mathrm{D} \in \mathbb{R}^{k \times |D|}$
   - 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*
   - Each document $d_i \in D$ is represented using a vector $v_{d_i} \in \mathbb{R}^k$
   - Embedding documents in a k-dimensional space is called the *Vector Space Model*
   - The set D can be represented by a matrix $\mathrm{D} \in \mathbb{R}^{k \times |D|}$
   - Vectors (v_{d_i}) should encode the semantic content of the documents
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*
   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
   - Support Vector Machines (SVM)
   - Neural Networks, E.g. CLASSI, NNet.PARC
   - 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, E.g. CLASSI, NNet.PARC
   - 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, E.g. CLASSI, NNet.PARC
   - 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

- 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

- High-dimensionality
- Sparsity
- Inability to encode word contexts
- Ignores word order
- Lack of similarity measures

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

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

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)

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

X is the Term-Document Matrix

Distributed Word Representations

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

Distributed Word Representations

Representation of 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

Representation of 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 modeling grow exponentially with the size of vocabulary

Distributed Word Representations

Representation of 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 modeling 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

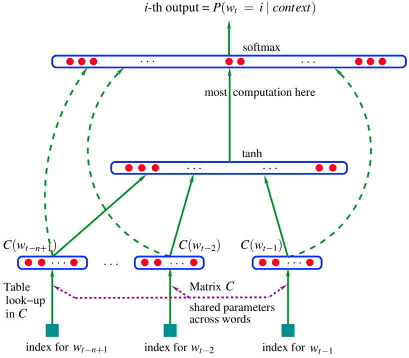
Bengio et al. [2] developed *Neural Probabilistic Language Model (NPLM)* to learn

1. Distributed word vectors
2. Probability function to learn a statistical model of language

Neural Probabilistic Language Model

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

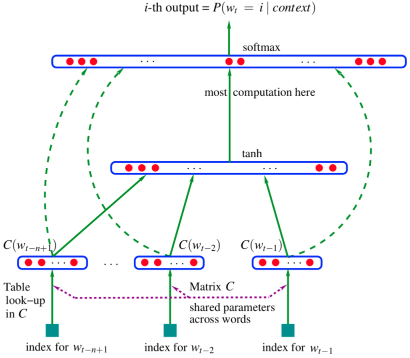
1. Distributed word vectors
2. Probability function to learn a statistical model of language



Neural Probabilistic Language Model

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

1. Distributed word vectors
2. Probability function to learn a statistical model of language

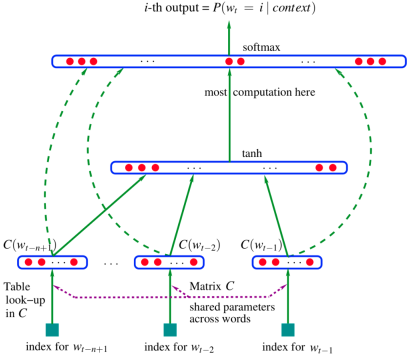


$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \tag{4}$$

Neural Probabilistic Language Model

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

1. Distributed word vectors
2. Probability function to learn a statistical model of language



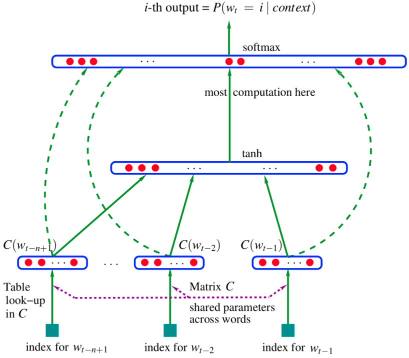
$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \qquad (4)$$

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

Neural Probabilistic Language Model

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

1. Distributed word vectors
2. Probability function to learn a statistical model of language



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

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

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

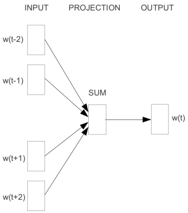
Log-Linear Models

Proposed by Mikolov et al. [10] to predict words in the context using word vectors

Log-Linear Models

Proposed by Mikolov et al. [10] to predict words in the context using word vectors

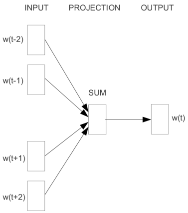
1. Continuous Bag-of-Words Model



Log-Linear Models

Proposed by Mikolov et al. [10] to predict words in the context using word vectors

1. Continuous Bag-of-Words Model

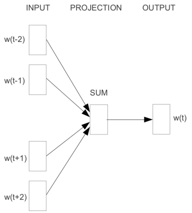


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

Log-Linear Models

Proposed by Mikolov et al. [10] to predict words in the context using word vectors

1. Continuous Bag-of-Words Model



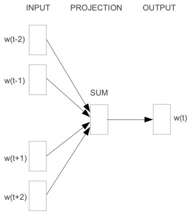
$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

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

Log-Linear Models

Proposed by Mikolov et al. [10] to predict words in the context using word vectors

1. Continuous Bag-of-Words Model



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

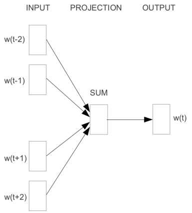
$$y = b + Uh, \quad y \in \mathbb{R}^{|V|} \quad (8)$$

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

Log-Linear Models

Proposed by Mikolov et al. [10] to predict words in the context using word vectors

1. Continuous Bag-of-Words Model

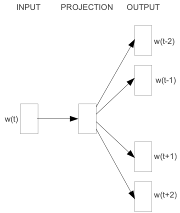


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

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

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

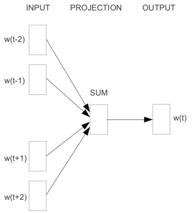
2. Skip-Gram Model



Log-Linear Models

Proposed by Mikolov et al. [10] to predict words in the context using word vectors

1. Continuous Bag-of-Words Model

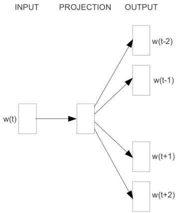


$$h = w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k} \quad (7)$$

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

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

2. Skip-Gram Model



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

Motivation for learning distributed document representations

Motivation for learning distributed document representations

1. Lack of semantic similarity measures. Therefore, cannot handle synonyms

Distributed Document Representations

Motivation for learning distributed document representations

1. Lack of semantic similarity measures. 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. Lack of semantic similarity measures. 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 [12, 18, 17, 6, 11]

