Learning Distributed Representations of Natural Language Text with Artificial Neural Networks

Boston College Computer Science Senior Thesis

Ali Aslam
Adviser: Sergio Alvarez

Contents

Abstract

Methods in natural language processing(NLP) often make use of handcrafted features or simplifying assumptions of language that work well for many tasks and allow for computational tractability. They can, however, fall short in expressing the full semantic and syntactic richness of natural language that is necessary for solving complex problems of language understanding. To address this shortcoming, artificial neural networks(ANN), robust computational models used for machine learning, have recently been applied to NLP in order to learn continuous-valued vector representations that capture some of the linguistic properties of words and even sentences. These representations have subsequently helped achieve state of the art performance in a number of tasks. In this thesis, I first introduce the motivation and basic concepts behind natural language processing and artificial neural networks. I then describe the representations that ANN approaches to NLP comprise and finally, I mention the latest developments at the cross-section of these two areas, along with the basic intuitions behind them.

1 Overview

This thesis is meant to serve as an introduction to artificial neural network(ANN) applications to natural language processing(NLP). Given that research in both ANNs and NLP extends back several decades and is relatively expansive, only a synopsis of both is provided. The focus is recent work related to using ANNs to learn distributed representations of words, their properties, and also various NLP tasks in which these representations are underlying components. An effort has been made to provide references to further reading as well.

2 Natural Language Processing

Natural language processing(NLP) aims to program computers to understand natural language and thus perform useful tasks. One of the obvious motivations for giving machines this capability is the sheer amount of natural language text and speech data currently available and the exponential rate at which it is increasing. NLP as one of its many goals attempts to simplify and aid in our comprehension of this data.

NLP is based heavily in the field of *linguistics*, the scientific study of language, and thus it is necessary to have a good grasp of linguistic principles before exploring more complex NLP applications. Linguistics has been divided into a number of distinct subfields. From them, perhaps the most important and most comprehensive are *syntax*, *semantics*, and *pragmatics*. The next few sections in this chapter provide a brief definition of these areas and also describe a few of the main tasks in NLP related to each.

2.1 Syntax

Noam Chomsky, often described as the "father of modern linguistics", defines syntax as "the study of the principles and processes by which sentences are constructed in particular languages" [3]. An important concept in syntax that should be noted is what is called a *grammar*. Grammars are sets of rules that govern the formulation of sentences. There are various types of syntactic structure, but the most common is the structure generated by *context-free grammars*.

Overall, there exists a substantial amount of formal theory for the study of syntax in linguistics. Many of these details are left out here. A much more comprehensive introduction is given in chapters 5 and 12 of [11].

The most common challenge related to syntax in NLP is what is called parsing. Parsing is assigning a syntactic structure to a sentence. In other words, it is to classify words and phrases into categories such as singular or mass noun(NN),

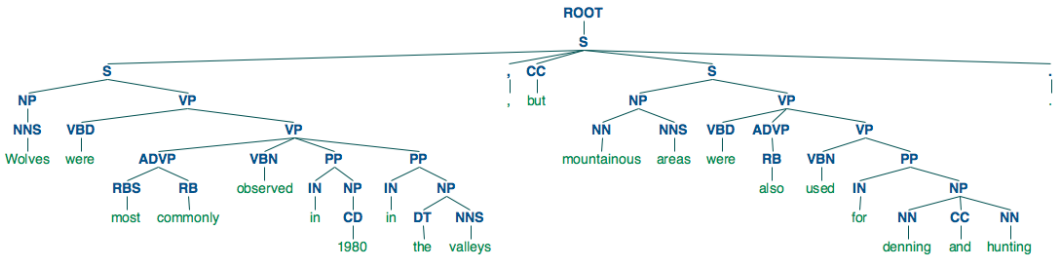


Figure 2.1: A parse tree drawn using NLTK [2].

personal pronoun(PRP), noun phrase(NP), verb phrase(VP), etc. as well as define relationships between them. An example of a typical parse tree is shown in Figure 2.1.

2.2 Semantics

Semantics is the branch of linguistics devoted to meaning. A distinction between syntax and semantics is necessary because sentences of very different syntactic structure may have the same meaning, as shown in Figure 2.2, and vice versa. All of the sentences in the figure convey the same meaning except for minor subtleties, and yet have different syntactic structures or parse trees.

(a) Does Boston College have a computer science major?

(b) Do they have a computer science major at Boston College?

(c) Is computer science a major at Boston College?

(d) Does Boston College offer a major in computer science?

Figure 2.2: Sentences that carry the same meaning, but different syntactic structures

Both semantics and syntax play a significant role in understanding. They are often treated as separate tasks and then in some way brought together for more complex NLP applications.

2.3 Pragmatics

Pragmatics is "the knowledge of the relationship of meaning to the goals and intentions of the speaker" [11]. It tends to deal with what is more informally

called context and how context affects meaning.

Given the ambiguities and vagueness of natural language, understanding the syntax and semantics of a sentence alone is not sufficient. It is necessary to further distinguish between the multiple semantic meanings and syntactic structures a sentence may carry based on the context. A specific example of why pragmatics is necessary is shown in Figure 2.3.

(a) I cooked waterfowl for her.

(b) I created the (plaster?) duck she owns.

(c) I caused her to lower her head or body.

