GloDETC : Global Document Embeddings for Multi-label Text Classification

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

B.Tech. - M.Tech. (Dual Degree)

by

Nitish Gupta
Roll No. : 10327461

under the guidance of

Prof. Harish Karnick
Prof. Rajesh M. Hegde

Department of Electrical Engineering

Indian Institute of Technology Kanpur

April, 2015

CERTIFICATE

It is certified that the work contained in this thesis entitled "*Merging Word Senses*", by *Sumit Bhagwani(Roll No. Y8127515)*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Prof. Harish Karnick)
Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur
Kanpur-208016

June, 2013

Abstract

Abstract goes here

Dedicated to
My Family

Acknowledgement

I wouldn't like to express my sincere gratitude towards my thesis supervisor, Prof. Harish Karnick, for his constant support and encouragement. I am grateful for his patient guidance and advice in giving a proper direction to my efforts. I am also grateful to Prof. Rajesh M. Hegde for giving me the freedom to work on a topic of my liking.

Last, but not the least, I would like to thank my parents and brother for their love and encouragement. Without their support and patience this work would not have been possible.

Nitish Gupta

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Related Work

The task of text classification, i.e. classification of documents into a fixed number of predefined categories has been long studied in-depth for many years now. This multi-class classification problem has further evolved into a multi-label text classification task where each document can belong to multiple, exactly one or no category at all.

Supervised machine learning techniques that learn classifiers to perform this category assignment task can be broken down into two main components, namely, text representation and learning algorithm. Text representation involves converting the documents, that are usually strings of characters, into numerical vectors that are suitable inputs to the learning algorithm while the learning algorithm uses pairs of labeled input text representations and the categories it is belongs in, to learn a model so as to classify new documents into categories.

1.1 Text Representation

Any text-based classification system requires the documents to be represented in an appropriate manner dictacted by the task being performed [Lewis, 1992a]. Moreover, [Quinlan, 1983] showed that the accuracy of the classification task depends as much on the document representation as on the learning algorithm being employed. Different from the data mining task, which deals with structureed documents, text classification deals with unstructured documents that need to be appropriately transformed into numerical vectors, i.e. the need for text representation. In this section we introduce the most effective and widely-used techniques to represent documents for text classification.

1.1.1 Bag of Words

It is found in information retrieval research that word stems work well as representations units for documents and that their ordering in a document is of minor importance for many tasks. This is attributed by the fact that the most widely-used used model to represent documents for the classification task is the *Vector Space Model (VSM)* [Salton and Yang, 1973].

In the Vector Space Model, a document d is represented as a vector in the term/word space, $d = (w_1, w_2, \ldots, w_{|V|})$ where $|V|$ is the size of the vocabulary. Each of the $w_i \in [0, 1]$, represents the weightage of the term i in the document d. This is called the *bag-of-words* model as it ignores word ordering and each document is reduced to a bag of words that it contains or not.

An important requirement of such a representation is that, the terms that help in defining the semantic content of the document and play an important role in classification be given higher weightage than the others. Over the years, there has been much research in the information retrieval field on term weighting schemes. The most important term-weighting techniques are described below :

1. **One Hot Representation** : This is the most trivial representation, where each document is represented by a vector that is size of the vocabulary. Each element in the vector is either a 0 or a 1 to denote the absence or presence of a specific term in the document.

2. **Term Frequency (tf))** : The term frequency representation weighs the terms present in the document relative to their occurence frequency in the document. Hence a document d is represented as, $d = (w_1, w_2, \ldots, w_{|V|})$, where, w_k is the number of times the term k appears in the document d.

3. **Inverse Document Frequency (idf)** : Though using *tf* as a term weighting scheme is a good starting point, it faces a challenge when high frequency terms are not concentrated in a few particular documents but are prevalent in the whole collection. Those terms then stop being characteristic of the semantic content of a few documents and need not be given high weightage. To overcome this problem, Salton and Buckley [1988] suggested a new term weighting called the inverse document frequency (idf). The *idf* weight of a term varies inversely with the number of documents n it belongs to in a collection of total N documents. A typical *idf* vector can be computed as

$$w_k = \log \frac{N}{n} \tag{1.1}$$

4. **Term Frequency Inverse Document Frequency (tf-idf)** : Given the above two term weighing schemes, it is clear that an important term in a document should have high *tf* but a low overall collection frequency (*idf*). This suggests that a reasonable measure for term importance may be then obtained by the *tf* and the *idf* ($tf \times idf$). As we will see in the results section, the *tf-idf* weighed bag-of-words document representation gives one of the best accuracies in the multi-label text classification task.