Distributed Document Representations

Motivation for learning distributed document representations

1. Lack of semantic similarity measures. 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 [12, 18, 17, 6, 11]
4. Recursive Tensor Neural Network (RTNN) [16] for learning sentence representations using the syntactic dependency has issues
    - 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,

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 $\mathrm{v}_i^D \in \mathbb{R}^k$

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 $\mathrm{v}_i^D \in \mathbb{R}^k$
2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$

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 $\mathrm{v}_i^D \in \mathbb{R}^k$
2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$
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

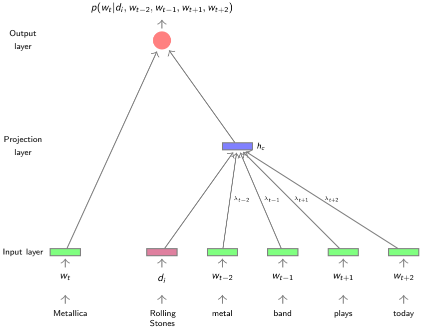
We present an unsupervised neural network model that,

1. Represents each document $d_i \in D$ by a vector $\mathrm{v}_i^D \in \mathbb{R}^k$
2. Each word $w_i \in W$, is represented by a vector $\mathrm{v}_i^W \in \mathbb{R}^k$
3. Given a sequence of words, $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, estimates

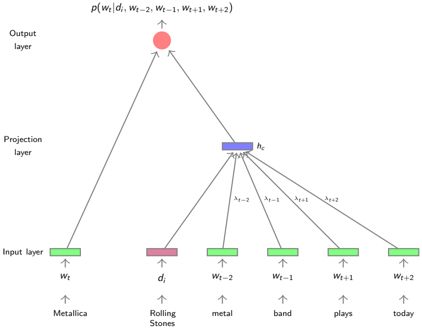
$$p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$$

4. Maximizes probability of predicting the middle word correctly to learn vectors

Our Model for Learning Document Representations



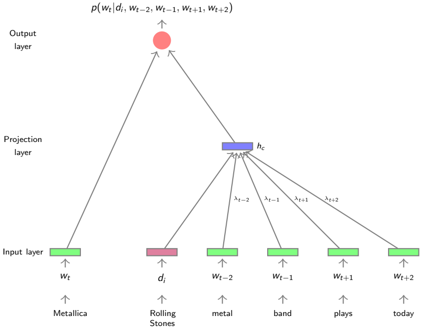
Our Model for Learning Document Representations



Context Representation :

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

Our Model for Learning Document Representations



Context Representation :

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

Probability Estimation :

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

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

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. Soft-max computation is expensive, $\mathcal{O}(V)$

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$
2. Speed-ups using **Hierarchical soft-max** [15] 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

Noise Contrastive Estimation

1. Soft-max computation is expensive, $\mathcal{O}(V)$
2. Speed-ups using **Hierarchical soft-max** [15] 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) [8] fits unnormalized probabilities
   - Reduces the problem of *probability density estimation* to *probabilistic binary classification*
   - Adaptation to NPLM [14] and learning word embeddings [13] show significant training time speed-ups

Noise Contrastive Estimation (contd.)

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

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

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

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

1. New labeled training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)} = 1\}_{m=1}^{m=M}$
2. For every positive training sequence, n negative training sequences introduced where,
   - The observed middle word w_t is replaced by a corrupt word w_x drawn from a noise distribution $P_n(w)$
   - E.g. $\{d_i, w_{t-c}, \ldots, w_{t-1}, w_x, w_{t+1}, \ldots, w_{t+c}, Y = 0\}$

Noise Contrastive Estimation (contd.)

1. Given a sequence of words $(w_{t-c}, \dots, w_{t+c})$ in document d_i
   - *Earlier objective* : Maximize $p(w_t | d_i, w_{t-c}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+c})$
   - *New objective* : Build binary classifier to distinguish between correct middle word w_t and random corrupt word

For NCE Binary Classification Objective :

1. New labeled training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \dots, w_{t+c}^{(m)}, Y^{(m)} = 1\}_{m=1}^{m=M}$
2. For every positive training sequence, n negative training sequences introduced where,
   - The observed middle word w_t is replaced by a corrupt word w_x drawn from a noise distribution $P_n(w)$
   - E.g. $\{d_i, w_{t-c}, \dots, w_{t-1}, w_x, w_{t+1}, \dots, w_{t+c}, Y = 0\}$
3. Complete training data : $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \dots, w_{t+c}^{(m)}, Y^{(m)}\}_{m=1}^{m=M+nM}$

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \dots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

- We build a probabilistic binary classifier to predict the label Y

$$P(Y = 1 | d_i, w_{t-c}, \dots, w_{t+c}, \Theta) = \sigma(\mathrm{v}_{w_t}^{W} \cdot h_c) \tag{17}$$

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

- We build a probabilistic binary classifier to predict the label Y

$$P(Y = 1 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = \sigma(\mathrm{v}_{w_t}^{W} \cdot h_c) \tag{17}$$

$$P(Y = 0 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = 1 - \sigma(\mathrm{v}_{w_t}^{W} \cdot h_c) \tag{18}$$

Noise Contrastive Estimation (contd.)

Given a sequence of words $(w_{t-c}, \ldots, w_{t+c})$ in document d_i, our new objective is to predict whether the sequence is legitimate

- We build a probabilistic binary classifier to predict the label Y

$$P(Y = 1 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = \sigma(\mathrm{v}_{w_t}^{W} \cdot h_c) \tag{17}$$

$$P(Y = 0 | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = 1 - \sigma(\mathrm{v}_{w_t}^{W} \cdot h_c) \tag{18}$$

$$P(Y | d_i, w_{t-c}, \ldots, w_{t+c}, \Theta) = [\sigma(\mathrm{v}_{w_t}^{W} \cdot h_c)]^{Y} [1 - \sigma(\mathrm{v}_{w_t}^{W} \cdot h_c)]^{1-Y} \tag{19}$$

Learning Objective with NCE

Given the training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)}\}_{m=1}^{m=M+nM}$, we maximize the log-likelihood of observing it

$P_\Theta(Y_m)$ is a shorthand notation for $P(Y_m | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, \Theta)$
Y_m is the predicted label

Learning Objective with NCE

Given the training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)}\}_{m=1}^{m=M+nM}$, we maximize the log-likelihood of observing it

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{20}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{M+nM} \log P_{\Theta}(Y_m = Y^{(m)}) \tag{21}$$

$P_{\Theta}(Y_m)$ is a shorthand notation for $P(Y_m | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, \Theta)$
Y_m is the predicted label