Figure 2.3: Possible meanings for "I made her duck." [11]. Knowledge of the context in which this statement was made is necessary to understand the intended meaning.

Under both semantics and pragmatics lies what is called *discourse*. Discourse is the study of larger linguistic units such as a series of sentences. It is the study of how sentences relate to one another and the meaning they convey as a collective unit. Discourse contributes to the overall context as well.

2.4 Common Approaches to NLP

Approaches in NLP often make simplifying assumptions of natural language because either specific tasks do not necessitate the extra complexity or to allow for computational tractability.

For example, certain approaches in NLP assume that words and phrases are conditionally independent of one another. This occurs most commonly in text classification and also in some parsing algorithms. Figure 2.4 illustrates the *bag-of-words* model. In this model, word order and relationships are ignored by figuratively throwing the words in a bag and simply taking word count into consideration.

It should be understood that as applications become more and more complex, a deeper understanding of language is necessary. This thesis describe methods in which NLP researchers are practically taking steps to capture the richness of natural language that is often ignored with other methods.

"If I have seen further it is by standing on the shoulders of giants."

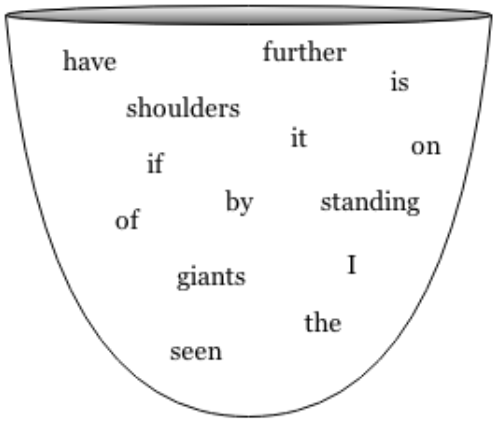


Figure 2.4: A figurative *bag-of-words* model

2.5 Applications

Natural language processing applications are prevalent in everyday technologies. Complex xamples include *conversational agents*, *information retrieval*, *machine translation*, *question answering*, and *speech recognition*.

3 Artificial Neural Networks

Before explaining how *artificial neural networks*(ANNs) have been applied to NLP, an introduction to ANNs is first given in this chapter.

Artificial neural networks are powerful computational models used for what is called *machine learning*. ANNs have a wide range of applications, but have been implemented particularly successfully in computer vision(e.g. recognizing objects in images), speech recognition(i.e. translating spoken words into text) and more recently, NLP.

3.1 Brief Introduction to Machine Learning

Machine learning is a branch of artificial intelligence concerned with the study and implementation of models and algorithms that are capable of learning. Tom Mitchell, a well known researcher in the field, provides a formal definition of learning, explaining that "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [17]. Experience often relates to what are called training examples, i.e. what one uses to train a machine learning model. A class of tasks refers to what exactly one would like a machine learning model to learn. A performance measure is used to determine how well a model performs with respect to the objective. A performance measure not only guides a model in the learning process but also provides a method of comparison between models of different structure and parameters.

It should be noted that a focal point of machine learning is application. There are numerous different models and algorithms used in the field with different approaches that may work well in certain cases but often do not provide absolute theoretical guarantees[25]. Choosing the right method for a specific dataset can at times be more of an art than a science. As such, the entire process of applying machine learning is iterative. The data must be thoroughly understood and tradeoffs related to computational complexity, accuracy, etc.

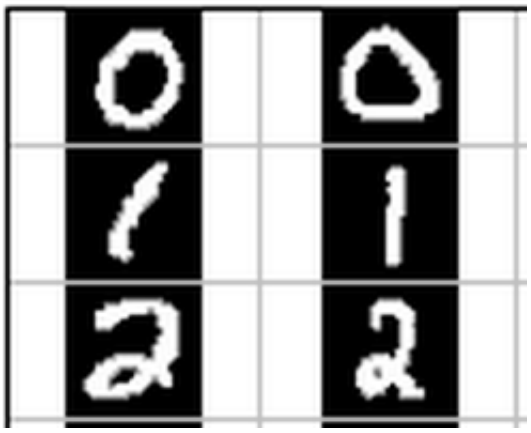


Figure 3.1: Examples of handwritten digits from the MNIST dataset

must be considered. Steps such as *preprocessing*, *feature extraction*, and model training may also be repeated several times.

As a concrete example, consider the *classification* problem of the MNIST dataset(examples shown in Figure 3.1). The task is to build a classifier or a categorizer that accurately determines the digit of a handwritten digit. As a preprocessing step, the dataset has been normalized such that each image is only a black and white pixel representation of a single handwritten digit. Prior to classification, one might also extract useful features from each image. Finally, one would train a model such as an artificial neural network that attempts to *fit* the data in such a way as to maximize the accuracy on unseen or test examples, i.e. *generalize*. If the results were not as expected, one might return to prior steps in the process to make improvements.

For a more detailed introduction to machine learning, see [17] and for advice in practically applying machine learning, see [5].

3.2 Introduction to ANNs

Artificial neural networks are powerful methods for learning, often real-valued, functions.

3.3 Logistic Neurons

The foundational component of an artificial neural network is a logistic neuron, illustrated in Figure 3.2.