A common feature in the bag-of-words document representation is the *normalization factor*[Salton and Buckley, 1988] introduced to reduce the effect of varying document lengths and give equal weightage to documents of all lengths when learning the classifier for text categorization. **TODO:** Do we put how normalization is done? Another feature added to the bag-of-words representation is the removal of stop-words (short function words that do not add to the semantic content of the document) and words that occur infrequently to make the document vector more meaningful.

1.1.2 Dimensionality Reduction / Feature Selection

The bag-of-words representation scheme has several drawbacks but the most important drawback it suffers from is that document vectors are very sparse and high dimensional. Typical vocabulary sizes of a moderate-sized document collection ranges from tens to hundereds of thousands of terms which is prohibitively high for many learning algorithms. To overcome this issue of high-dimensional bag-of-words document representations, automatic feature selection is performed that removes uninformative terms according to corpus statistics and constructs new orthogonal features by combining several lower level features (terms/words). Several techniques used in practice are discussed below,

1. **Information Gain** : Information Gain is widely used as a term-goodness criterion in the field of machine learning, mainly in decision trees [Quinlan, 1986] and also in text classification [Lewis and Ringuette, 1994], [Moulinier et al., 1996]. It is a feature space pruning technique that measures the number of bits of information obtained(entropy) for category prediction by knowing the presence or absence of a term in a document. For terms where the information gain was below some predefined threshold are not considered in the document vector representation. The information gain of a term t is defined as

$$G(t) = -\sum_{i=1}^{|C|} P(c_i) \log P(c_i) + P(t) \sum_{i=1}^{|C|} P(c_i|t) \log P(c_i|t) + P(\ t) \sum_{i=1}^{|C|} P(c_i|\ t) \log P(c_i|\ t) \tag{1.2}$$

2. **Mutual Information** : Similar to the Information Gain scheme, Mutual Information estimates the information shared between a term and a category and prunes terms that are below a specific threshold. The mutual information between a term t and a category c is estimated in the following fashion,

$$I(t,c) = \log \frac{P(t \wedge c)}{P(t) \times P(c)} \tag{1.3}$$

To measure the goodness of a term in global feature selection, the category specific scores of a term are combined using,

$$I_{avg}(t) = \sum_{i=1}^{|C|} P(c_i) I(t, c_i) \tag{1.4}$$

3. χ^2 **Statistic** : The χ^2 statistic measures the lack of independence a term t and a category c and can be compared to the χ^2 distribution with one degree of freedom. The term-goodness factor is calculated for each term-category pair and is averaged as above. The major difference between Mutual Information and χ^2 statistic is that the later is a normalized value and the goodness factors across terms are comparable for the same category.

4. **Latent Semantic Indexing (LSI)** : LSI first introduced by Deerwester et al. [1990], is a popular linear algebraic dimensionality reduction technique that uses the term co-occurence statistics to capture the latent semantic structure of the documents and represent them using low-dimensional vectors. It is an efficient technique to deal with synonymy and polysemy. LSI aims to find the best subspace approximation to the original document bag-of-word vector space using Singular Value Decomposition. Given a term-document matrix $X = \left[x_1, x_2, \ldots, x_{|D|}\right] \in \mathbb{R}^{|V|}$, its k-rank approximation as found using SVD, can be expressed as,

$$X = TSD^T \tag{1.5}$$

where, $T \in \mathbb{R}^{|V| \times k}$ and $D \in \mathbb{R}^{|D| \times k}$ are orthonormal matrices called the left and right singular vectors respectively. The matrix $S \in \mathbb{R}^{k \times k}$ is a diagonal matrix of singular values arranged in descending order. The k-dimensional rows of the matrix D contain the dimensionality reduced representations of the $|D|$ documents in the collection. The representations obtained using LSI alleviate the issue of data sparsity and high-dimensionality in bag-of-words representations and also helps unfold the latent semantic structure of the documents.