Learning Objective with NCE

Given the training data $\mathcal{T} = \{d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, Y^{(m)}\}_{m=1}^{m=M+nM}$, we maximize the log-likelihood of observing it

$$\hat{\Theta} = \arg\max_{\Theta} \ l(\mathcal{T}, \Theta) \tag{20}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{M+nM} \log P_{\Theta}(Y_m = Y^{(m)}) \tag{21}$$

The logarithm of the probability estimate is given by,

$$\log P_{\Theta}(Y_m = Y^{(m)}) = Y^{(m)} \log \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)}) + (1 - Y^{(m)}) \log(1 - \sigma(\mathrm{v}_{w_t^{(m)}}^{W} \cdot h_c^{(m)})) \tag{22}$$

$P_{\Theta}(Y_m)$ is a shorthand notation for $P(Y_m | d_i^{(m)}, w_{t-c}^{(m)}, \ldots, w_{t+c}^{(m)}, \Theta)$
Y_m is the predicted label

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

$d = \mathrm{v}_{w_t}^{W} \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$d = \mathrm{v}_{w_t}^{W} \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \frac{\partial \sigma(d^{(m)})}{\partial \theta} \tag{24}$$

$d = \mathrm{v}_{w_t}^{W} \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \frac{\partial \sigma(d^{(m)})}{\partial \theta} \tag{24}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \left[\sigma(d^{(m)})(1 - \sigma(d^{(m)})) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{25}$$

$d = \mathrm{v}_{w_t}^{W} \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \frac{\partial \sigma(d^{(m)})}{\partial \theta} \tag{24}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \left[\sigma(d^{(m)})(1 - \sigma(d^{(m)})) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{25}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} - \sigma(d^{(m)}) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{26}$$

$d = \mathrm{v}_{w_t}^{W} \cdot h_c$, is the pre-sigmoid activation

Parameter Estimation

We use SGD to learn parameters i.e. document and word vectors and the neural network weights

$$\theta_i^{(x)} = \theta_i^{(x-1)} + \gamma \frac{\partial l(\mathcal{T}, \Theta)}{\partial \theta_i} \tag{23}$$

Gradient of $\log P_\Theta(Y_m = Y^{(m)})$ with respect to parameter θ,

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \frac{\partial \sigma(d^{(m)})}{\partial \theta} \tag{24}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} \frac{1}{\sigma(d^{(m)})} - (1 - Y^{(m)}) \frac{1}{(1 - \sigma(d^{(m)}))} \right] \left[\sigma(d^{(m)})(1 - \sigma(d^{(m)})) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{25}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} - \sigma(d^{(m)}) \right] \frac{\partial d^{(m)}}{\partial \theta} \tag{26}$$

$$\frac{\partial \log P_\Theta(Y_m = Y^{(m)})}{\partial \theta} = \left[Y^{(m)} - \sigma(\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)}) \right] \frac{\partial (\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)})}{\partial \theta} \tag{27}$$

$d = \mathrm{v}^W_{w_t} \cdot h_c$, is the pre-sigmoid activation

Update rule for Parameters

1. Document Vector :

$$(\mathrm{v}^{D}_{d_i^{(m)}})^{(i+1)} = (\mathrm{v}^{D}_{d_i^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^{W}_{w_t^{(m)}} \cdot h_c^{(m)}))\mathrm{v}^{W}_{w_t^{(m)}} - \beta \mathrm{v}^{D}_{d_i^{(m)}} \right] \tag{28}$$

Update rule for Parameters

1. Document Vector :

$$(\mathrm{v}^{D}_{d_i^{(m)}})^{(i+1)} = (\mathrm{v}^{D}_{d_i^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^{W}_{w_t^{(m)}} \cdot h_c^{(m)}))\mathrm{v}^{W}_{w_t^{(m)}} - \beta \mathrm{v}^{D}_{d_i^{(m)}} \right] \tag{28}$$

2. Middle Word Vector :

$$(\mathrm{v}^{W}_{w_t^{(m)}})^{(i+1)} = (\mathrm{v}^{W}_{w_t^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^{W}_{w_t^{(m)}} \cdot h_c^{(m)}))h_c^{(m)} - \beta \mathrm{v}^{W}_{w_t^{(m)}} \right] \tag{29}$$

Update rule for Parameters

1. Document Vector :

$$(\mathrm{v}^{D}_{d_i^{(m)}})^{(i+1)} = (\mathrm{v}^{D}_{d_i^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^{W}_{w_t^{(m)}} \cdot h_c^{(m)}))\mathrm{v}^{W}_{w_t^{(m)}} - \beta \mathrm{v}^{D}_{d_i^{(m)}} \right] \tag{28}$$

2. Middle Word Vector :

$$(\mathrm{v}^{W}_{w_t^{(m)}})^{(i+1)} = (\mathrm{v}^{W}_{w_t^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^{W}_{w_t^{(m)}} \cdot h_c^{(m)}))h_c^{(m)} - \beta \mathrm{v}^{W}_{w_t^{(m)}} \right] \tag{29}$$

3. Context Word Vectors :

$$(\mathrm{v}^{W}_{w_{t+j}^{(m)}})^{(i+1)} = (\mathrm{v}^{W}_{w_{t+j}^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^{W}_{w_t^{(m)}} \cdot h_c^{(m)}))\lambda_{t+j}\mathrm{v}^{W}_{w_t^{(m)}} - \beta \mathrm{v}^{W}_{w_{t+j}^{(m)}} \right] \tag{30}$$

Update rule for Parameters

1. Document Vector :

$$(\mathrm{v}^D_{d_i^{(m)}})^{(i+1)} = (\mathrm{v}^D_{d_i^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)}))\mathrm{v}^W_{w_t^{(m)}} - \beta \mathrm{v}^D_{d_i^{(m)}} \right] \tag{28}$$

2. Middle Word Vector :

$$(\mathrm{v}^W_{w_t^{(m)}})^{(i+1)} = (\mathrm{v}^W_{w_t^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)}))h_c^{(m)} - \beta \mathrm{v}^W_{w_t^{(m)}} \right] \tag{29}$$

3. Context Word Vectors :

$$(\mathrm{v}^W_{w_{t+j}^{(m)}})^{(i+1)} = (\mathrm{v}^W_{w_{t+j}^{(m)}})^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)}))\lambda_{t+j}\mathrm{v}^W_{w_t^{(m)}} - \beta \mathrm{v}^W_{w_{t+j}^{(m)}} \right] \tag{30}$$