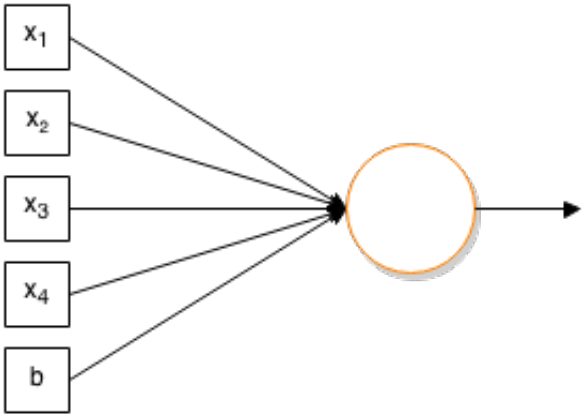


Figure 3.2: A logistic neuron

The neuron takes as its input a vector x, computes a dot product with a set of weights w, and finally applies a non-linearity returning a single real-valued output, also known as an *activation* value. More precisely, a logistic neuron first computes:

$$z = b + \sum_i x_i w_i \tag{3.1}$$

where b is a bias term that serves as an "always on" feature. The logistic neuron then applies the *sigmoidal* non-linearity(Figure 3.3a) to the result z:

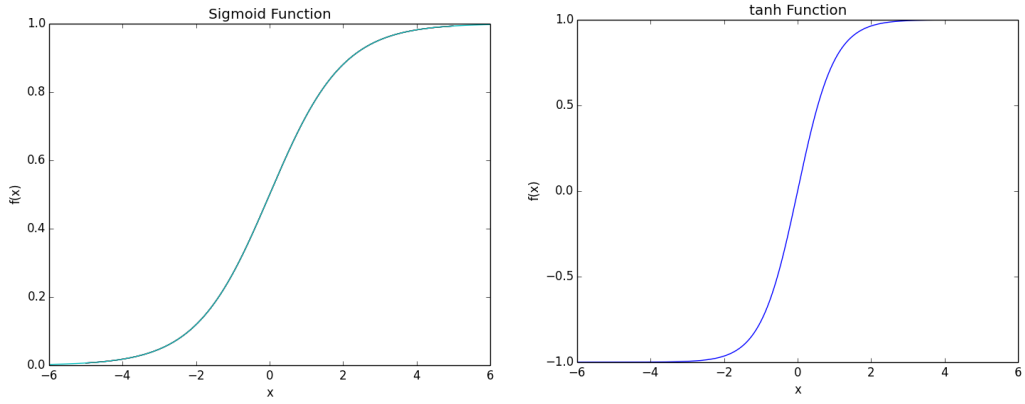
$$y = \frac{1}{1 + e^{-z}} \tag{3.2}$$

A performance measure or cost function for a logistic neuron is also necessary. A common measure is the *squared difference error*:

$$E = \frac{1}{2} \sum_{n \in training} (t^n - y^n)^2 \tag{3.3}$$

t_n is the expected output and y_n is the predicted output of the model for a training example n.

Learning in machine learning often comes down to optimizing a set of parameters. Here, the set of parameters is w and each distinct set of weights defines a function or a *hypothesis*. The objective is to update w with Δw or in other words find a hypothesis such that the error E is minimized.



(a) Sigmoid function

(b) tanh function where $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

Figure 3.3: Two common types of non-linearity functions used with ANNs

One way of minimizing the error is to move in the direction, i.e. update the weights, such that the error is maximally decreased. A visualization of this concept is shown in Figure 3.4.

Recall from calculus that the direction of greatest decrease is the direction opposite to the gradient vector. We use this fact to derive the individual weight-update rule Δw_i for the logistic neuron.

The partial derivative of Equation 3.1 with respect to w_i is:

$$\frac{\partial z}{\partial w_i} = x_i \tag{3.4}$$

The derivative of Equation 3.2 with respect to z is:

$$\frac{dy}{dz} = y(1-y) \tag{3.5}$$

This then allows us to compute the derivative of the cost, Equation 3.3, with respect to each weight w_i:

$$\frac{\partial y}{\partial w_i} = \frac{\partial z}{\partial w_i}\frac{dy}{dz} = x_i y(1-y) \tag{3.6}$$

$$\frac{\partial E}{\partial w_i} = \sum_n \frac{\partial y_n}{\partial w_i}\frac{\partial E}{\partial y_n} = -\sum_n x_i^n y^n(1-y^n)(t^n - y^n) \tag{3.7}$$

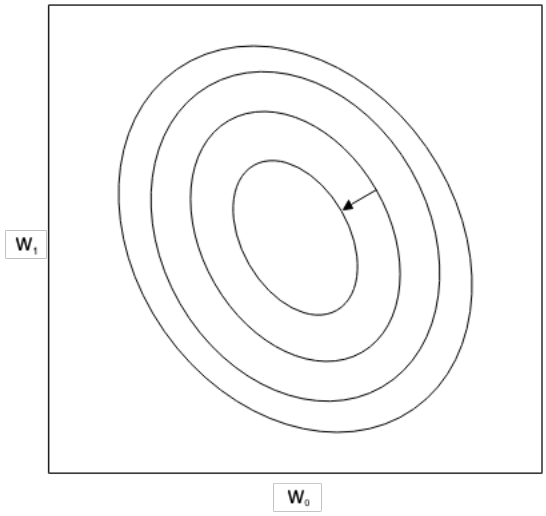


Figure 3.4: Visualizing the error of different hypotheses

3.4 Feedforward Neural Networks

The most basic ANN is the feedforward neural network, shown in Figure 3.5. Note that the outputs from one layer's logistic neurons are passed as input to another's(e.g. $L_2 \rightarrow L_3$).