1.2 Learning Algorithms

Multi-label text classification has seen growing number of statistical learning methods being applied to it. Over the years, various larning algorithms like, Regression models ([Cooper et al., 1994], [Fuhr et al., 1991]), Conditional Random Field ([Ghamrawi and McCallum, 2005]), Nearest Neighbour techniques ([Yang, 1994], [Zhang and Zhou, 2005], [Zhang and Zhou, 2007]), Bayesian classifier and topic modelling ([Lewis and Ringuette, 1994], [McCallum, 1999], [Nigam et al., 2000], [Rubin et al., 2012], [Nigam et al., 1999], [Ueda and Saito, 2002]), SVM ([Joachims, 1998], [Elisseeff and Weston, 2001]), Neural Networks ([Wiener et al., 1995], [Ng et al., 1997]), Decision Trees ([Tong and Appelbaum, 1994]), Online learning algorithms ([Lewis et al., 1996], [Crammer and Singer, 2002]), Non-negative Matrix Factorization ([Liu et al., 2006]) etc. have been used or developed for Multi-label document categorization.

Earlier learning algorithms reduced the problem of multi-label classification into multiple binary classification problems and independently learned binary classifiers for each category. While these algorithms performed well, their drawback of considering correlation among categories led to the development of algorithms that learn a single classifier and jointly classify each document.

Multi-label classification problems can be also be classified into classification-based and ranking-based approaches, where the former assigns each test instance a $|L|$-sized label vector of ones and zeros indicating the presence and absence of labels. In the case of a ranking-based approach, the ranking system outputs the list of labels arranged in the increasing order of a ranking score which is then thresholded at an optimum and the top labels are considered appropriate label assigments for test instances.

Below we describe some of the famous learning algorithms for multi-label text classification,

1.2.1 With Multiple Binary Classifiers

The most common approach of multi-label text classification treats each label independently and learns multiple binary classifiers, one for each category and then assigns to a test document all the categories for which the corresponding classfier says *'yes'*. Below we describe some of the algorithms, in the context of multi-label text classification, that learn multiple independent binary classifiers.

1. **Logistic Regression (LR)** : Introduced by [Hosmer and Lemeshow, 1989], LR is a probabilistic binary classification regression model, that, for binary text classification learns a category weight vector and estimates the probability of a document belonging to the category using dot-product and the logistic link function. LR can be extended for multi-label document classification by learning multiple category vectors, specifically, one for each category. At test time, one would need to query all category vectors for each document to make the category assignments. In our work, we use logistic regression for multi-label text classification. The details for the model are given in **TODO:** Future link to description.

2. **Support Vector Machines (SVM)** : Support Vector Machines ([Cortes and Vapnik, 1995], [Vapnik, 2000]) based on the *Structural Risk Minimization* principle, are universal learners. In their basic form, SVMs learn linear threshold functions to find linear hyperplanes in the input data space to separate data of the two differnt classes. In the case, where data is not linearly separable, SVMs can be plugged-in with appropriate kernel functions to learn polyniomial classifiers, radial basic fucntions etc. For multi-label text classification, training data is treated separately for each category and maximum margin separating hyperplanes are found for each category independently [Joachims, 1998].

   Elisseeff and Weston [2001] study a ranking based variant of SVM, where the positive/negative distance from the separating hyperplane of a specific category is the score assigned to the particular instance for that category.

Their formulation then aims to maximize the margin between the score of a category that belongs to the document and a category that does not belong to do the document. This is also called the Rank-SVM.

3. **Neural Networks (NNet)** : Classification-baed, Neural Network approaches to multi-label text classification were mainly studied by Wiener et al. [1995], developed at Xerox PARC and called NNet.PARC and Ng et al. [1997], called CLASSI. Both neural networks are examples of multiple-classifier based approaches where a separate neural network was trained for each category to make binary classifications. While CLASSI used a linear perceptron approach to classify text into categories, NNet.PARC built a three-layered nonlinear neural network that extends logistic regression by modelling higher order term interactions and hence finding non-linear decision boundaries.