4. Neural Network Weights :

$$\lambda_{t+j}^{(i+1)} = \lambda_{t+j}^{(i)} + \gamma \left[(Y^{(m)} - \sigma(\mathrm{v}^W_{w_t^{(m)}} \cdot h_c^{(m)}))(\mathrm{v}^W_{w_t^{(m)}} \cdot \mathrm{v}^W_{w_{t+j}^{(m)}}) - \beta \lambda_{t+j} \right] \tag{31}$$

Algorithm for learning Document Representations

1: **Input:** D, k, c, n, β, γ, $epochs$
2: **Output:** Document Vectors D, Word Vectors W

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*

Output: Document Vectors D, Word Vectors W

$V \leftarrow Extractfrom(D)$

$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$

$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$

$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$

$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow Extractfrom(D)$
$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ $\triangleright$ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ $\triangleright$ $2c$-sized vector of 1s
while *epochs* ≥ 1 **do**
    for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
        $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$

Algorithm for learning Document Representations

Input: D, k, c, n, β, γ, *epochs*
Output: Document Vectors D, Word Vectors W
$V \leftarrow Extractfrom(D)$
$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s
while *epochs* ≥ 1 **do**
  for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
    $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$
    $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma \left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta \mathrm{v}^D_{d_i} \right]$

Algorithm for learning Document Representations

Input: $D, k, c, n, \beta, \gamma, epochs$

Output: Document Vectors D, Word Vectors W

$V \leftarrow Extractfrom(D)$

$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$

$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$

$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$

$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s

while $epochs \geq 1$ **do**

  for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**

    $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$

    $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma \left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta \mathrm{v}^D_{d_i} \right]$

    $\mathrm{v}^W_{w_t} \leftarrow \mathrm{v}^W_{w_t} + \gamma \left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))h_c - \beta \mathrm{v}^W_{w_t} \right]$

Algorithm for learning Document Representations

Input: $D,\ k,\ c,\ n,\ \beta,\ \gamma,\ epochs$

Output: Document Vectors D, Word Vectors W

$V \leftarrow Extractfrom(D)$

$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$

$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$

$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$

$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s

while $epochs \geq 1$ **do**

  for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**

    $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$

    $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta\mathrm{v}^D_{d_i}\right]$

    $\mathrm{v}^W_{w_t} \leftarrow \mathrm{v}^W_{w_t} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))h_c - \beta\mathrm{v}^W_{w_t}\right]$

    for all $j \in \{t-c, \ldots, t-1, t+1, \ldots, t+c\}$ **do**

      $\mathrm{v}^W_{w_{t+j}} \leftarrow \mathrm{v}^W_{w_{t+j}} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\lambda_{t+j}\mathrm{v}^W_{w_t} - \beta\mathrm{v}^W_{w_{t+j}}\right]$

Algorithm for learning Document Representations

Input: $D, k, c, n, \beta, \gamma, epochs$
Output: Document Vectors D, Word Vectors W
$V \leftarrow Extractfrom(D)$
$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$
$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$
$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$
$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s
while $epochs \geq 1$ **do**
  for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**
    $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$
    $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta\mathrm{v}^D_{d_i}\right]$
    $\mathrm{v}^W_{w_t} \leftarrow \mathrm{v}^W_{w_t} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))h_c - \beta\mathrm{v}^W_{w_t}\right]$
    for all $j \in \{t-c, \ldots, t-1, t+1, \ldots, t+c\}$ **do**
      $\mathrm{v}^W_{w_{t+j}} \leftarrow \mathrm{v}^W_{w_{t+j}} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\lambda_{t+j}\mathrm{v}^W_{w_t} - \beta\mathrm{v}^W_{w_{t+j}}\right]$
      $\lambda_{t+j} \leftarrow \lambda_{t+j} + \gamma\left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))(\mathrm{v}^W_{w_t} \cdot \mathrm{v}^W_{w_{t+j}}) - \beta\lambda_{t+j}\right]$

Algorithm for learning Document Representations

Input: $D, k, c, n, \beta, \gamma, epochs$

Output: Document Vectors D, Word Vectors W

$V \leftarrow Extractfrom(D)$

$\mathrm{D} \leftarrow random(\mathbb{R}^{k \times |D|})$

$\mathrm{W} \leftarrow random(\mathbb{R}^{k \times |V|})$

$\mathcal{T} \leftarrow Extractfrom(D, c, n)$ ▷ $|\mathcal{T}| = M + nM$

$\Lambda \leftarrow \mathbf{1}^{2c}$ ▷ $2c$-sized vector of 1s

while $epochs \geq 1$ **do**

  for all $\{d_i, w_{t-c}, \ldots, w_{t+c}, Y\} \in \mathcal{T}$ **do**

    $h_c \leftarrow \mathrm{v}^D_{d_i} + \lambda_{t-c}\mathrm{v}^W_{w_{t-c}} + \ldots + \lambda_{t+c}\mathrm{v}^W_{w_{t+c}}$

    $\mathrm{v}^D_{d_i} \leftarrow \mathrm{v}^D_{d_i} + \gamma \left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\mathrm{v}^W_{w_t} - \beta \mathrm{v}^D_{d_i} \right]$

    $\mathrm{v}^W_{w_t} \leftarrow \mathrm{v}^W_{w_t} + \gamma \left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))h_c - \beta \mathrm{v}^W_{w_t} \right]$

    for all $j \in \{t-c, \ldots, t-1, t+1, \ldots, t+c\}$ **do**

      $\mathrm{v}^W_{w_{t+j}} \leftarrow \mathrm{v}^W_{w_{t+j}} + \gamma \left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))\lambda_{t+j}\mathrm{v}^W_{w_t} - \beta \mathrm{v}^W_{w_{t+j}} \right]$

      $\lambda_{t+j} \leftarrow \lambda_{t+j} + \gamma \left[(Y - \sigma(\mathrm{v}^W_{w_t} \cdot h_c))(\mathrm{v}^W_{w_t} \cdot \mathrm{v}^W_{w_{t+j}}) - \beta \lambda_{t+j} \right]$

    $epochs \leftarrow epochs - 1$

return D, W

Hyper-parameters of the Model

1. Embedding Dimensionality (k)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)
3. Number of Negative Samples (n)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)
3. Number of Negative Samples (n)
4. Number of Epochs ($epochs$)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)
3. Number of Negative Samples (n)
4. Number of Epochs ($epochs$)
5. Learning Rate (γ)

Hyper-parameters of the Model

1. Embedding Dimensionality (k)
2. Window Size (c)
3. Number of Negative Samples (n)
4. Number of Epochs (*epochs*)
5. Learning Rate (γ)
6. Regularization Constant (β)