Learning with respect to ANNs refers to updating the weights of each of the neurons. Due to the fact that the neurons' weights act as a collective unit, the learning algorithm must account for each weight's effect on the output. A popular training algorithm that deals with this complexity is called *Backpropagation* [19]. Despite its simplicity, it is still effectively and widely used for training artificial neural networks.

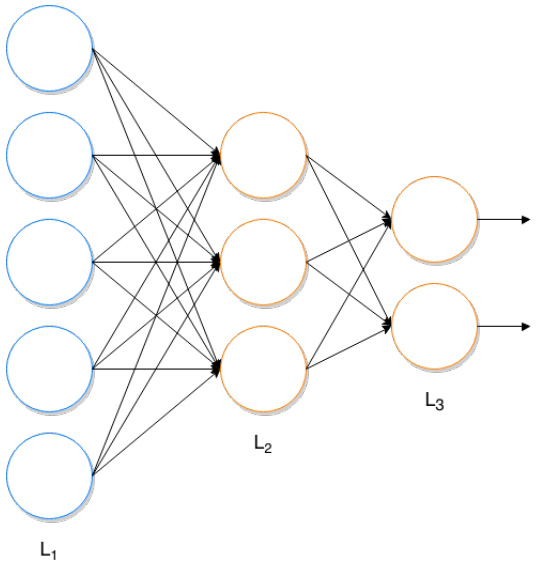


Figure 3.5: A feedforward neural network

3.4.1 Backpropagation

Backpropagation accounts for the multiple layers of an ANN. The algorithm states that $\frac{\partial E}{\partial w_{ij}}$ where w_{ij} is the weight from unit i to unit j is as follows:

$$\frac{\partial E}{\partial z_j} = \frac{dy_j}{dz_j}\frac{\partial E}{\partial y_j} = y_j(1-y_j)\frac{\partial E}{\partial y_j} \tag{3.8}$$

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{dz_j}{dy_i}\frac{\partial E}{\partial z_j} = \sum_j w_{ij}\frac{\partial E}{\partial z_j} \tag{3.9}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}}\frac{\partial E}{\partial z_j} = y_i\frac{\partial E}{\partial z_j} \tag{3.10}$$

This quantiy, $\frac{\partial E}{\partial w_{ij}}$, is also multiplied by a small constant η called the learning rate to get the actual weight update Δw_{ij}. This is used to prevent the ANN from getting caught in local minima. This and other issues are explored in Section 3.7.

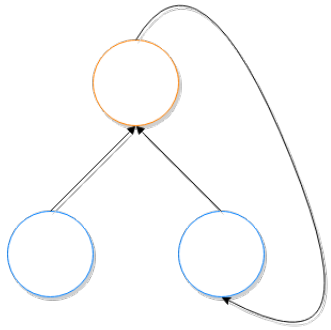


Figure 3.6: A recurrent neural network

3.5 Recurrent Neural Networks

Recurrent Neural Networks(RNNs) are recursive models in which the output is fed back in as input. RNNs are capable of storing "short-term memories" for learning. They are trained using a modified version of Backpropagation called Backpropagation Through Time(BPTT) [24]. The basic idea is to expand the RNN through time and backpropagate the errors just as one would with a feedforward neural network. This concept is illustrated in Figure 3.7.

Learning long sequences with RNNs is typically very difficult with BPTT. The procedure suffers from either vanishing or exploding gradients. In other words, as the RNN is unwrapped through time, the gradients tend to get either very small or very large. Several optimization methods have been developed specifically with this in mind. Nevertheless, RNNs trained for NLP operate almost exclusively at the word level within a window size of a sentence. Due to the fact that sentences are fairly short in terms of the number of words, BPTT has worked well.

3.6 Softmax Layer

In many machine learning tasks, probabilities for class predictions are desired. A common method of representing probabilities with artificial neural networks is the use of a *softmax* layer. It is used to represent a categorical probability distribution and if used is typically the last layer in an ANN. The softmax

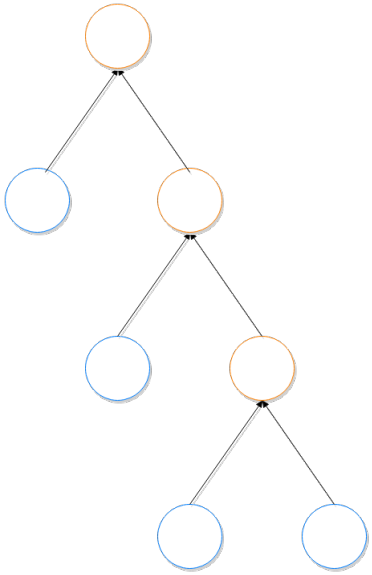


Figure 3.7: A RNN that has been unwrapped 3 time steps. It is equivalent to a Feedforward Neural Network.

function is defined as:

$$y_i = \frac{e^{z_i}}{\sum_{j \in classes} e^{z_j}} \tag{3.11}$$

The partial derivative of y_i with respect to the input z_i is defined as:

$$\frac{\partial y_i}{\partial z_i} = y_i(1 - y_i) \tag{3.12}$$

The cost function *cross-entropy* is used with a softmax layer instead of the squared difference error measure. It is defined as:

$$C = -\sum_j t_j \log y_j \tag{3.13}$$

Its partial derivative with respect to z_i is:

$$\frac{\partial C}{\partial z_i} = \sum_j \frac{\partial C}{\partial y_j} \frac{\partial y_j}{\partial z_i} = y_i - t_i \tag{3.14}$$

3.7 Practical Issues in Training ANNs

Mentioned in this section are a few issues pertaining to training ANNs and as well as common solutions to each.