4. **Naive Bayes (NB)** : Naive-bayes as studied in Lewis [1992b] and Lewis and Ringuette [1994], is one of the most effective and simple statistical model for text classification. For multi-label classification, classifiers are learnt so as to estimate $P(C_j = 1|D)$, i.e., the probability that the document, D belongs to the category C_j, for each category. This probability is estimated by estimating the probability $P(W_i = 1|C_j = 1)$, i.e. probability that a particular word appears in the document when it belongs to a particular category. Though this approach makes the assumption of word independence, experiments show that this fast-learning algorithm can yield excellent results.

Although, approaches to multi-label classification discussed above give competitive accuracies in the task, they suffer from inefficiencies due to the following reasons,

- make assumptions of category independence and learn 1-vs-All binary classifiers. It is realized that such assumption would not hold true in most real-life situations. Fine-grained categorization of texts usually involve strongly correlated category classes and information about the presence of one gives information about the presence/absence of many others. For eg. in the sentence,

> *Chicago Board of trade grain traders and analysts voiced a lot of interest in how farmers planned to handle their upcoming spring plantings prompting sales of new crop months of corn and oats and purchases in new crop soybeans in the futures markets*

information from words about the presence of categories like *oats*, *corn* etc. can also aid the prediction of the *agriculture* category which can be boosted using joint classification.

Apart from inefficiencies induced by ignoring category correlations, learning independent classifiers poses other drawbacks, such as, in case of millions of labels, learning millions of high-dimensional classifiers is a computationally expensive. Secondly, the cost of prediction for each test instance would be high as all the classifiers need to be evaluated to make a single prediction.

1.2.2 With Single Joint Classifier

To overcome the difficulties and drawback of learning multiple binary classifiers, researchers have since developed learning algorithms that jointly classify each document into categories it belongs to. Outputs of such algorithms are $|L|$-dimensional label vectors $\boldsymbol{y} \in \{0, 1\}^L$, with $\boldsymbol{y}_l = 1$ if label l is relevant for the particular document. Below we describe algorithms for multi-label text classification that learn a single classifier for assigning all relevant labels to a document jointly.

1. **k-Nearest Neighbor (kNN)** : k-nearest neighbor classification is one of the most effective lazy learning approaches to classification. Given an arbitrary text document input, the algorithm first ranks the nearest neighbors among the training documents using some similarity measure. It then uses the category information of the top-k ranked nearest neighbors to predict the categories of the input test document. One simple approach is to take a weighted average of the label vector of the k-nearest neighbors, weights being the similarity score while estimating document distances. This yields a category ranking for the

test input which can be thresholded to yield binary classifications.

Other approach as devised by Zhang and Zhou [2007] is based on the k-NN and the maximum a posteriori(MAP) principle. Their approach is, given a test instance, to first identify its k-nearest neighbors and then based on the statistical information gained from the label sets of the neighboring instances, use the MAP principle to determine the label set of the given input. The prior probability of label occurences and the posterior probability, $P(C_l = n|l = 1)$ i.e. given a document belongs to label l, exactly n of its k neighbors also belong to the label l is determined from the training instances to utilize the MAP principle.

2. **Linear Least Squares Fit (LLSF)** : LLSF[Yang and Chute, 1992] learns a multivariate regression model automatically from a training set of documents and their categories. Documents are input as vectors in the desired representation and the corresponding output is a $|L|$-dimensional binary label vector. By solving a linear least squares fit on the training pairs of vectors a matrix of word-category regression coefficients is learnt, which defines the mapping from an arbitrary document to a weighted category label vector. This weighted vector can be sorted to yield a ranked list of categories for the input document.

3. **Probabilistic Models** : Generative probabilistic models described in McCallum [1999], Nigam et al. [1999], Ueda and Saito [2002] etc. argue that the words in a document belonging to a multi-category class can be regarded as a mixture of characteristic words related to each of the categories. Therefore, they represent the multi-label nature of the document by specifying each document with a set of mixture weights, one for each class and also indicate that each document is generated by a mixture of word distributions, one distribution for each label. Once the word distributions are learnt using the training data, classification is performed using the Bayes Rule which selects the labels that are most likely to generate the given test document. Hence, along with

giving the information on the labels responsible for generating the document, such models also fill the missing information of which labels were responsible for generating each word.