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

The task is to assign categories to a new document d_x
To model document category relation

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

The task is to assign categories to a new document d_x
To model document category relation

1. Each $d_i \in D$ is represented using $\mathrm{v}_{d_i}^{D} \in \mathbb{R}^k$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

The task is to assign categories to a new document d_x
To model document category relation

1. Each $d_i \in D$ is represented using $\mathrm{v}_{d_i}^{D} \in \mathbb{R}^k$
2. Represent each $c_i \in C$ using $\mathrm{v}_{c_i}^{C} \in \mathbb{R}^k$

Document Categorization using Logistic Regression

Given,

1. Set of documents, $D = \{d_1, \ldots, d_{|D|}\}$
2. Set of categories, $C = \{c_1, \ldots, c_{|C|}\}$
3. Training Data, $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, $y^{(m)} \in \{0, 1\}$

The task is to assign categories to a new document d_x
To model document category relation

1. Each $d_i \in D$ is represented using $\mathrm{v}_{d_i}^D \in \mathbb{R}^k$
2. Represent each $c_i \in C$ using $\mathrm{v}_{c_i}^C \in \mathbb{R}^k$
3. Learn a probabilistic logistic classifier to assign categories

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

$$P(y = 1|d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}^{D}_{d_i} \cdot \mathrm{v}^{C}_{c_j}) \quad (32)$$

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

$$P(y = 1 | d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}^{D}_{d_i} \cdot \mathrm{v}^{C}_{c_j}) \tag{32}$$

$$P(y = 0 | d_i, c_j, \mathrm{D}, \mathrm{C}) = 1 - \sigma(\mathrm{v}^{D}_{d_i} \cdot \mathrm{v}^{C}_{c_j}) \tag{33}$$

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

$$P(y = 1|d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}^D_{d_i} \cdot \mathrm{v}^C_{c_j}) \tag{32}$$

$$P(y = 0|d_i, c_j, \mathrm{D}, \mathrm{C}) = 1 - \sigma(\mathrm{v}^D_{d_i} \cdot \mathrm{v}^C_{c_j}) \tag{33}$$

$$P(y|d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}^D_{d_i} \cdot \mathrm{v}^C_{c_j})^y (1 - \sigma(\mathrm{v}^D_{d_i} \cdot \mathrm{v}^C_{c_j}))^{1-y} \tag{34}$$

Logistic Classifier for Categorization

Given document category pair, $\{d_i, c_j\}$,

- We build a probabilistic logistic classifier to predict the label y

$$P(y = 1 | d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C}) \tag{32}$$

$$P(y = 0 | d_i, c_j, \mathrm{D}, \mathrm{C}) = 1 - \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C}) \tag{33}$$

$$P(y | d_i, c_j, \mathrm{D}, \mathrm{C}) = \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C})^{y} (1 - \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C}))^{1-y} \tag{34}$$

$$\log P(y | d_i, c_j, \mathrm{D}, \mathrm{C}) = y \log \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C}) + (1 - y) \log(1 - \sigma(\mathrm{v}_{d_i}^{D} \cdot \mathrm{v}_{c_j}^{C})) \tag{35}$$

Learning Category Embeddings

Given the training data $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, learn category embeddings $(\Theta = C)$ by maximizing log-likelihood of training data

$P_{\mathrm{D},\mathrm{C}}(y_m = y^{(m)})$ is a shorthand notation for $P(y_m = y^{(m)} | d_i, c_j, \mathrm{D}, \mathrm{C})$
y_m is the predicted label

Learning Category Embeddings

Given the training data $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, learn category embeddings $(\Theta = C)$ by maximizing log-likelihood of training data

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{36}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{T} \log P_{\mathrm{D,C}}(y_m = y^{(m)}) \tag{37}$$

$P_{\mathrm{D,C}}(y_m = y^{(m)})$ is a shorthand notation for $P(y_m = y^{(m)} | d_i, c_j, \mathrm{D}, \mathrm{C})$
y_m is the predicted label

Learning Category Embeddings

Given the training data $\mathcal{T} = \{d_i^{(m)}, c_j^{(m)}, y^{(m)}\}_{m=1}^{m=T}$, learn category embeddings ($\Theta = C$) by maximizing log-likelihood of training data

$$\hat{\Theta} = \arg\max_{\Theta} \; l(\mathcal{T}, \Theta) \tag{36}$$

$$l(\mathcal{T}, \Theta) = \sum_{m=1}^{T} \log P_{\mathrm{D,C}}(y_m = y^{(m)}) \tag{37}$$

Similar to learning document embeddings, category embeddings updates are given by,

$$(\mathrm{v}^C_{c_j^{(m)}})^{(i+1)} = (\mathrm{v}^C_{c_j^{(m)}})^{(i)} + \gamma \left[(y^{(m)} - \sigma(\mathrm{v}^D_{d_i^{(m)}} \cdot \mathrm{v}^C_{c_j^{(m)}}))\mathrm{v}^D_{d_i^{(m)}} - \beta \mathrm{v}^C_{c_j^{(m)}} \right] \tag{38}$$

$P_{\mathrm{D,C}}(y_m = y^{(m)})$ is a shorthand notation for $P(y_m = y^{(m)} | d_i, c_j, \mathrm{D}, \mathrm{C})$
y_m is the predicted label

Algorithm for learning Document Representations

Algorithm 1 Learning Category Vector Representations

Input: D, C, $\mathcal{T}$, k, β, γ

Output: Category Vectors C

$\mathrm{C} \leftarrow random(\mathbb{R}^{k\times|C|})$

while not converged **do**

  for all $\{d_i, c_j, y\} \in \mathcal{T}$ **do**

    $\mathrm{v}^C_{c_j} \leftarrow \mathrm{v}^C_{c_j} + \gamma \left[(y - \sigma(\mathrm{v}^D_{d_i} \cdot \mathrm{v}^C_{c_j}))\mathrm{v}^D_{d_i} - \beta \mathrm{v}^C_{c_j} \right]$

return C

Advantages of Multinomial Logistic Regression Algorithm

1. Predicting relation between a document-category tuple is $\mathcal{O}(1)$

Advantages of Multinomial Logistic Regression Algorithm

1. Predicting relation between a document-category tuple is $\mathcal{O}(1)$
2. Categories are embedded in the same space as words and documents

Advantages of Multinomial Logistic Regression Algorithm

1. Predicting relation between a document-category tuple is $\mathcal{O}(1)$
2. Categories are embedded in the same space as words and documents
3. Though learns multiple category vectors, exploits the low-rank structure in the document-category relation