Learning rate : The learning rate, as mentioned in Section 3.4.1, is an attempt to keep the ANN from getting caught in local minima. There are several methods for choosing learning rates. One recently developed method that has been applied successfully in many real-world applications is *Adagrad* [6]. Adagrad keeps a learning rate for each individual weight and adapts these rates based on historical information.

Regularization : One key concept that should be understood in machine learning is that of *generalization*. The goal of machine learning applications is to generalize to unseen examples outside of the training set. This concept plays a significant role in training artificial neural networks. *Regularization* is meant to prevent overfitting, i.e. fitting noise, and promote generalization. Examples of regularization include but are not limited to: penalizing large weights(*Occam's razor*), *Dropout* [10], and early stopping.

Training Methods : There are several algorithms for training ANNs. From them include: stochastic vs. minibatch vs. batch gradient descent, LBFGS, and Hessian-free optimization. Each has its own drawbacks and advantages [18].

4 Artificial Neural Networks and Natural Language Processing

Although the fields of natural language processing and artificial neural networks independently have a lengthy history, there has been limited work at the cross-section of the two except within the last two decades. This is for various reasons. For one, training artificial neural networks is a very computationally intensive process and so they had previously been used on relatively small datasets. Due to this, early work in applying ANNs to NLP typically involved simpler, handcrafted datasets such as toy grammars [7]. Understanding natural language and performing well for common NLP tasks requires a machine learning model to learn the vast number of structures, meanings and ambiguities of language and this did not seem feasible for ANNs.

However, recently there have been several theoretical advancements to ANNs in terms of regularization and training as well as computational improvements that have drastically reduced the amount of time required for training and have improved the accuracy of ANN models.

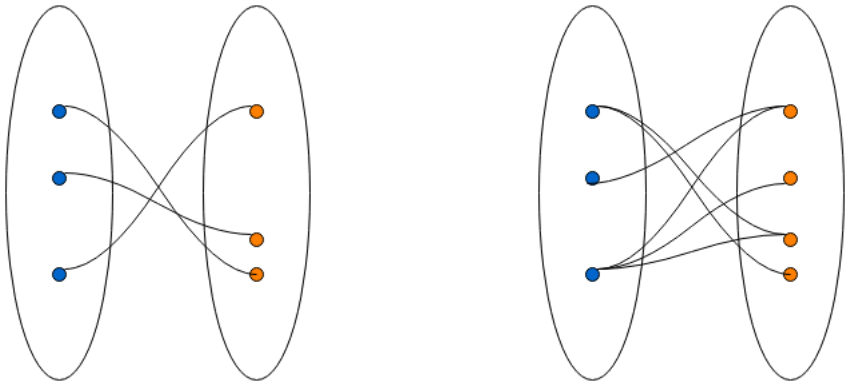
4.1 Motivation

Evidence seems to suggest that artificial neural networks are capable of mimicking, to a certain degree, the language processing of the human brain. It has been shown that artificial neural networks can be trained to resemble the underlying processes of the brain in certain tasks. Hinton et al. demonstrated that when an ANN is trained to read and its weights are then randomly perturbed, it behaves very similarly to someone with acquired dyslexia [9]. The random perturbations are a parallel to what an acquired dyslexic would experience from brain injury.

Although not entirely conclusive, this suggests that ANNs are a natural choice for machine learning tasks such as those in computer vision and NLP that the human brain performs naturally.

4.2 Representing Words as Continuous-Valued Vectors

The primary component used in recent ANN applications to NLP involves *distributed representations* of words. A distributed representation differs from what is called a *local representation* in that each feature in a local representation is mutually exclusive. Also, a distributed representation denotes a many-to-many mapping(Figure 4.1b) while a local representation denotes a one-to-one mapping(Figure 4.1a). The focus here is on representations of natural language, but one can read in more detail about this area in general in [8].



(a) A one-to-one mapping

(b) A many-to-many mapping

Figure 4.1: Mapping comparison

As illustrated in Figure 4.2a, an example of a local representation for a word is a vector in which only one feature is *on*, i.e. 1. In this representation, each individual word maps to only one feature and therefore only n words can be described with vectors of length n. This is obviously very inefficient with respect to memory as each word vector is of length $|V|$, where V represents the vocabulary.

Alternatively, each feature in a vector could signify a specific attribute of words, such as gender, plurality, transitivity, etc. and whether this characteristic is on. With n binary features, it would be possible to describe 2^n words. This is an example of a distributed representation(Figure 4.2b) and with it, one can model componential structure along with a stronger notion of similarity.