McCallum [1999] and Ueda and Saito [2002] define a multinomial distribution $\boldsymbol{\theta}_l = \{\theta_{l1}, \theta_{l2}, \ldots, \theta_{l|V|}\}$ over the vocabulary for each label, and the word distribution for a document for a given label vector $\boldsymbol{y}$, is computed by taking a weighted average of the word distributions of the labels that are present in the document. Therefore, if $\boldsymbol{\phi}(\boldsymbol{y}) = \{\phi_1(\boldsymbol{y}), \phi_2(\boldsymbol{y}), \ldots, \phi_2(\boldsymbol{y})\}$ is the required word distribution, it can be representated by,

$$\boldsymbol{\phi}(\boldsymbol{y}) = \sum_{l=1}^{|L|} h_l(\boldsymbol{y})\boldsymbol{\theta}_l \tag{1.6}$$

where $h_l(\boldsymbol{y})$'s are the mixing proportion that add upto 1. The word distributions for each label are found by maximizing the posterior in [Ueda and Saito, 2002] and by employing the Expectation-Maximization algorithm in [McCallum, 1999].

Chapter 2

Distributed Document Embeddings

In this chapter we describe the concept of distributed word and document embeddings and why distributed representations of words and documents are better than one-hot or bag-of-words representations as described in 1.1. We then give a background on different models that learn distributed representations for words in a fully unsupervised manner and finally describe in detail our proposed model for learning distributed embeddings for documents that can be used for multi-label text classification.

2.1 Motivation

TODO: Get in tune to document representations. Say words and documents suffer in the same manner with one-hot or bow representations. Express problems in docs with changing words. Give example of sentence

TODO: Can be tackled with distributed repr. Similarity measures as simple as cos-distance can be introduced in documents. Lets model joint distributions of words with continuous distributions. Words have distributed representaions but not docs.

Words are regarded as atomic symbols in most rule-based and statistical natural language processing(NLP) tasks and hence need the appropriate representation to solve the NLP tasks with greater ease and accuracy. Words are traditionally expressed as one-hot vectors, i.e. as vectors of the size of the vocabulary where exactly one element is 1 and the rest all are zero. Though these representations have been widely used, one-hot representations have a plethora of drawbacks that pose problems and limit the ability of systems to perform better.

1. **Curse of Dimensionality** : One-hot representations lead word vectors to be the size of the vocabulary which often consists of tens to hundereds of thousands of words. Due to this curse of dimensionality, language modelling becomes almost impossible where the number of parameters would grow exponentially with the size of the vocabulary if the words are represented as one-hot vectors.

2. **No Word Similarity** : As words are represented by sparse orthogonal vectors, there is no notion of word similarity that can be introduced. In one-hot representation, the word “symphony” is equally close to the words “bark” and “guitar”. We would want word representations such that they capture the semantic or topical similarity between words.

Due to the problems dicussed above there is a need for more robust, low-dimensional, non-sparse vector representations for words that capture the semantic similarity between them, can be used to model language with continuous distributions and can be used as inputs for various other NLP tasks.

2.2 Background

Distributed word representations are dense fixed-sized feature vectors learnt for words in an unsupervised manner from large text corpus that capture the semantic similarity between words. Each word w_i in the corpus is represented by a vector, $v_{w_i} \in \mathbb{R}^m$, where m usually ranges from $50 - 300$. These dense representations help deal with sparsity and high-dimensionality issues in ont-hot representations and also provide provision for estimating similarities between words; which is as simple as taking the dot-product or calculating the cosine-distance between the vectors.

All of the word vector learning models make use of neural networks ([Bengio et al., 2003], [Mnih and Kavukcuoglu, 2013], [Mikolov et al., 2013b], [Collobert et al., 2011], [Bottou, 2014], [Turian et al., 2010], [Levy and Goldberg, 2014]) but differ in their training objectives.