Advantages of Multinomial Logistic Regression Algorithm

1. Predicting relation between a document-category tuple is $\mathcal{O}(1)$
2. Categories are embedded in the same space as words and documents
3. Though learns multiple category vectors, exploits the low-rank structure in the document-category relation
4. Easy incorporation of additional relational data of documents for more accurate categorization as shown in Gupta and Singh [7]

Advantages of Multinomial Logistic Regression Algorithm

1. Predicting relation between a document-category tuple is $\mathcal{O}(1)$
2. Categories are embedded in the same space as words and documents
3. Though learns multiple category vectors, exploits the low-rank structure in the document-category relation
4. Easy incorporation of additional relational data of documents for more accurate categorization as shown in Gupta and Singh [7]
5. Usage of SGD makes algorithm completely online

Performance Evaluation : Datasets

1. **Reuters-21578** : Standard dataset for categorization evaluation

	$\|D\|$	$\|C\|$	$\|V\|$	**Data Points**	**Sparsity**
Train Set	7,767	90	39,853	9,585	0.0137
Test Set	3,019	90	39,853	3,745	0.0138

2. **Wikipedia Datasets** : Extracted for 4 top categories

	$\|D\|$	$\|C\|$	$\|V\|$	**Data Points**	**Sparsity**
Physics	4,229	2,999	81,614	14,070	0.0010
Biology	1,604	2,051	63,767	5,908	0.0018
Sports	1,529	2,829	59,058	3,745	0.0008
Mathematics	1,193	1,519	43,398	3,916	0.0013

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories
2. Capitalization in words is preserved

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories
2. Capitalization in words is preserved
3. Numbers are converted to '$\backslash$*num*'

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories
2. Capitalization in words is preserved
3. Numbers are converted to '$\backslash$*num*'
4. Words occurring less than 5 times ignored

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories
2. Capitalization in words is preserved
3. Numbers are converted to '$\backslash num$'
4. Words occurring less than 5 times ignored
5. Elements of document and word vectors are initialized by drawing uniformly from $[-\frac{1}{k}, \frac{1}{k}]$

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories
2. Capitalization in words is preserved
3. Numbers are converted to '$\backslash num$'
4. Words occurring less than 5 times ignored
5. Elements of document and word vectors are initialized by drawing uniformly from $[-\frac{1}{k}, \frac{1}{k}]$
6. Hyper-parameters are fixed using performance on the validation set

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories
2. Capitalization in words is preserved
3. Numbers are converted to '*num*'
4. Words occurring less than 5 times ignored
5. Elements of document and word vectors are initialized by drawing uniformly from $[-\frac{1}{k}, \frac{1}{k}]$
6. Hyper-parameters are fixed using performance on the validation set
7. Noise Distribution for NCE is chosen as $P_n(w) \sim U(w)^{\frac{3}{4}}$

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories
2. Capitalization in words is preserved
3. Numbers are converted to '$\backslash$*num*'
4. Words occurring less than 5 times ignored
5. Elements of document and word vectors are initialized by drawing uniformly from $[-\frac{1}{k}, \frac{1}{k}]$
6. Hyper-parameters are fixed using performance on the validation set
7. Noise Distribution for NCE is chosen as $P_n(w) \sim U(w)^{\frac{3}{4}}$
8. Typically $k = 100$, $c = 2$, $n = 10$, $epochs = 100$ is used

Experimental Setup

1. **Evaluation Criteria** : Micro-averaged F1 score is used to evaluate performance. Micro-averaging considers all predictions equally across categories
2. Capitalization in words is preserved
3. Numbers are converted to '*num*'
4. Words occurring less than 5 times ignored
5. Elements of document and word vectors are initialized by drawing uniformly from $[-\frac{1}{k}, \frac{1}{k}]$
6. Hyper-parameters are fixed using performance on the validation set
7. Noise Distribution for NCE is chosen as $P_n(w) \sim U(w)^{\frac{3}{4}}$
8. Typically $k = 100$, $c = 2$, $n = 10$, $epochs = 100$ is used

For document categorization evaluation, 80% *of the documents are used for training and the rest are equally divided for test and validation purposes*

Baselines

1. **Bag-of-Words** : Most widely used representation with *tf-idf* weighing

Baselines

1. **Bag-of-Words** : Most widely used representation with *tf-idf* weighing
2. **Latent Semantic Indexing** : Most effective dimensionality reduction technique for text

Baselines

1. **Bag-of-Words** : Most widely used representation with *tf-idf* weighing
2. **Latent Semantic Indexing** : Most effective dimensionality reduction technique for text
3. **Word Vector Averaging** : Document representation by averaging word vectors with *tf-idf* weighting

Baselines

1. **Bag-of-Words** : Most widely used representation with *tf-idf* weighing
2. **Latent Semantic Indexing** : Most effective dimensionality reduction technique for text
3. **Word Vector Averaging** : Document representation by averaging word vectors with *tf-idf* weighting
4. **Probabilistic Matrix Factorization** : Simple matrix factorization of the document-category relation matrix

Document Categorization Performance Evaluation Reuters-21578

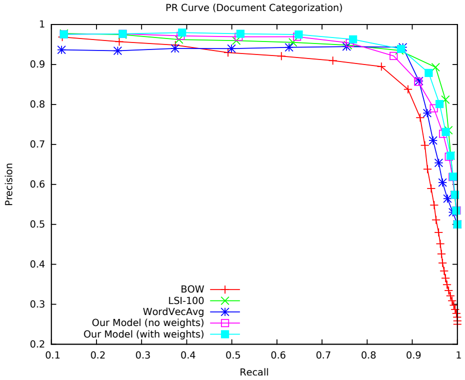
Reuters-21578	P	R	**F1**
BOW	77.8	91.5	84.1
LSI-100	84.8	96.7	90.4
WordVecAvg	94.1	88.1	91.0
SVM (poly) [9]	-	-	86.0
SVM (rbf) [9]	-	-	86.4
CMLF (CRF) [5]	-	-	87.0
Binary-MFoM [4]	-	-	88.4
MC-MFoM [4]	-	-	88.8
Our Model (no weight)	92.1	86.1	89.0
Our Model (with weights)	94.1	89.3	**91.7**