Manually assigning binary features, however, is not feasible for large vocabularies. Instead, one could train an artificial neural network to discover features automatically and map a word into a feature space. These distributed rep-

$$\begin{bmatrix} \cdots & 0 & 0 & 1 & 0 & 0 & 0 & \cdots \end{bmatrix}$$

(a) A local representation denoted as a one-hot encoding in which only one feature is "on"

$$\begin{bmatrix} \cdots & 1 & 0 & 1 & 0 & 1 & 1 & \cdots \end{bmatrix}$$

(b) A distributed representation denoted as a binary feature vector in which many features are "on"

$$\begin{bmatrix} \cdots & 0.32 & 0.57 & -0.18 & -0.83 & 0.46 & -0.24 & \cdots \end{bmatrix}$$

(c) A distributed representation denoted as a continuous-valued vector

Figure 4.2: Local vs. Distributed Representations

resentations would be continuous-valued(Figure 4.2c) as that simplifies the problem of optimization/learning.

The resulting word distribution is also called a *language model*. By doing this, it is hoped that these vectors capture semantic and syntactic similarity between words for further use in ANNs and other machine learning models. Such similarity can be observed in Figure 4.3 and is explored further in Section 4.2.3.

4.2.1 Methods for Learning Language Models

Language models typically try to approximate $P(w_t|w_1, w_2, \cdots, w_{t-1})$. This is the probability of a word w_t given the entire prior context, i.e. all words that came before w_t. Models may instead use a fixed context size n and attempt to approximate $P(w_t|w_{t-n}, w_{t-n+1}, \cdots, w_{t-1})$.

One of the fundamental works in using ANNs to learn word representations was the Neural Probabilistic Language Model [1]. Another well known paper described use of a *convolutional neural network*, an ANN capable of handling variable sized inputs, to learn word representations along with trying to solve a variety of different NLP tasks [4]. More recently, there have also been language models based on recurrent neural networks [12]. The specific model described below is the *skip-gram model*.

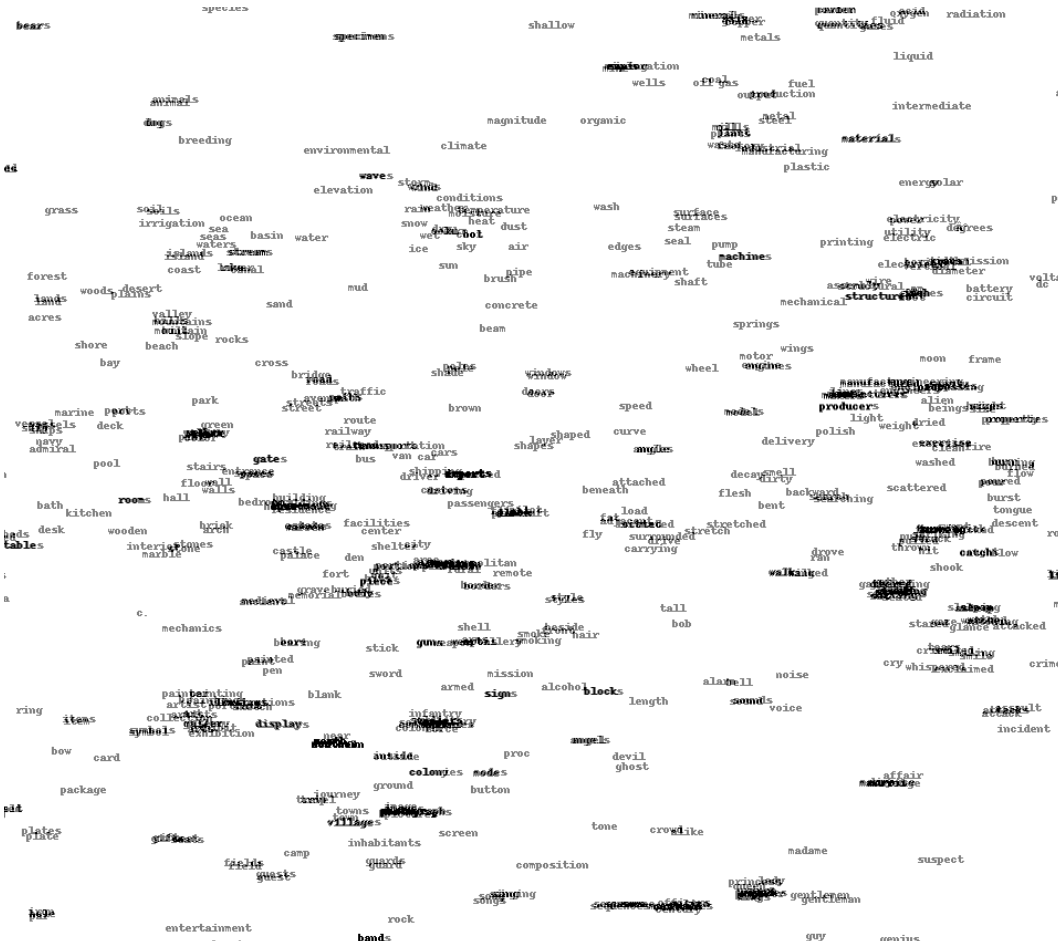


Figure 4.3: A sample of word vectors trained on 100B words from Google^{TM} News plotted to illustrate their linguistic similarity. The vectors were first mapped to $\mathbb{R}^2$ using t-SNE [23] and then plotted with the Python Imaging Library.