Below we describe in detail two models to show how models with very different learning objectives and architechture can lead to learning high-quality word vectors.

2.2.1 Neural Probabilistic Language Model (NPLM)

Introduced by Bengio et al. [2003], their model aims to learn distributed word vectors and a probability function that uses these vectors to learn a statistical model of language. In their model, the probability of a word sequence is expressed as the product of conditional probabilities of the next word given the previous ones.

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t|w_1^{t-1}) \tag{2.1}$$

And making the n-gram assumption,

$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-n+1}^{t-1}) \tag{2.2}$$

i.e. the probability of the next word in the sequence is mostly affected by the local context, in this the previous n-words and not the whole past sequence.

Their model maps each word to a m-dimensional vector in a matrix $C \in \mathbb{R}^{|V|\times m}$ and estimates the probability $P(w_t = i|w_{t-n+1}^{t-1})$ i.e. the probability that the t^{th} word in the sequence is w_i. The neural network that is used to estimate this probability using the word vectors is shown in Figure 2.1 For each input sequence, the neural network outputs a vector $y \in \mathbb{R}^{|V|}$, where y_i is the unnormalized log-probability that

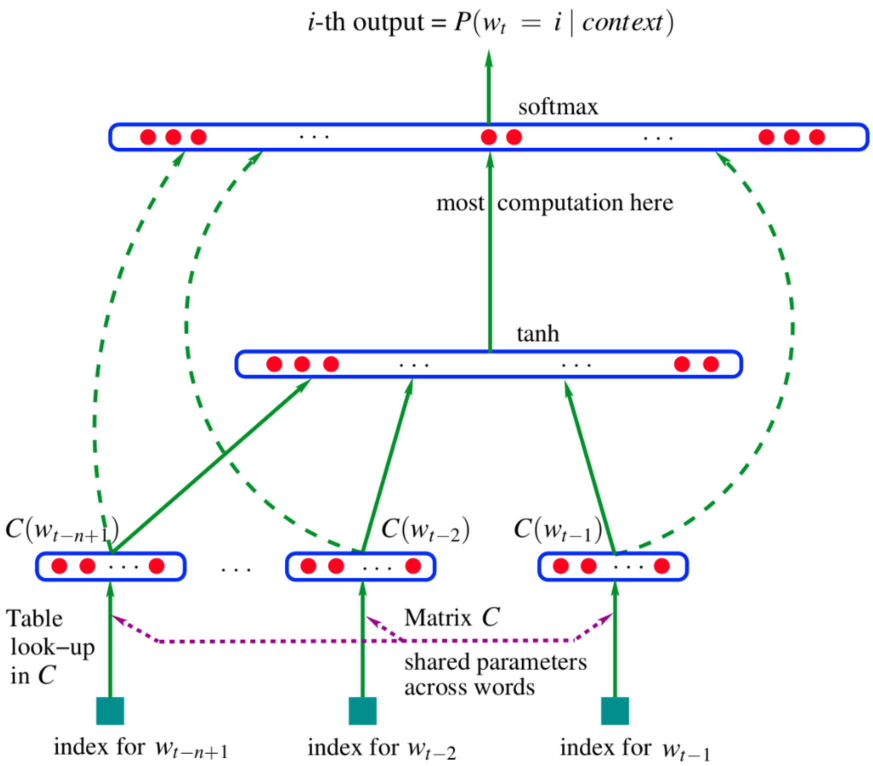

Figure 2.1: Bengio's Neural Network Architechture for Neural Probabilistic Language Model

the t^{th} word in the sequence is w_i.

$$y = b + Wx + Utanh(d + Hx) \tag{2.3}$$

where $tanh$ is the hyperbolic tangent applied to introduce non-linearity and x is the word feature layer activation vector constructed by the concatenation of the context word vectors,

$$x = (C(w_{t-1}), C(w_{t-2}), \ldots, C(w_{t-n+1})) \tag{2.4}$$

The unnormalized log probabilities in y are converted to positive probabilities summing to 1 by using a *softmax* output layer that computes,

$$P(w_t = i|w_{t-1}, \ldots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{2.5}$$

The parameters of the model (b, d, W, U, H) and the word vectors C are estimated by maximizing the log-likelihood of the training corpus.