Precision/Recall/F1 for Document Categorization on Reuters-21578

Document Categorization Performance Evaluation Reuters-21578

Reuters-21578	P	R	**F1**
BOW	77.8	91.5	84.1
LSI-100	84.8	96.7	90.4
WordVecAvg	94.1	88.1	91.0
SVM (poly) [9]	-	-	86.0
SVM (rbf) [9]	-	-	86.4
CMLF (CRF) [5]	-	-	87.0
Binary-MFoM [4]	-	-	88.4
MC-MFoM [4]	-	-	88.8
Our Model (no weight)	92.1	86.1	89.0
Our Model (with weights)	94.1	89.3	**91.7**

Precision/Recall/F1 for Document Categorization on Reuters-21578



Document Categorization Performance Evaluation Physics - Wikipedia

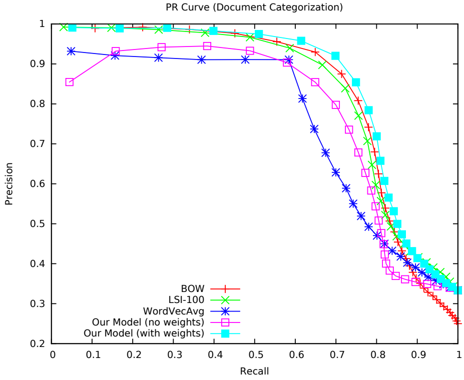
Physics (Wikipedia)	P	R	**F1**
BOW	87.8	70.1	77.9
LSI-100	83.4	69.5	75.8
WordVecAvg	91.0	59.1	71.7
Our Model (no weights)	86.1	64.6	73.8
Our Model (with weights)	88.6	72.4	**79.7**

Precision/Recall/F1 for Document Categorization on Physics dataset

Document Categorization Performance Evaluation Physics - Wikipedia

Physics (Wikipedia)	P	R	**F1**
BOW	87.8	70.1	77.9
LSI-100	83.4	69.5	75.8
WordVecAvg	91.0	59.1	71.7
Our Model (no weights)	86.1	64.6	73.8
Our Model (with weights)	88.6	72.4	**79.7**

Precision/Recall/F1 for Document Categorization on Physics dataset



Document Categorization Performance Evaluation Biology - Wikipedia

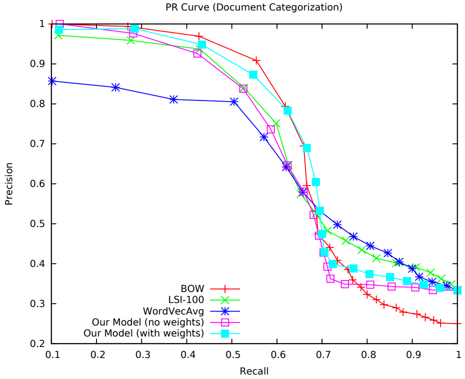
Biology (Wikipedia)	P	R	**F1**
BOW	90.3	59.5	**69.0**
LSI-100	82.1	51.6	63.4
WordVecAvg	79.4	50.4	61.6
Our Model (no weights)	80.3	53.8	64.4
Our Model (with weights)	79.7	59.0	67.8

Precision/Recall/F1 for Document Categorization on Biology dataset

Document Categorization Performance Evaluation Biology - Wikipedia

Biology (Wikipedia)	P	R	F1
BOW	90.3	59.5	**69.0**
LSI-100	82.1	51.6	63.4
WordVecAvg	79.4	50.4	61.6
Our Model (no weights)	80.3	53.8	64.4
Our Model (with weights)	79.7	59.0	67.8

Precision/Recall/F1 for Document Categorization on Biology dataset



Document Categorization Performance Evaluation Mathematics - Wikipedia

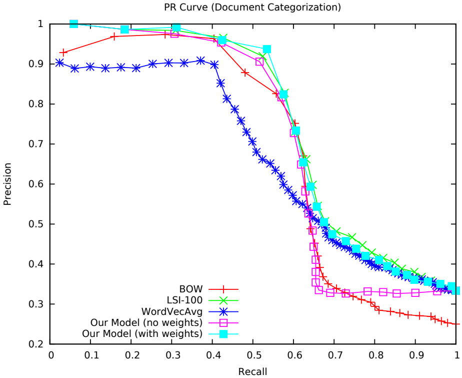
Mathematics (Wikipedia)	P	R	**F1**
BOW	65.6	65.1	65.3
LSI-100	89.7	50.3	64.4
WordVecAvg	90.5	40.3	55.7
Our Model (no weights)	78.4	57.4	66.3
Our Model (with weights)	85.3	56.8	**68.2**

Precision/Recall/F1 for Document Categorization on Mathematics dataset

Document Categorization Performance Evaluation Mathematics - Wikipedia

Mathematics (Wikipedia)	P	R	**F1**
BOW	65.6	65.1	65.3
LSI-100	89.7	50.3	64.4
WordVecAvg	90.5	40.3	55.7
Our Model (no weights)	78.4	57.4	66.3
Our Model (with weights)	85.3	56.8	**68.2**

Precision/Recall/F1 for Document Categorization on Mathematics dataset



Document Categorization Performance Evaluation
Sports - Wikipedia

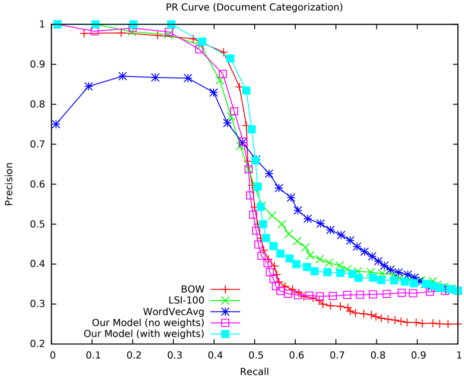
Sports (Wikipedia)	P	R	F1
BOW	91.7	41.3	56.9
LSI-100	91.2	40.1	55.7
WordVecAvg	81.8	37.5	51.4
Our Model (no weights)	80.5	40.1	53.6
Our Model (with weights)	82.1	44.0	**57.3**

Precision/Recall/F1 for Document Categorization on Sports dataset

Document Categorization Performance Evaluation Sports - Wikipedia

Sports (Wikipedia)	P	R	**F1**
BOW	91.7	41.3	56.9
LSI-100	91.2	40.1	55.7
WordVecAvg	81.8	37.5	51.4
Our Model (no weights)	80.5	40.1	53.6
Our Model (with weights)	82.1	44.0	**57.3**