4.2.2 Continuous Skip-gram Model

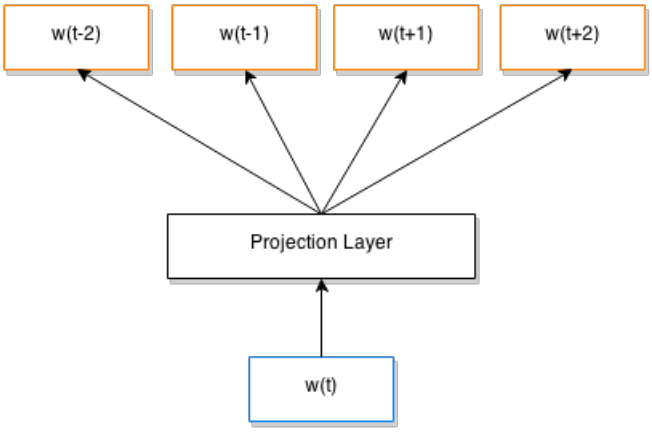


Figure 4.4: The continuous skip-gram model [14]

What differentiates the skip-gram model(Figure 4.4) from previous architectures is that it is strikingly fast to train and that it was developed to scale well to large quantities of data [14]. Unlike other models which are theoretically more powerful, the skip-gram model is a simpler log-linear model. The model tries to find word representations for predicting words both prior to and after w_t. The underlying idea is that words in language are defined according to the contexts in which they are used. The skipgram model objective is to maximize the following:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{4.1}$$

where $p(w_{t+j}|w_t)$ is defined using the definition of softmax(Section 3.6)

$$p(w_O|w_I) = \frac{\exp(v'_{w_O}\cdot v_{w_I})}{\sum_{w=1}^{W}\exp(v'_w\cdot v_{w_I})} \tag{4.2}$$

Note that each word has both a word vector and a context vector associated to it. Thus a word such as *jump* would have separate vectors used depending on whether it is used within a context(v'_{w_O}) or not(v_{w_I}).

Also note that computing the denominator in Equation 4.2 is a time consuming computation given that it is a sum over all words in the vocabulary. To handle

this, suggestions have been made to use what is called a *hierarchical softmax*. Details of it and other methods can be found in the referenced literature.

4.2.3 Linguistic Properties of Word Vector Representations

Word vectors learned by artificial neural networks display interesting properties [15]. Several of these are described below.

Word Clustering: Semantically and syntactically similar words are clustered together in the feature space. This is visualized in Figure 4.3.

Analogies: The word vectors can be used to complete analogies. For example, given the analogy "man is to king as woman is to _", one would first calculate the difference d between the vectors of "man" and "king". Then one would add this difference to the vector of "woman" and find that its nearest neighbor is in fact the vector of "queen".

$$Vector(\text{“}man\text{”}) - Vector(\text{“}king\text{”}) = d \tag{4.3}$$

$$d + Vector(\text{“}woman\text{”}) \approx Vector(\text{“}queen\text{”}) \tag{4.4}$$

Language Mapping: A mapping can be learned between word vector spaces trained for different languages. This is useful in statistical machine translation especially when words and phrases are missing. The method, even with a simple linear mapping, is highly effective as demonstrated by Mikolov, et al. [13].

4.3 Methods of Compositionality

Representations of words that carry semantic and syntactic meaning are a step in the right direction, but language is not just about single words. Text and speech is composed of phrases and sentences where meaning is conveyed through the words that they comprise as well as the rules that combine them. The next few sections describe how learned word vectors can be used to compose longer expressions.

A number of papers have been published in this area [16]. The simplest composition function is addition. This captures some information, e.g. $Vector(\text{“}capital\text{”}) + Vector(\text{“}Russia\text{”}) \approx Vector(\text{“}Moscow\text{”})$, as well as some syntactic and semantic properties of other short phrases and sentences with actual meaning.

Another method of compositionality is a weighted sum:

$$s = \sum_{i=1}^{n} w_i x_i \tag{4.5}$$

The weight vector w may be learned. Using inverse document frequency(idf) for w is also a possibility. The idea is that words that occur commonly throughout the entire document set would be given a lower weight because words such as "the", "it", etc. add less to the overall meaning of a sentence. Another option is to assign weights to particular *part of speech*(POS) tags with the assumption that nouns and verbs play a much larger role in conveying meaning than for example determiners.

Nevertheless, the methods described above are simply linear combinations that fail to take into account the rules that combine words together. Natural language is recursive and as such the rules that combine words are recursive in their nature and change from sentence to sentence. In the next few sections several models are briefly described that have successfully used word representations trained from ANNs and take into account natural language's recursivity.

4.4 Parsing

A state of the art parsing(Section 2.1) model was recently developed using an RNN and word vector representations called a *Compositional Vector Grammar*(CVG) [21]. It, as stated in the paper, is basically an efficient reranker. The model first produces k highly probable parse trees for a given sentence using a *probabilistic context free grammar*(PCFG). The RNN is then searches this limited hypothesis space and predicts the most likely parse tree for a given sentence based on the parameters it has learned.