2.2.2 Log-Linear Models : word2vec

Simple log-linear models are proposed in Mikolov et al. [2013a] as opposed to the non-linear NPLM model to bring down the training time complexity without sacrificing with the quality of the word vectors. The models bring down the complexity of learning vectors by not having a non-linear layer and matrix weighting of the input vectors that are the costliest operations in NPLM. The two models proposed in Mikolov et al. [2013a] are,

- **Continuous Bag-of-Words (CBOW) model** : This model is different from the NPLM in that the projection layer is shared for all words; i.e. all words get projected into the same hidden layer vector (their vectors are averaged). This architechture hence neglects the ordering of the words as opposed to NNLM that uses the concatenation of input vectors for the projection layer. The training criteria in this model is to to classify the current (middle) word given its context. It also uses word sequence from the future to aid this task with the relaxation that the aim is not to learn a language model. The model archtitechture is given in Figure 2.2. The model first computes the hidden

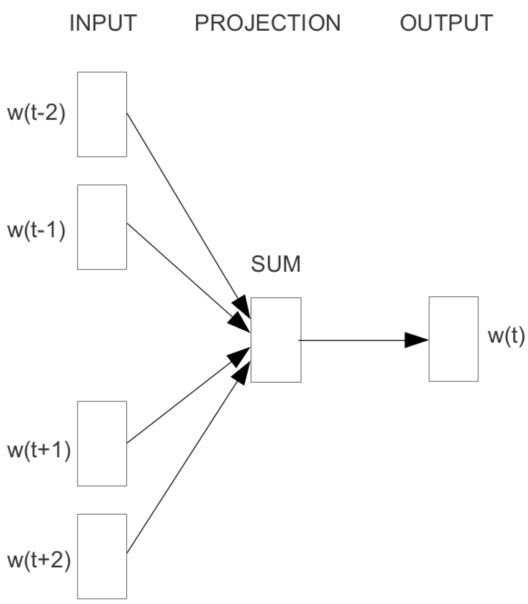

Figure 2.2: Continuous Bag-of-Words Model (CBOW) **TODO:** Add ref?

layer vector h,

$$h(w_{t-k}, \ldots, w_{t+k}) = \frac{w_{t-k} + \ldots + w_{t-1} + w_{t+1} + \cdots + w_{t+k}}{2k} \tag{2.6}$$

where, w_{t-i} is the i-th previous word in the context of the middle word w_t and k is the window length. The neural network then computes a unnormalized log-probability vector y similar to Sec.2.2.1, and uses the *softmax*-classifier to estimate $P(w_t | w_{t-k}, \ldots, w_{t+k})$,

$$y = b + Uh(w_{t-k}, \ldots, w_{t+k}) \tag{2.7}$$

$$P(w_t | w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{2.8}$$

The parameters of the CBOW model, (b, U) and the word vectors (w_i) are learnt by maximizing the average log probability (Eq. 2.8) of the training corpus.

- **Continuous Skip-gram model** : This model is similar to the CBOW model, but instead of predicting the middle word based on the context, it tries to maximize the classification of a word based on another word in the context. More precisely, given each word, the skip-gram model tries to predict words within a certain range before and after the current word. The model architecture is given in Figure 2.3 Formally, given a sequence of words in a context $w_{t-k}, \ldots, w_{t+k}$, the skip-gram model defines $P(w_{t+j} | w_t)$ using the *softmax*-classifier in the following manner,

$$P(w_{t+j} | w_t) = \frac{\exp(v_{w_t} \cdot v_{w_{t+j}})}{\sum_i \exp(v_{w_t} \cdot v_{w_i})} \tag{2.9}$$

The only parameters of the Skip-gram model are the word vectors (v_{w_i}) that are learnt by maximizing the average log probability (Eq. 2.9) of predicting all the context words for all the words in the training corpus.