Precision/Recall/F1 for Document Categorization on Sports dataset



Imputing Missing Categories in Wikipedia

1. Real-life databases contain missing information
2. Wikipedia is a large-scale database with non-expert annotators

We evaluate our model on imputing missing categories in the Wikipedia datasets

	Physics			Biology			Mathematics			Sports			**Combined**		
	P	R	**F1**	P	R	**F1**	P	R	**F1**	P	R	**F1**	P	R	**F1**
PMF	73.0	64.3	68.4	72.1	47.5	57.3	41.6	58.2	48.5	51.3	35.6	42.0	63.0	54.8	58.6
LSI-100	59.5	82.3	69.0	49.9	71.6	58.8	47.1	73.0	57.3	43.1	68.2	52.8	52.5	76.3	62.2
BOW	76.1	79.4	77.7	69.7	67.7	68.7	70.9	63.5	67.0	64.8	49.3	56.0	72.5	69.4	70.9
WordVecAvg	88.0	63.5	73.8	80.7	50.3	61.9	71.8	46.7	56.6	87.2	35.4	50.3	84.2	53.4	65.4
Our Model (without weights)	88.6	69.1	77.7	80.5	55.3	65.6	74.3	53.1	61.9	84.7	40.2	54.5	85.4	58.5	69.2
Our Model (with weights)	89.9	74.5	**81.5**	84.9	63.8	**72.9**	79.9	60.7	**69.0**	81.1	45.6	**58.4**	86.3	65.2	**74.3**

Estimating Similarity between Categories and Words

1. We embed words, document and categories in the same k-dimensional space
2. This allows us to estimate similarity between entities non directly related

Category	Nearest Neighbors
Evolutionary Biology	gene, phylogenetics, speciation, ancestor, Darwin, lineage, evolutionary, interbreeding
Statistical Mechanics	ergodicity, Eigenstate, Universality, DMFT, Markovian, Parisi, Combinatorics
Thermodynamics	Convection, ecosystem, Enthalpy, Joule, calorimetric, compressible, Thermodynamic
Trade	import, Pledges, Tariff, Trade, competitiveness, toll, billion, basket, Ditch, Worldwide
Money-FX	Borrowing, franc, banker, Currency, banks, nervous, sideways, Markets, FORWARD
Virology	nucleoside, ribozyme, adenoviruses, Virology, retroviruses, poliovirus, Viroid
Neurobiology	purinergic, cyclase, vertebral, Ehrlich, nexus, steroid, lean, gendered, reticular
Physical Exercise	Fitness, aerobics, metabolic, workout, Exercise, Stretching, pelvic, Physiology, fibers
Algebra	subalgebra, Algebras, nilpotent, adjoints, octonions, bicommutant, diagonalizable
Theoretical Physicists	Dipankar, DSc, Hubert, Aneesur, Uri, Ignaz, Chia, Stig, Diderot, Dannie
Mathematical Physics	covectors, pseudotensor, spacelike, dyadic, Curl, torque, contractions, wavefunctions
Sports Venues	stadion, decoration, tracks, seating, buildings, parcourse, architectural, arenas, circular
Indian Mathematics	utkrama, ecliptic, Siddhanta, Hellenistic, Brahmi, sexagesimal, scribe, Islamic, Sanskrit

Conclusion

1. We presented an unsupervised neural network model that
   - Jointly learns fixed-length low-dimensional distributed vector representations for documents and words
   - Encode semantic content of words and documents in these representations

Conclusion

1. We presented an unsupervised neural network model that
   - Jointly learns fixed-length low-dimensional distributed vector representations for documents and words
   - Encode semantic content of words and documents in these representations
2. We overcome some of the problems with the bag-of-words representations

Conclusion

1. We presented an unsupervised neural network model that
   - Jointly learns fixed-length low-dimensional distributed vector representations for documents and words
   - Encode semantic content of words and documents in these representations
2. We overcome some of the problems with the bag-of-words representations
3. Our model is a log-linear model that uses NCE

Conclusion

1. We presented an unsupervised neural network model that
   - Jointly learns fixed-length low-dimensional distributed vector representations for documents and words
   - Encode semantic content of words and documents in these representations
2. We overcome some of the problems with the bag-of-words representations
3. Our model is a log-linear model that uses NCE
4. We improve state-of-the-art results on multi-label document categorization
   - On the Reuters-21578 dataset we improve by 3.26%
   - On the Reuters-21578 dataset we improve over BOW by 9%

Conclusion

1. We presented an unsupervised neural network model that
   - Jointly learns fixed-length low-dimensional distributed vector representations for documents and words
   - Encode semantic content of words and documents in these representations
2. We overcome some of the problems with the bag-of-words representations
3. Our model is a log-linear model that uses NCE
4. We improve state-of-the-art results on multi-label document categorization
   - On the Reuters-21578 dataset we improve by 3.26%
   - On the Reuters-21578 dataset we improve over BOW by 9%
5. We show the best performance on imputing missing categories in Wikipedia

Conclusion

1. We presented an unsupervised neural network model that
   - Jointly learns fixed-length low-dimensional distributed vector representations for documents and words
   - Encode semantic content of words and documents in these representations
2. We overcome some of the problems with the bag-of-words representations
3. Our model is a log-linear model that uses NCE
4. We improve state-of-the-art results on multi-label document categorization
   - On the Reuters-21578 dataset we improve by 3.26%
   - On the Reuters-21578 dataset we improve over BOW by 9%
5. We show the best performance on imputing missing categories in Wikipedia
6. Learned distributed representations allow semantic similarity estimation

Future Work

1. Improving compositionality of Word Vectors

Future Work

1. Improving compositionality of Word Vectors

2. Joint Document Representation Learning and Document Categorization

Future Work

1. Improving compositionality of Word Vectors
2. Joint Document Representation Learning and Document Categorization
3. Supervised Multi-view Relational Learning

References I

[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] S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua. A mfom learning approach to robust multiclass multi-label text categorization. In *Proceedings of the twenty-first international conference on Machine learning*, page 42. ACM, 2004.

[5] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM, 2005.

[6] 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.

[7] N. Gupta and S. Singh. Collectively embedding multi-relational data for predicting user preferences. *arXiv preprint arXiv:1504.06165*, 2015.

[8] 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.

[9] T. Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

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

References II

[11] 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.

[12] J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8): 1388–1429, 2010.

[13] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273, 2013.

[14] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[15] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer, 2005.

[16] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[17] A. Yessenalina and C. Cardie. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Association for Computational Linguistics, 2011.

[18] F. M. Zanzotto, I. Korkontzelos, F. Fallucchi, and S. Manandhar. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271. Association for Computational Linguistics, 2010.

Thank You!
Questions?

Snacks outside