One of the aims of using an RNN is to account for the limited number of labeled parse trees available for training. One way in which the RNN does this is by taking advantage of the syntactic and semantic information stored in word vector representations. It draws conclusions for new words and phrases that it had not seen during training based on words it had already seen that are close in the feature space.

The CVG also takes combinations of syntactic categories into account by storing a different set of weights for each combination, e.g. DT-NP, ADJP-NP. Through this, the RNN learns a soft notion of a *head word*. The head of a phrase is the word that determines the syntactic category of the phrase. For

example, the RNN learned to recognize that a noun phrase(NP) plays a much larger role in determining the category of a phrase than a determiner(DT).

Although the objective of the CVG is to parse sentences, it outputs distributed representations of input sentences as well. However, given that the RNN is trained specifically for parsing, the representations capture mostly syntactic information. Nearest neighbors in the feature space have very similar syntactic structures, but may at the same time have very different semantic meanings. An example of this is shown in Figure 4.5.

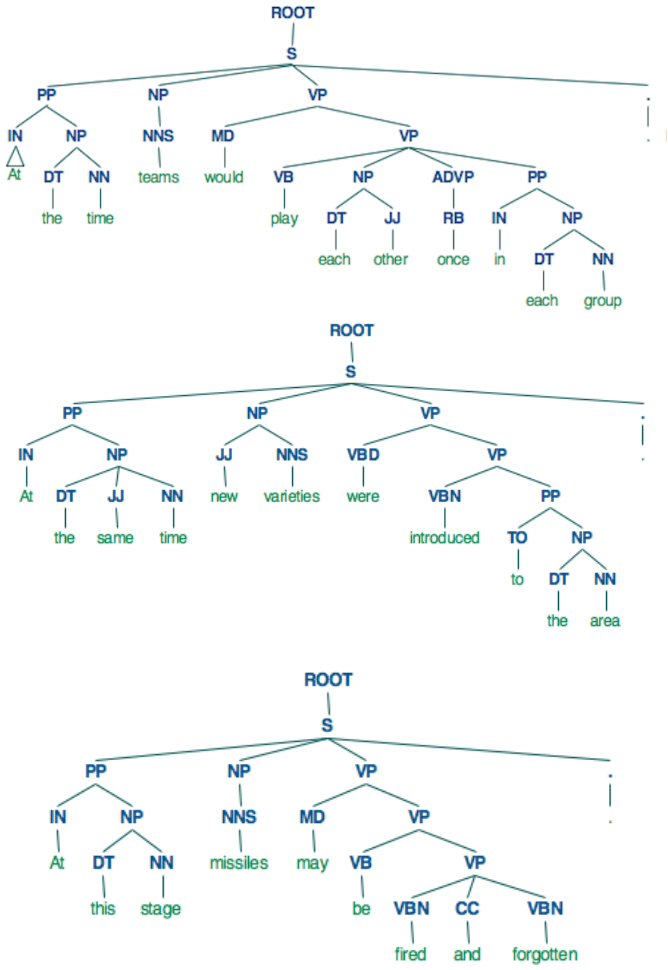


Figure 4.5: Nearest syntactic neighbors displayed with NTLK [2]

4.5 Sentiment Analysis

A subsequent model called a *Recursive Neural Tensor Network*(RNTN) also took advantage of word representations, but for sentiment analysis. The RNTN does not take pre-trained word vector representations as input, but rather learns word representations on its own.

Along with the ability to handle the recursivity of language, the RNTN also learns a *tensor* or composition function. This tensor directly relates words and thus captures how words interact and affect one another.

A sentence is first parsed in order to obtain the syntactic structure that describes which words and phrases hold relationships. Each binary relationship is pass through the RNTN and a sentiment score is computed at each step when iterating through the parse tree.

The RNTN is capable of storing and understanding negation as well. Simpler models, because they assume that words are conditionally independent of one another, fail to learn negations of positive attributes.

4.6 Additional Applications

There also exist applications in common sense knowledge base reasoning [22] and statistical machine translation [20].

5 Conclusions and Future Work

This research area is currently very active and is progressing quickly. It continues to attract more attention from researchers in different subfields in NLP.

Future areas of emphasis are compositionality of word vector representations and also potential applications of these representations. In terms of compositionality, methods that capture longer semantic meaning and also context are currently in their early stages.

Overall though, research in the field is very promising and it will be interesting to see how artificial neural networks are applied to natural language processing in the future.

Bibliography

[1] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

[3] Noam Chomsky. *Syntactic structures*. Walter de Gruyter, 2002.

[4] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[5] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[7] Jeffrey L Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225, 1991.

[8] Geoffrey E Hinton. Distributed representations. 1984.

[9] Geoffrey E Hinton, David Plaut, and Tim Shallice. Simulating brain damage. *Scientific American*, page 77, 1993.

[10] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[11] James H Martin and D Jurafsky. *Speech and language processing*. prentice hall, 2008.

[12] Tomas Mikolov. Recurrent neural network based language model.

[13] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[14] 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, 2013.

[15] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751, 2013.

[16] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition.

[17] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[18] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272, 2011.

[19] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning representations by back-propagating errors*.

[20] Holger Schwenk, Anthony Rousseau, and Mohammed Attik. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics, 2012.

[21] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer, 2013.

[22] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[23] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

[24] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[25] David H Wolpert and William G Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.