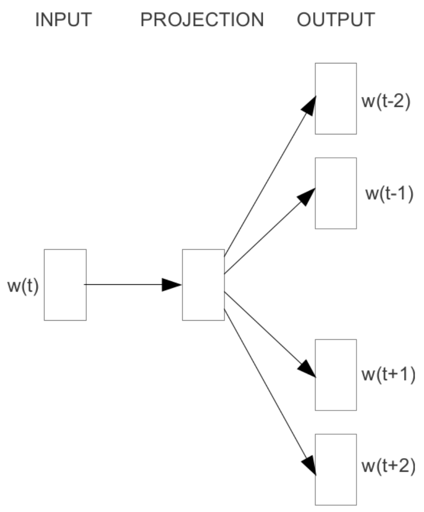

Figure 2.3: Continuous Skip-gram Model **TODO:** Add ref?

The CBOW and the Skip-gram models use the *hierarchical softmax* [Morin and Bengio, 2005] instead of the full softmax to speed-up the learning process.

2.3 Distributed Document Embeddings : Our Approach

Bibliography

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

Léon Bottou. From machine learning to machine reasoning. *Machine learning*, 94 (2):133–149, 2014.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

W Cooper, Aitao Chen, and F Gey. Full text retrieval based on probabilistic equations with coefficients fitted by logistic regression. *NIST SPECIAL PUBLICATION SP*, pages 57–57, 1994.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Koby Crammer and Yoram Singer. A new family of online algorithms for category ranking. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 151–158. ACM, 2002.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6): 391–407, 1990.

André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687, 2001.

Norbert Fuhr, Stephan Hartmann, Gerhard Lustig, Michael Schwantner, Kostas Tzeras, and Gerhard Knorz. *AIR, X: a rule based multistage indexing system for large subject fields*. Citeseer, 1991.

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

David W Hosmer and Stanley Lemeshow. Applied logistic regression. 1989. *New York: Johns Wiley & Sons*, 1989.

Thorsten Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

Omer Levy and Yoav Goldberg. Dependencybased word embeddings. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2: 302–308, 2014.

David D Lewis. Text representation for intelligent text retrieval: a classification-oriented view. *Text-based intelligent systems: current research and practice in information extraction and retrieval*, pages 179–197, 1992a.

David D Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, volume 33, pages 81–93, 1994.

David D Lewis, Robert E Schapire, James P Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–306. ACM, 1996.

David Dolan Lewis. *Representation and learning in information retrieval.* PhD thesis, University of Massachusetts, 1992b.

Yi Liu, Rong Jin, and Liu Yang. Semi-supervised multi-label learning by constrained non-negative matrix factorization. In *Proceedings of the national conference on artificial intelligence*, volume 21, page 421. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

Andrew McCallum. Multi-label text classification with a mixture model trained by em. 1999.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013b.

Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273, 2013.

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

Isabelle Moulinier, Gailius Raskinis, and J Ganascia. Text categorization: a symbolic approach. In *proceedings of the fifth annual symposium on document analysis and information retrieval*, pages 87–99, 1996.

Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *ACM SIGIR Forum*, volume 31, pages 67–73. ACM, 1997.

Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

J Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine learning*, pages 463–482. Springer, 1983.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Timothy N Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. Statistical topic models for multi-label document classification. *Machine Learning*, 88(1-2):157–208, 2012.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Gerard Salton and Chung-Shu Yang. On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351–372, 1973.

Richard M Tong and Lee A Appelbaum. Machine learning for knowledge-based document routing (a report on the trec-2 experiment). *NIST SPECIAL PUBLICATION SP*, pages 253–253, 1994.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

Naonori Ueda and Kazumi Saito. Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems*, pages 721–728, 2002.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2000.

Erik Wiener, Jan O Pedersen, Andreas S Weigend, et al. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th annual symposium on document analysis and information retrieval*, pages 317–332. Citeseer, 1995.

Yiming Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 13–22. Springer-Verlag New York, Inc., 1994.

Yiming Yang and Christopher G Chute. A linear least squares fit mapping method for information retrieval from natural language texts. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 447–453. Association for Computational Linguistics, 1992.

Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest neighbor based algorithm for multi-label classification. In *Granular Computing, 2005 IEEE International Conference on*, volume 2, pages 718–721. IEEE, 2005.

Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.